TOWARDS COSMOPOLITAN ROBOTS: INTELLIGENT NAVIGATION IN EXTENDED MAN-MADE ENVIRONMENTS

Ronald C. Arkin

COINS Technical Report 87-80

September 1987

Research supported in part by: Defense Advanced Research Projects Agency DACA76-85-C-008, The National Science Foundation DCR-8318-776, and The General Dynamics Corporation DEY-601550.

TOWARDS COSMOPOLITAN ROBOTS: INTELLIGENT NAVIGATION IN EXTENDED MAN-MADE ENVIRONMENTS

A Dissertation Presented

By

RONALD CRAIG ARKIN

Submitted to the Graduate School of the

University of Massachusetts in partial fulfillment

of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 1987

Department of Computer and Information Science

Towards Cosmopolitan Robots:
Intelligent Navigation in Extended Man-made Environments

A Dissertation Presented

by

Ronald Craig Arkin

Approved as to style and content by:

Dr. Edward M. Riseman, *Committee Chair*

Dr. Allen R. Hanson, *Member*

Dr. Michael A. Arbib, *Member*

Dr. Victor R. Lesser, *Member*

Dr. Steven P. Levitan, *Outside Member*

Dr. W. Richards Adrion, *Department Chair*

Computer and Information Science Department

Research supported in part by:
Defense Advanced Research Projects Agency
DACA76-85-C-008,
The National Science Foundation
DCR-8318-776,
and The General Dynamics Corporation
DEY-601550.

ACKNOWLEDGMENTS

First and foremost I thank God for giving me the strength, stamina, ability and knowledge to complete this long and arduous path. Without the belief that my work was to be of some durable value I never could have completed it.

I owe much to my committee. Ed Riseman, whose pen I have come to dread, gave me invaluable advice and criticism not only in this dissertation but for my professional life in general. Al Hanson, whose availability was much appreciated, helped in countless ways. Michael Arbib introduced me to perceptual robotics in general and mobile robotics in particular. His stimulating discussions on schemas paved the way for many of the ideas in this dissertation. Victor Lesser pushed me for answers to hard questions. Hopefully they have been answered to his satisfaction. Steve Levitan provided the perspective that was essential for me to successfully complete the work, pointing out pitfalls that other unwary students may have been trapped in.

An array of institutions and industries provided assistance in many ways. Dr. Hans Moravec of Carnegie-Mellon University provided code that was of value for use in AuRA's short-term memory. Drs. Chuck Thorpe and Martial Hebert enabled us to use the CMU NAVLAB for a motion sequence. Michael Swain of the University of Rochester supplied code for use in camera calibration. The support and assistance from Denning Mobile Robotics (special thanks to Mark Kadonoff) was superb.

A large debt of gratitude is owed to the members of VISIONS and the Laboratory for Perceptual Robotics. Special thanks to Randy Ellis, Bruce Draper, Les Kitchen, Phil Kahn, Igor Pavlin, and Mark Snyder for sharing their knowledge, expertise and code with me. Bob Heller saved me countless hours of programming thanks to his incredibly deep knowledge of the VAXen and VISIONS systems. Bennie Murah helped me to retain my sanity through this ordeal by getting me out of the lab on occasion. Our technicians, Robyn Bari, Chris Helgesen, and Jim Drewett all contributed meaningfully to this research. Thanks also to Val Cohen, Janet Turnbull, and Laurie Waskiewicz for maintaining the day-to-day operations of the VISIONS group in a manner which freed me from concern for administrative details.

The secretarial staff of the COINS department also deserves a place in these acknowledgments. Rose Korowski answered my seemingly endless questions regarding duplicating, slides, etc. Renee Stephens interfaced with the graduate school providing the

reassurance that I had not neglected any requirements for my doctorate. Thanks to you all.

Finally, I acknowledge those people to whom I owe the most, my family. My Dad whose financial assistance made this all possible and whose confidence that I could complete the work drove me on. My late mother whose memory inspired me to strive for my best. I hope I have proven to be a worthy son. My children, Matthew, Rebekah, Sarah, and Hannah, who have unquestionably made the greatest sacrifice during this time, thanks for your patience. Hopefully now I will have time to catch up on being your Dad. To Michaelle, my beloved wife, who spent countless hours taking care of the children in my absence, proofing my papers, cheering me up when I was down and keeping me on the right track, my gratitude is beyond words. This research is as much yours as mine for without you it would not have been possible.

ABSTRACT

TOWARDS COSMOPOLITAN ROBOTS:
INTELLIGENT NAVIGATION IN EXTENDED MAN-MADE ENVIRONMENTS
SEPTEMBER 1987

RONALD CRAIG ARKIN

B.S., University of Michigan

M.S., Stevens Institute of Technology

Ph.D., University of Massachusetts

Directed by: Professor Edward M. Riseman

In the past, mobile robots have been constrained to operate in either an indoor or an outdoor environment, not both. Special purpose representations and ad hoc sensor techniques geared towards tasks of narrow focus have dominated these efforts. It is the purpose of this dissertation to lead towards the development of a more cosmopolitan robot; one whose domain of interaction is not as restricted as these previous attempts.

The Autonomous Robot Architecture (AuRA) has been developed to meet these challenges. A "meadow" map, used for global path planning and containing embedded *a priori* knowledge to guide sensor expectations, serves as the robot's long term memory. A layered short term memory based on instantiated meadows represents the currently perceived world. A hierarchical path planner produces a global path free of collisions with all modeled obstacles.

Schema theory is extended to include the mobile robot domain and serves as the principal theoretical framework. The schema-based path execution system handles unexpected and dynamic obstacles not present in the robot's world model. This motor schema-based navigation system produces reactive/reflexive behavior in direct response to sensor events. In addition, new techniques in the treatment of robot uncertainty which expedite sensory processing are presented. These include the use of a spatial error map with associated error growth and reduction techniques.

Several computer vision sensor strategies have been developed for use within AuRA. These include a fast line-finding algorithm, a fast region segmentation algorithm, and a depth-from-motion algorithm. Experiments using our mobile vehicle HARV demonstrate the use of these vision algorithms for navigational purposes. Schema-based navigation using ultrasonic sensing is also demonstrated experimentally.

TABLE OF CONTENTS

List of Figures

List of Tables

CHAPTER I

INTRODUCTION

Mobile robots, in order to be successful in a world as unconstrained as a human's, must be capable of responding intelligently to changes in their environment. Safe and successful goal-oriented navigation can only occur if the robot is able to emulate intelligent behavior. It should be recognized that in the real world:

1. Things are not always as they appear.
2. The world changes over time.
3. The spatial limitations of sensing when combined with mobility lead to new perceptions as previously unknown parts of the world are encountered.

Intelligent behavior can be defined as the ability to respond to changes in the perceived world in an advantageous manner. It should be made clear that a sharp distinction exists between reality and the perceived reality of the senses. A view of the world is just that, a view, and is not the world itself. If this fact is ignored and all perceptions are viewed as valid unambiguous interpretations of reality, any cybernetic system (animal or machine) is doomed to a short-life span. Consequently, it is of fundamental importance for such a system to be able to extract from the wealth of data provided by its senses a coherent interpretation that is subject to later revision and possible revocation based on freshly acquired information. Simply put, a robot must be able to alter its beliefs.

An AI system with the ability of handling this difficulty is usually vulnerable to the frame problem [46] i.e. the system is not fully aware of all the consequences of any action that is undertaken. Fortunately, action-oriented sensing can serve as a means to cope with this ill.

In order to design a mobile robot system with the ability to behave intelligently, the real-world characteristics enumerated above must be addressed. Handling errorful perception requires a mobile robot system capable of frequent sensor sampling and uncertainty management. Multiple sensors utilizing appropriate sensor fusion techniques can also ameliorate problems arising from the differences between the real and perceived worlds. Spatio-temporal changes in the real world require frequent updating of the robot's internal world model. It is unsafe to assume, except in highly structured workplaces, that a static environment exists. World modeling and navigation should additionally extend to regions where the robot has never been before or which it has not encountered recently. Ideally, learning can also be used to adapt to slowly varying changes in the world and to add new locales to the robot's memory.

Most mobile robot systems built to date have had a narrow focus of interaction with their environment. There are road-following vehicles, hall-following robots, guide-dog robots, etc. Little effort has been placed, however, on the development of a more general purpose robot capable of functioning well in both indoor and outdoor environments. If robots are expected to be other than special-purpose (although programmable) machines, a bridge must be built to allow the transition from constrained environments to more open ones. The key to accomplishing this transition is the construction of an architecture and accompanying representations that are less restrictive than those previously employed.

The thrust of this dissertation is the development of a mobile robot navigation system (AuRA - Autonomous Robot Architecture) that can operate in environments to which civilized man is accustomed. This system is developed and experimentally tested in what are termed extended man-made environments. These environments include the interiors of buildings, streets, sidewalks, city and campus settings. It is not intended to deal with navigation in jungles, swamps, forests and other unstructured environments, in which indeed a human - without a compass, clear skies, or additional information - can become hopelessly lost. The principal reason for the choice of these environments for the mobile robot's domain is the wealth of visual cues they afford the vision system. The assumption is made that man-made objects are more readily discernible via machine vision than natural ones, largely due to a significant population of regularly shaped edges or characteristic colors of such objects. These cues can be used by the robot for localization, guiding it to greater confidence in its position relative to the world, as well

as goal recognition and obstacle avoidance.

§1. Navigation

Navigation can be simply defined as moving from one place to another in an effective manner. One may navigate amongst that which is known ahead of time, or in unknown territory. In a perfectly modeled world, a path can be computed in advance that is completely acceptable. The robot's task becomes simply to maintain its bearings relative to the precomputed global path so that it does not deviate from this predetermined route. At the other extreme, where nothing is modeled ahead of time, the robot reacts to its environment, seeking an unknown goal amidst unforeseen barriers. The first case, moving in a known world, will be referred to as map-navigation, and the second case, reacting to unknown events, will be referred to as piloting.

Most real world situations involve both navigational forms. When given a task, any *a priori* knowledge of the world (if available) is mustered prior to the initiation of movement. In a perfectly modeled world, simple path adherence is all that is required. In a less than completely known world, (i.e. the real world), the robot starts moving along a predetermined path. If no unusual events occur, the pilot maintains the map-navigator's path. If an unexpected event occurs, the pilot initiates dynamic sensor-based replanning. As long as the deviations are minor, the pilot handles the replanning. If the path deviations are too severe, the pilot informs the map-navigator, who computes an alternate path.

An important distinction is that map-navigation is model-driven and sensor-independent whereas piloting is largely sensor-data driven. It is entirely possible for the pilot to inform the map-navigator of newly sensed information so the world model can be updated to incorporate this new data. This is in essence a learning mechanism. It is important to note however that not everything the pilot detects should be entered into the map-navigator's model. Moving obstacles (cars, people, etc.) may be present only for a very short time and should not be added to the navigator's map. This implies that semantic interpretation of the sensed data is essential for navigational updating of the world map.

Another important distinction between map-navigation and piloting lies in scope.

Map-navigation is more global in nature, ignoring the small detail to arrive at a satisfactory global solution (based on available knowledge). The short-sighted pilot, on the other hand, is concerned with the immediate environs and deals with sensed but unmodeled obstacles that are not represented in the navigator's map. Meystel describes the role of scope in navigational planning in [83].

Sidestepping the issue of learning for the moment, the maintenance of two distinct representations for the different levels of navigation facilitates planning: long-term memory (LTM) for the static world representation used by the map-navigator, and short-term memory (STM) which is used by the pilot for building up a perceived model of the world. Localization (determining the robot's position relative to the global LTM map) becomes a matter of correlating STM with LTM, and learning involves moving relevant features from STM to LTM. The pilot also draws on LTM but in a more limited way, LTM providing specific cues for the pilot to look for (e.g. landmarks) which can be used for localization. The pilot monitors only those landmarks in its vicinity and thus can more effectively use its available computational resources.

For the purposes of piloting in AuRA there exist two strategies: a low-level reactive or reflexive approach using schema-based control structures which does not draw directly on memory (either STM or LTM) but instead uses sensor data as it is received, and another method using both the local context of LTM and the accumulated sensor data in STM when the reflexive approach fails. Unmodeled and changing world conditions are treated through relevant schema and STM representations. The distinctions between these two methods will be discussed in Chapters 3 and 5.

A partial *a priori* model (LTM) of the domain of interaction is provided for the map-navigator's use. It includes static objects, (lampposts, walls, buildings, etc.), but omits dynamic ones (people, cars, chairs, etc.). The question of the robot learning and adapting its representations to meet the demands of a changing world is considered in the theoretical development of the architecture presented in Chapter 3.

§2. Characteristics of mobile robotics

Mobile robotics in many respects is decidedly different from conventional robotics. It is worth describing some of the characteristics that distinguish it from the more con-

ventional robot arms and manipulators. Some of the material in this section is loosely adapted from Thorpe [126] and Andresen et al [3]. These characteristics include:

- Inherent inaccuracy
- Limited degrees of freedom
- Cumulative error
- Incomplete model
- Environmental uncertainty
- Non-repetitive paths
- On-line, continuous path planning

Inherent inaccuracy

The sensors relied upon by mobile robots can easily give rise to imprecise and inaccurate data. Even if the devices themselves are highly accurate, (e.g. shaft encoders), the correspondence of the changes in the sensors to the changes in the robot's environment may be poor (e.g. due to wheel slippage). Feedback in the traditional sense of control theory is not immediately applicable and can only be used as a guide to establish expectations for higher level processing.

Limited degrees of freedom

The number of degrees of freedom for the mobile robot are significantly less than those of a robotic manipulator. Assuming no translational motion is allowed in the vertical (up and down) direction (generally a necessity due to available locomotion systems - this would change if applied to legged, flying and submarine robots), there are 2 DOFs of translation and one DOF of rotation. This is half of the 6 DOFs commonly found in a robot arm and wrist, a decided decrease in complexity.

Cumulative error

Errors if left uncorrected will tend to increase. This is typical of any type of dead-reckoning system. A path cannot be computed and the robot sent off to execute it

without frequently verifying and correcting the robot's internal model of its position. Merely avoiding obstacles along the way is insufficient to guarantee that a robot will reach its goal or even recognize when it reaches it. Consequently, information must be maintained in a representation that enables this type of updating to be performed.

Incomplete model

Any model by definition is incomplete, otherwise it would not be a model. Internal world representations for mobile robots are perhaps more incomplete than most, due to the larger and more unstructured world in which it operates when compared to industrial robots. Space-versus-time computational tradeoffs must be made in order to meet the real-time constraints of path planning, replanning, and obstacle avoidance. Excess representational baggage is a luxury that generally cannot be afforded. It is difficult to see how any representation can be maintained, updated and accessed by algorithms that must process the data in *real-time* with *existing* hardware and yet is complete enough for accurate positioning of the robot, semantic interpretation of high level commands and objective statements, supporting multi-modal sensors, coping with uncertainty, handling goal recognition and choosing alternate path-planning strategies dependent upon external factors.

Representational incompleteness can also be encountered when the robot is required to traverse areas in which it has never been before. The representation may have to be built dynamically from only partially correct and possibly contradictory sensor data.

Environmental uncertainty

Things are not always where they are expected to be, even when they are modeled. Not only is the robot's position uncertain, but the location of objects will also have to be treated with skepticism. In addition, objects may have moved since their last observation or even be in motion relative to the world or the robot. This is in marked contrast to the robot manipulator's highly structured workplace.

Non-repetitive paths

The path an autonomous mobile robot executes is unlikely to be the same twice. Although the general route may be the same, changing conditions combined with positional

errors usually will require the actual path taken to differ from the high level specifications each time it is traversed. If this were not so, as in some manufacturing situations, a stripe or wire following automatic guided vehicle (AGV) would be the robot of choice instead of an autonomous vehicle.

On-line, continuous, path planning

The robot must not close its "eyes" for long while moving. Constant monitoring for collision avoidance is essential. To obtain enhanced performance, path planning should be maintained during robot motion, just in case an unexpected event arises that would necessitate a path change. These might include such things as an unanticipated barrier (necessitating a detour) or the absence of a modeled obstacle (opening up a better path). This dynamic replanning must be conducted in real-time.

§3. Overview of the dissertation

The ultimate goal of this dissertation is to provide a broad, relatively unrestricted approach to the problem of mobile robot navigation. To accomplish this, several specific issues will be addressed. These will include:

1. The development of an architecture to support intelligent navigation of a multisensory (predominantly visual) robot in a "civilized world".

   Approach: The Autonomous robot architecture (AuRA) is forwarded as the structure to accomplish this goal. The "civilized world" includes both indoor and restricted outdoor travel. The outdoor case will assume a significant population of man-made objects (roads, paths, sidewalks, buildings, etc.) but also will allow for substantial natural surroundings (grass, trees, sky, etc.). Extensibility and generalizability are considered fundamental design goals.

2. Effective fusion of the visual data with other sensory input (e.g. shaft encoders and ultrasonics). The development of techniques that are appropriate for resolving conflict between contradictory sensory data while enhancing cooperative input.

   Approach: The use of motor schemas and associated perceptual schemas as control mechanisms to funnel relevant sensory data to the appropriate motor task.

3. Representation of spatial uncertainty and its use to guide expectations for perception. The use of sensing to restrict the limits of uncertainty through feedback.

   Approach: The use of specific modules in the AuRA architecture dedicated to the management of uncertainty, including the spatial uncertainty map and its manager, and the Expecter used to provide expectations to the perception subsystem.

4. Selection of appropriate vision algorithms for specific tasks.

   Approach: Associating specific strategies through perceptual schemas to provide action-oriented perception. These include:

   - A depth-from-motion algorithm for obstacle avoidance.
   - A line extraction algorithm for path following.
   - A region segmentation for path following and localization.
   - Scene interpretation and interest operators for landmark recognition.

5. The use of knowledge representations that can effectively deal with navigation in "civilized" environments and are general enough to be used both indoors and outdoors.

   Approach: A multi-level representation managed by a cartographic process that maintains both an *a priori* model of the environment in addition to a dynamic model of newly encountered obstacles and features.

Experimental testing of the resulting robot system is in two arenas: navigation within the Graduate Research Center at the University of Massachusetts - Amherst (UMASS) and in the outdoor area surrounding the same building. The results are presented in Chapter 8.

The dissertation is structured as follows:

- Chapter 1 is this introduction.
- Chapter 2 describes relevant prior work in the field of mobile robot navigation, including an analysis of the types of representations and control strategies used. An overview of work in the use of vision as a sensor for robot navigation is also presented.

- Chapter 3 presents the structure of the Autonomous Robot Architecture (AuRA) that is used as the framework for the experimental work of this dissertation. Motivation for its structure is also included.

- Chapter 4 describes in detail the role of the navigator and long-term memory representations used to develop a path for the robot based on *a priori* knowledge. The inclusion of both indoor and outdoor terrain types are important extensions to previous work. The roles of the mission planner, pilot and their accompanying representations are also presented.

- Chapter 5 puts forth the motivation for motor schema based mobile robot piloting. Action-oriented perception is a fundamental tenet of this approach. The use of schemas in AuRA is described. The inter-relationship between the motor schema manager and pilot is also discussed.

- Chapter 6 describes the specific sensor algorithms, both visual and ultrasonic, that are used to provide environmental data to AuRA.

- Uncertainty management is discussed in chapter 7. An exposition of the role of the spatial uncertainty map and its manager and the use of represented uncertainty to guide perceptual expectations is provided.

- Chapter 8 presents the actual experiments performed to validate the concepts presented in this dissertation. These include both indoor and outdoor runs with differing levels of *a priori* knowledge available.

- Chapter 9 concludes the dissertation with a summary of accomplishments as well as a discussion of future work.

CHAPTER II

A REVIEW OF MOBILE ROBOT RESEARCH IN NAVIGATIONAL PATH PLANNING, MOTOR CONTROL AND VISION

Considerable work has been undertaken, principally in the last decade, to provide a robot with the ability to navigate autonomously. This chapter will first survey the different approaches researchers have used for equipping their vehicles with navigational planning ability, concentrating on the control and representational strategies. The final part of the chapter will review various attempts to use machine vision as a sensor to effectively execute a plan developed within a navigational planning system. Spatial reasoning systems as in [82], although important for the semantic treatment of path planning, will not be dealt with in this review.

Path planning is a well-studied problem having an extensive history. It has been analyzed in the context of operations research, engineering, computer science, as well as other fields. The mathematics has been studied intensively and many algorithms have been developed, typically using graph representations, to determine a path in a completely known environment.

Navigation for a mobile robot, however, is a non-traditional path planning problem. The underlying reason is that a mobile robot operates at best in only a partially known or understood environment. The causes are manifold: uncertainty and conflicting information arising from inaccurate sensor data, dynamically changing environments, and incomplete models are some examples.

The goals of a navigational planning system for a mobile robot are:

- to maintain a model of the world surrounding it.
- to update that model using sensor data.
- to determine a path to a specified goal based on some criteria for "best" path.

Monolithic control

sensor dependent

Hierarchical control

single level representation

sensor dependent

sensor independent

multi-level representation

sensor dependent

sensor independent

Distributed control

simple blackboard representation

multi-level blackboard representation

Spatial reasoning Systems (independent of mobile robots)

Figure 1: Mobile Robot System Taxonomy

- to modify that path as necessary, during execution, based on new or conflicting sensor data.

A plan cannot be generated by a planner and blindly executed by the robot without additional sensor feedback. This would be courting disaster. Consequently, many approaches have been taken to try and cope with the uncertainty and non-monotonicity inherent in the problem.

A survey of the work done in the field of path planning for mobile robots is presented in the section following. A taxonomy of systems based primarily on control is given in Figure 1. This taxonomy is somewhat artificial but hopefully can shed light on the differences between the systems developed. It is not intended to be interpreted as an absolute classification of any system, as indeed there may be considerable overlap of categories. Some of the systems discussed later in this chapter may possibly be considered misclassified by some readers, but that should not affect the overall discussion.

A monolithic control structure, although occasionally implemented, is generally inappropriate for mobile robot path planning and navigation. Instead, a hierarchical planning system incorporating model and data representations that provide for easy exchange of

information as well as adaptation to changing conditions is unquestionably more suitable. Many variants of this strategy have been implemented. In a hierarchical control system, it may be desirable to use multiple representations instead of one single global all encompassing representation. Finally, a concurrent, distributed, cooperative system may be considered the ultimate in control schemes, typically utilizing a blackboard architecture for the passage of information. This strategy has several distinct advantages, not least of which is its ease of adaptation to parallel processing, so crucial to meeting the real-time constraints imposed by the mobile robotics domain. For distributed systems, multi-level blackboards may be preferred.

Some of the mobile robot systems developed have allowed their representations to be affected by their choice of sensors rather than taking a truly sensor-independent approach. A sensor-independent approach to representation and control is theoretically preferable, but sensor specific models are quite common due to the relative ease of implementation and other expediency issues.

Artificial intelligence techniques are far more evident in mobile robot path planning than are the traditional operations research algorithms. Heuristic search methods are commonplace. Considerable work has been dedicated to the treatment of uncertainty. Spatial reasoning and non-monotonic logic are also important although currently underdeveloped for this application.

§1. Survey of control strategies

The tasks facing a mobile robot are many: path planning, obstacle avoidance, object detection, landmark recognition, position evaluation and world model building, to name a few of the goals and required capabilities. The way in which control is applied and the exchange of information between the different task modules constitute principal differences in the mobile robot systems developed to date. Three basic approaches can be seen:

- monolithic
- hierarchical
- distributed

Each will be described using existing systems where applicable as examples.

§1.1 *Monolithic Control Systems*

Monolithic control is a fairly simplistic approach to a complex problem. No feedback is provided after the initial path generation decision and all planning and navigation is conducted at essentially a single representational level. Virtually all navigational knowledge is procedural. Consequently, monolithic systems would be used for quick-and-dirty type systems, typically in a commercial environment, that are constrained by limited computing power. In addition, several research systems that have been built to demonstrate sensory capabilities are monolithic, but no claims are made for their general use. In these cases, the representations used are strongly sensor dependent and thus hamper the possibility of generalization. Although it is theoretically possible to build a system that uses both monolithic control and sensor independent representations, none has been encountered to date.

One of these systems [130] uses a teaching pendant style approach to navigation, so common in commercial robotic arms. The robot is rolled to a particular location on the desired path where sensory data is recorded (for recognition purposes) and then it is moved to the next point. This process is repeated until the path is complete. The global path is never modified, and changing conditions are handled by using local obstacle avoidance strategies. In other words, once the global plan is made it is not modified.

The principal advantage of this control regime lies in its ability to respond rapidly, certainly crucial for meeting the real-time needs of a mobile robot. Although monolithic control is a short-term limited solution, this approach would best be avoided even by commercial developers so that prospective users (and buyers) of mobile robotic systems will not be discouraged by its shortcomings to the point of abandoning the technology.

§1.2 *Hierarchical Control Systems*

The preponderance of the control systems for mobile robots documented in the literature utilize hierarchical control. That is not to say they are similar, quite the contrary. What they have in common is a structured and clearly identifiable subdivision of functionality. This functionality is relegated to distinct program modules which communicate with each other in a predictable and predetermined manner. Numerous examples illustrate this technique [90,100,108]. Two are described below.

A system developed by Hughes Artificial Intelligence [63] exploits hierarchical planning control. At the highest level is the MISSION PLANNER which establishes objectives, choice of optimality, and is responsible for invoking spatial and commonsense reasoning. The LONG RANGE PLANNER is at the intermediate level and charged with developing a global path subject to the constraints of the mission planner. The LOCAL PATH PLANNER, the lowest level, is used for obstacle avoidance and navigation around obstacles not modeled at the higher levels. A clear segmentation of control is evident. Replanning can be invoked due to the unattainability of goals at lower levels (e.g. unanticipated barriers).

One highly developed architecture from the University of Florida, implemented only in a simulation, is described in [66]. The modules for navigation are the planner, the navigator, the pilot, and low-level controller. The first three correspond in function with the three levels described in the Hughes system above. The low-level controller is responsible for issuing vehicle commands and status monitoring. In addition, a cartographer module is present, whose responsibility is to maintain the world model. This map-builder is accessible from all levels of the planning hierarchy. This planning decomposition, typical of many mobile robot systems, is reflected in AuRA as well.

§1.3 *Distributed Control Systems*

Distributed control operates in an asynchronous manner. Global data structures such as blackboards are the keys to coordination and cooperation between the functional modules. These systems lend themselves extremely well to multiprocessing.

HILARE [51,52] is one of the more advanced and highly respected mobile robot systems, developed at LAAS in France. Its control structure falls in line with distributed systems. Specialized decision modules (SDMs) utilize a global database ("bulletin-board")

for communication purposes. These SDMs are used for planning, navigation, scene analysis and the like. The blackboard is bi-level: an announcement database and an information database. Communication between the SDMs occurs through this structure.

Ambitious work at Carnegie-Mellon University [120,127] uses knowledge sources to provide and interpret information on a multi-level local map. The local map-builder functions as a scheduler for the knowledge sources. Cognition modules are present, requesting information from the local map-builder for purposes of path planning, landmark recognition, global map updating and vehicle status monitoring. A major advantage of this system lies in its ability to easily adopt new knowledge sources, cognition modules and sensors into its structure. Clearly defined interface specifications will aid in the development of code at sites other than CMU and speed up the overall development of this system. A brief discussion of the blackboard representation used for the local map appears in Section 2.2.

Brooks [24], in a trend-setting paper, proposes a horizontally decomposed planning system. This allows for the concurrent operation of multiple behaviors in a layered manner. Although a pseudo-hierarchy exists, with certain of the behaviors having a higher priority than others, the concurrent operation of the system classifies it as a distributed control system.

Payton [107] and Kadonoff et al [60] also propose multiple asynchronous control systems that can produce multiple behaviors. Both of these systems as well as Brooks' are compared in Chapter 5 to the distributed control used in AuRA's motor schema manager. It should be noted that AuRA encompasses both hierarchical and distributed control: hierarchical for the development of the plan through its mission planner, navigator and pilot; then distributed for the actual path execution under the control of the motor schema manager.

§1.4 *Assessment of Control Systems*

For any but the simplest mobile robotic systems, a distributed or hierarchical system should be chosen. Only if a particular theory regarding sensor usage in navigation is being tested should a monolithic control structure be employed. Monolithic control is too inflexible to handle the serious issues in navigation. In the non-monolithic approaches, the scope of planning can vary. Real-time responses can be facilitated using small-scale

planning on a local, sensor-driven level, whereas global plans can be developed (based on *a priori* knowledge) beyond the range of environmental sensing. Indeed the problem of navigation itself can be viewed as being solved by humans in either a hierarchical or distributed fashion.

The main question is: hierarchical, distributed, or some combination thereof? The advantages of each lies in the intended use of the vehicle test bed. Hierarchical control systems, in our opinion, are easier to develop, debug, and implement. For a production or commercial system, where all code is being developed on-site, the hierarchical approach enables a system to be built more rapidly, while still maintaining high functionality. A distributed system is in some ways more difficult to develop. Following the execution trace during asynchronous operation can be quite problematic. Although techniques are being developed for debugging distributed systems [16,35,75,122], much work remains to be done.

One major advantage the distributed approach affords is in incremental development of software. If it is properly designed, individual components (knowledge sources, SDMs, expert modules, motor schemas, etc.) can be plugged into the system with a minimum of disturbance. This leads to greater longevity for properly developed distributed systems. Incorporating new sensors is more readily accomplished. Additionally, by clearly defining the interface specifications, code can be developed by different groups, speeding up the overall software development process. The sharing of information is an additional major asset. Real-time processing demands are more likely to be satisfied, in part due to the greater ease in which a distributed system can be transported to parallel processors (in theory). The principal difficulty, however, is the development of a solid framework in which the individual components can execute and communicate effectively.

Essentially, if a system is to be developed rapidly with reasonably full capability and is not expected to undergo significant growth, but rather will be replaced by a new release, a hierarchical control system is the appropriate choice. If on the other hand, full functionality, high potential for growth, better high level reasoning support, and the ability to test out new ideas without a major redesign effort are more important than initial developmental difficulties, a distributed system is to be preferred.

§2. Survey of Representations

This section contains a description of the characteristics of a good representation for the mobile robotics domain, a survey of some of the techniques used to meet those objectives, and an assessment of their merits. This survey is not intended to be exhaustive but rather representative of some of the approaches used in world modeling for mobile robots.

§2.1 *Characteristics of a Good Representation*

To be able to evaluate different representational techniques, it becomes necessary to specify what properties are either essential or desirable for such a representation. Some of the characteristics described below are pertinent to artificial intelligence representations in general, while some are peculiar to the task at hand.

The principal characteristics of a good representation for the mobile robot domain include:

- efficiency
- representation of uncertainty
- multiple frames of reference
- sensor independence
- robot vehicle independence
- support for semantic information
- facilitation of parallel processing
- support of localization

Efficiency

In a real-time domain such as navigation in robotics, it is necessary to ensure that the algorithms used to process the data contained in a representation are efficient. If certain information is not explicitly contained in the representation, but must rather be computed at run time, real-time deadlines may be exceeded. The time-space tradeoff should

generally be resolved in favor of time with subsequent cost of memory. Nonetheless, integrity must be maintained to be certain that the stored data is not self-contradicting. For example, if an environmental feature's position needs to be updated, any other stored data that was determined based on the prior location of the feature must be updated as well. Obviously, if this will result in a large amount of change to the underlying data structures, it would be best to compute those quantities dynamically only when needed instead of storing them. The role and extent of uncertainty must also be considered in making the decision as to what should be explicitly represented.

Representation of uncertainty

Sensors do not provide exact information about the world surrounding them. Not only is that information inexact, it can be contradictory or erroneous. Consequently, any representation chosen to express the robot's world must take into account that things are not always as they appear to be, and that some data are more reliable or precise than others.

Multiple frames of reference

The robot's world is many faceted; it consists of objects relative to the robot, objects relative to each other, and objects relative to some absolute frame of reference. Uncertainty may exist regarding the absolute position of an obstacle within the world, but its position relative to the robot may be well established. Clearly, the object's relation to itself, i.e. its height, width, depth, and other features, does not depend on the absolute location of the object. By utilizing multiple frames of reference, these relationships can be expressed more readily.

Sensor independence

To be able to expand sensor capabilities, the most general case should be considered. The representation should encode sensor data in a form that is independent of the sensors themselves. Not all sensors provide the same type of data and each varies significantly in accuracy. Even when only one sensor device is used (e.g. video camera) there exist many different ways of processing the data provided, some more accurate, some more robust, some more reliable than others. In order to incorporate as much information as possible

into the representation, interfaces must be defined that are as sensor independent as possible. It should not matter that depth data came from ultrasound or a laser rangefinder or optic flow, or even from more than one visual depth algorithm executing concurrently. Only what the data are and an estimate of their reliability and accuracy is important. Granted, certain sensor data could best be combined directly in the presence of sensor models and algorithms for fusing sensor data. Nonetheless, the ultimate representations used specifically for path planning and navigation are more general if they are removed from sensor-dependent representations.

Robot vehicle independence

The use of a "virtual" vehicle, as described in [127], is the preferred modality. If a new vehicle is brought onboard, there would be no need to redesign the representation.

Support for semantic information

Symbolic reasoning will be essential if the robot is to do more than simply get from point A to point B. If spatial reasoning [as in 82] is to be implemented at the mission planner level, some means for storing symbolic knowledge about the world must be provided.

Facilitation of parallel processing

Realistically, to obtain real-time response to sensory data in an intelligent fashion to at best a partially known environment, parallelism must be exploited. This will enable the robot to move continuously through the world instead of in "lurch" mode.

Support of localization

It is not sufficient for the robot to only know a path to get from the start to the goal. It must also maintain information to determine where it currently is located relative to that path (i.e. localize itself). This necessitates the representation of more information than would be necessary if there was no drift in the robot's position from the path specified by the path planner.

§2.2 *Representation Techniques*

Several different approaches have been used in representing the information for path planning. These include:

- Pure free-space methods
- Vertex graphs
- Hybrid free-space vertex graph methods
- Potential Fields
- Regular grid
- Quadtree
- Automaton
- Multi-level

Each has an associated history and advantages. They will be discussed in turn.

Pure free-space methods

Two principal techniques fall under free-space methods: Voronoi diagrams and generalized cylinders. What is represented in both these approaches is the space between the obstacles, rather than the obstacles themselves.

Voronoi diagrams play an important role in computer science and mathematics, in particular when dealing with closest point problems [116]. A free space representation resulting from a Voronoi diagram represents space as a series of connected straight-line segments that typically split the distance between the closest pair of line segments of surrounding obstacles (including bounding walls). In appearance, the result can resemble a medial axis transform. More formally, a Voronoi diagram is produced by the generation of a set of polygonal regions, each representing an enclosed area in which all contained points are closer to one particular point in a given point set than to any other point in the set. This set of polygons partitions the plane into a convex net. The resulting diagram or its straight-line dual can be used to compute the safest path for navigation of a mobile vehicle.

Several problems are evident with this approach, including the fact that information regarding the obstacles themselves is discarded and is no longer available for high-level

reasoning processes. In addition, Voronoi diagrams cannot be readily used in changing environments or for moving objects. Every time the diagram is updated to navigate around unexpected obstacles, information is lost regarding the original clutter-free path. Maintaining the assumption that the paths traversed by the robot will seldom be identical, the utility of the Voronoi approach wanes.

One variation of the Voronoi diagram that is well adapted for robot localization using sonar is described by Miller [84]. He subdivides free space into regions that are characterized by their ability to provide localization information. Generally, the series of regions which yields the path through which the robot can maintain its position with the highest degree of certainty is chosen. A Voronoi diagram is produced of the regions within this path to yield a safe and certain route for robot travel using sonar navigation. The approach is limited in its applicability.

Another method using Voronoi diagrams coupled with a spatial vertex graph is proposed for the robot HERMIES [59]. Dynamic map making capabilities are provided using this representation. The Voronoi diagram is used to partition space into equivalence classes, guiding selection of the appropriate spatial vertex from which to initiate path planning. Two major assumptions are made, however, which may prove invalid in realistic situations: first, that the obstacle locations are unchanging and the world is static, and second, that sensor information is precise. Voronoi diagrams are poorly adapted in general to handling a dynamic world, whether or not the world is modeled *a priori*.

Brooks has developed a different free space approach that has extensions to three dimensions for classical pick and place operations for robot arms [26]. In a form not alien to computer vision specialists, he subdivides space (2D in the case of mobile robotics) into "generalized cones" or "freeways". (See [27] for the algorithm). These freeways, represented as spines with information regarding their clearance to the left and right, are produced by sweeping a 2 dimensional cross section through space according to a sweeping rule. An unusual characteristic of this representation is the allowed overlap of the freeways that are produced, contrary to the disjoint convex regions found with other methods. This representation, however, throws away too much information too soon, suffering from the same problems as the Voronoi approach. Recognizing these limitations, Brooks and co-workers have developed a hybrid extension [71] using both

passages (freeways) and convex regions of free space.

Vertex graphs

This technique owes its origins to path planning for robotic manipulators. Most forms have their basis in a two-dimensional version of Lozano-Perez's configuration space approach [76]. Here, the vertices of an obstacle are modeled by a polygon. The vertices are then grown by a distance equal to the radius of a circle enclosing the robot plus some margin of safety. Although the configuration space approach does not require making the assumption that the robot is circular, most working mobile robot systems do, thereby simplifying the computational problem dramatically.

A conspicuous feature of vertex graphs is their lack of explicit representation of the space between the obstacles. The advantage of this technique lies in the fact that the robot can now be treated as a point and occupies no space for path planning purposes. Extended versions of this algorithm have been developed for robots that are not circular, for path planning in 3 dimensions, and for conventional robot arms. Computational complexity for path planning can be of major concern for higher dimensions. See [85] for a discussion of the complexity issues regarding path planning algorithms.

Moravec, in his thesis [93], used an unusual representation that is not unlike vertex graphs. Obstacles were modeled as circles rather than polygons, and paths were constructed as a series of tangents to the circular obstacles in so-called "circle space". Needless to say, planning for an optimal path is considerably more complex with this representation than with a pure vertex graph approach. This lead Moravec to develop a sub-optimal algorithm that approximates the shortest path in circle space and produces the optimal path 95% of the time. This faster algorithm was used for a practical implementation in the robot Rover.

Hybrid free space and vertex graph

Representation schemes combining concepts from both vertex graph and free space representations are common in mobile robotics. Several examples are cited below.

A hybrid approach used by Chatila and Giralt in HILARE [34,51], Crowley with Neptune [36], and others (including our own work in AuRA), represents both the free space and the obstacles by vertex graphs. The space between obstacles is subdivided into

convex regions typically termed "meadows" or "places". It is a characteristic of a convex region that any point can be reached from another point within that same region without a collision (for all modeled obstacles). This reduces path planning to simply determining a sequence of piecewise linear traversals of meadows. This can be readily deduced from a connectivity graph.

Giralt [51,52] and Chatila [33] in HILARE have a highly developed free space representation that is developed dynamically from sensor data. It does not grow obstacles *a lá* Lozano-Perez and consequently path planning is complicated as the robot cannot be treated as a point. Space is modeled at two levels; a topological level which maintains information regarding places and their connectivity, and a geometric level which assigns dimensions to the components of the topological graph. Multiple frames of reference are available. This representation allows the robot to develop a map in totally unknown areas using only sensory data (laser rangefinder and vision). Its use is currently limited to indoor environments and fixed obstacles. Another problem is that uncertainty is not directly represented.

Crowley [36], in one of the more lucid papers on planning for navigation, uses an approach similar to Giralt's, but the convex region is first grown (shrunken actually) in a configuration space style. The algorithm used maximizes the area of the largest convex region. A "network of places" is produced which are connected not by vertices but by "adits" (definition [138]: a nearly horizontal passage from the surface of a mine). Despite the unusual terminology, these adits allow navigation through free space in a manner similar to that of the generalized cones approach described above.

Monaghan [90] uses a hybrid approach dubbed complex configuration space. Its only distinguishing characteristic is the use of arbitrary polygons instead of solely convex ones. As a consequence, it bears a stronger resemblance to the vertex-graph model than the other representations of this section, and thus inherits the advantages and disadvantages of that technique.

Brooks [25] argues strongly against using any two-dimensional representation for expressing the robot's world. Additionally, the claim is made that uncertainty cannot be modeled accurately by using any representation based on an absolute coordinate system. Instead a "rubbery, stretchy" relational map is proposed, incorporating both freeways and convex regions, as in [71], and explicitly storing the uncertainty in the represen-

tation. An abstract graph which deliberately avoids the use of a global 2D coordinate system is the principal representation medium. Local coordinate systems and their associated transforms (with their error functions) are used to relate the modeled regions. His claims are a bit extreme for the case where *a priori* knowledge is available for the modeled world. When a dynamic world map is developed from multiple and disparate sensor readings, his arguments seem more plausible.

Potential fields

This representational scheme uses a gravitational analog, converting obstacles into peaks and clear paths into valleys. The robot's initial position is placed at a higher elevation than the goal and the algorithm treats the robot as a marble rolling towards a hole. Vectors representing the attraction of the goal, the avoidance of obstacles weighted by urgency, and the existing acceleration are combined to yield the movement of the robot. Although extension is possible to mobile robotics [64], most of the work to date deals with manipulator trajectories. Potential fields in AuRA have found a place in the lower levels of a multi-level representation for a mobile robot; providing information for the motor schema manager to cope with obstacles, seek goals, follow paths and other behaviors during short range navigation, while using alternate representations for higher level planning.

Krogh and Thorpe [69] also apply potential fields techniques to the problem of mobile robot obstacle avoidance in conjunction with Thorpe's path relaxation techniques [126] based on a regular grid.

A principal advantage of this technique lies in its ability to represent uncertainty by changing the slope of the obstacle peaks. High confidence levels produce very steep cliffs, while uncertain obstacles result in a slowly rising broad slope. Optimal solutions were not part of the formulation of this control scheme, although the results are said to be "in some sense efficient" [68 p. 6]. Other problems exist in the susceptibility of the system to local potential energy minima, requiring specialized extensions (typically involving subgoals) to avoid "box canyons" [68].

Regular Grid

Grid representations have the richest history in mobile robot research. SRI's SHAKEY, Berkeley's JASON, and JPL's ROVER all used versions of the grid representation for navigational purposes [47]. This technique represents space in a classical 2-dimensional cartesian grid. Current systems using this technique have been developed by Thorpe [126], Mitchell [88] and others. A major impetus for the use of this representation lies in the fact that the Defense Mapping Agency (DMA) uses a similar format for their maps. Defense contractors working on the Autonomous Land Vehicle (ALV) Project will need to use the DMA data as a primary source of topographical knowledge.

A navigation representation using a regular grid approach has been developed by researchers at Hughes Artificial Intelligence [88] (see also *multi-level representations* below). This grid represents a 2D model of the world, each "pixel" representing a space 12.5 m by 12.5 m. The design decision is based on the availability of DMA data in a similar, though not identical, format. Connectivity is maintained through 8 arcs to each of the pixel's nearest neighbors. This poses significant problems for path planning, as paths, if unchanged, can only occur at angles of 45 or 90 degrees from node to node. This could cause any path developed by a navigator to be excessively long and thus non-optimal (using distance as a metric). Digitization bias is the term applied to this specific problem. Considerable effort has been made into minimizing the bias [87]. Additionally, compensation must be built into the search algorithm for the fact that a step in the diagonal direction is 1.4 times as long as one in the vertical or horizontal direction.

Thorpe [89,126] also uses a regular grid approach with 4 or 8 neighbor connectivity. An approach to overcoming the digitization bias, called path relaxation, is discussed in Section 2.4.

Moravec and Elfes [92] utilize a "world occupancy map" for representation of sonar data. Cells in the map (typical resolution of 6 inches by 6 inches) are used to represent the likelihood that an obstacle is present at a particular location. This model is one of the few to explicitly incorporate uncertainty. A numeric value in the range (-1,1) is associated with the cell. Negative numbers represent unoccupied regions, positive numbers occupied regions. The greater the absolute value of the number, the more certainty is associated with the conclusion; zero denotes no information.

One of the more unusual approaches using the regular grid method is presented by

Parodi [104]. Although the representation used is similar to the Hughes model, its use in global path planning is markedly different. The global planner accepts many different cost criteria from the mission planner: energy consumption, potential hazard, travel time, etc. and their bounds. Instead of computing a conventional point to point path from start to goal using a graph search algorithm, dynamic programming techniques coupled with relaxation are used to provide a cost function map. This in turn is used to develop a path description list which is passed to the pilot for execution. The major drawback of this method for path planning, as would be expected, is prohibitively high computation cost (up to 20 minutes of VAX-780 CPU time) necessitating the use of specialized processors for this algorithm's practical implementation.

Quadtree

SHAKEY was one of the first systems to use a quadtree representation of space for mobile robot navigation [97]. Researchers at the University of Maryland [3] have extended its use as a representation for planning purposes. Basically, space is recursively decomposed into 2^i by 2^i areas, not unlike what is found in the regular grid approach. If larger blocks have similar occupancy (binary valued - 0's for free space, 1's for obstacles), the decomposition stops before reaching the lowest levels of resolution.

The representation is used to generate a path which consists of a sequence of blocks through free space. Only vertical and horizontal neighbors, not diagonals, are used as a safety factor to minimize the likelihood of clipping of obstacles during path traversal. This only amplifies the already existing problem of digitization bias that this technique shares with regular grids.

The advantage afforded by this representation lies in the reduced computational cost for path construction as compared to the regular grid (although the initial map building is more expensive). On the other hand, free space representations are more expensive to construct than quadtrees, but are cheaper for path planning purposes. The question of how environmental uncertainty is handled in quadtrees is not dealt with by the Maryland authors but it would probably fall in between these two representations (regular grid and free space) in complexity. The principal drawbacks are perceived to be the possibly coarse and certainly varying resolution for path construction and the loss of uncertainty information due to a simple binary encoding of occupancy. Finer encoding of the occu-

pancy values for uncertainty would cause this representation strategy to degenerate into the regular grid approach. Also, additional information must be maintained about the nature of the obstacles in another level of representation if it is to be used for semantic processing or other reasoning.

Automaton Representation

In the most unorthodox representation scheme encountered, Tachi and Komoriya [121], in a well-funded and seemingly successful venture with "guide-dog" robots (more in Section 3.2), develop an automaton representation map for navigation purposes. Landmarks constitute the states of the automaton, each state containing information regarding the type of the landmark (currently only intersections). Inputs to the states are directional commands (left, right, straight) and outputs include steering angles for the robot and distance to the next landmark (among other things). The MELDOG system for visual navigation has received considerable attention, and therefore, although the automaton map is somewhat unusual, it warrants consideration.

Multi-level Representations

Multi-level representations are most appropriate for systems that afford multiple levels of control, i.e. hierarchical or distributed. The main idea is to represent information in forms that are best suited for different types of processing: high-level structures that can be tagged semantically for mission planning, absolute coordinate systems for low-level navigation and obstacle avoidance, etc. The greatest danger lies in the potential for inconsistent information to be stored in the different representations, so deliberate steps must be taken to maintain the overall integrity of the system. The choice of which representation to use at what level is still a major issue.

A multi-level blackboard system is being developed at CMU for DARPA's ALV project [120,127]. Sensor-dependent data resides at the lowest level. Here, representations will be chosen that are strongly influenced by the sensors employed. A three-dimensional coordinate system will be used. An intermediate blackboard level will consist of hypotheses or "partially instantiated models", not unlike a schema strategy as in the UMASS VISIONS system [55,53,42], which have been refined from the sensor-dependent data or established as expectations from higher level models. The blackboard's top level contains objects

that have been declared to be identified ("fully-labelled"), expressing components of the ALV's world in 3D coordinates. Information in the local map (short-term memory) is written to the global map (long-term memory) by knowledge sources when appropriate.

In another system developed for potential ALV use by Hughes Artificial Intelligence Center [63] involving the hierarchical control scheme described in Section 1.2, multiple levels of representation are used. At the mission planner level, symbolic manipulation of the data represented is important. An object-oriented data structure termed the "symbolic pixel array" is used. The details of this representation were not disclosed in the literature encountered at the time of this writing. The navigator, charged with developing long range plans, utilizes a two-dimensional integer lattice (described in the *regular grid* discussion above). This has obviously been influenced by the availability of DMA map information in a similar form. At the pilot level, and specifically geared for use in obstacle avoidance, a polygonal representation (as in HILARE) is proposed. It is worth noting that the scale used for path planning lies in miles rather than feet and that open terrain instead of indoor environments constitute the robot's domain. Using different levels of representation at different levels of control makes sense only if provision is made for the exchange of information between representations. This is necessary to preserve the integrity of the world model. Although planning control changes predictably from one level to the next in this system, it is unclear how any one of the representation levels could benefit from new information incorporated into another.

§2.3 *Assessment of Representation Techniques*

Ad hoc representational strategies (particularly those developed for limited task domains), although useful for sensor-dependent approaches, do not support the future of mobile robotics. Most are too narrow in scope to contribute significantly to the field. Hopefully, a cogent approach to representation that is broadly applicable and easily extendible will appear.

One of the biggest problems faced by all developers is a means for supporting semantic processing. A consensus appears to exist that a rule-based reasoning capability is needed to achieve the full potential for decision-making in navigation (e.g. see [86]). None of the representations discussed above are geared specifically for this form of reasoning. Although representations have been developed for abstract spatial reasoning systems

used for path planning purposes, few of the implemented robotic systems encountered does more than offer lip service to symbolic reasoning. Of those that do [117,137], the problem is considered in isolation, and no real effort has been made to drive a robot equipped with multi-modal sensors.

Practical systems must deal with multi-level representations in order to support different types of processing requirements. Hierarchical or distributed control systems are a prerequisite for multi-level representations (see Section 1). The problem confronting designers lies not in which representation to choose for which level as much as in ensuring that the information maintained is consistent throughout the system. This favors global data structures. The use of a map-builder or cartographer, whose sole function is to maintain the consistency of the multi-level representations in a changing world, is crucial to a properly functioning system. Cartographers (map-builders) have been found in both hierarchical and distributed planning control systems [66,117,127]. Standard database techniques should be used, such as locking at different levels of granularity to allow for efficient operation during the frequent updating anticipated.

Certainly both freespace and the obstacles themselves should be represented somehow. Many researchers have recognized this need and as a result various hybrid systems exist. Whether the hybridization is done using generalized cones, vertex graphs, Voronoi diagrams, regular grids or whatever, poses a relatively fine point at this stage in the history of robotic navigation. It is expected that representation and control architecture development will be evolutionary, changing in response to new and different types of sensors, vehicles, and tasks. Any system designer would do well to recognize this phenomenon and make an effort to allow for adaptation and accommodation of new ideas and technology through inherent design flexibility.

§2.4 *Path Generation Strategies*

Generating the path from most of the representations above is a classic artificial intelligence problem, requiring the search of a possibly large state space. The representation used determines the nature and extent of the space to be searched. Some representations will require greater search effort than others.

In the vast majority of cases, the A^* algorithm is the search technique used. Although Dijkstra's graph search is occasionally mentioned (e.g. [88]), it is generally agreed that A^*

is the algorithm of choice. The basic question instead is which heuristic should be used. Generally the most straightforward is employed: the remaining straight-line distance from the current position to the goal. This is sufficient to guarantee an admissible solution based on a distance metric for optimality. In several cases however, (including AuRA). criteria other than just straight-line distance are considered. Factors such as traversability of terrain, safety, etc. are weighed in developing an appropriate cost function. It has been stated that the heuristics are developed largely on an ad hoc basis [66], and more theoretical research is needed to understand just what constitutes an optimal path for mobile robots. The question of how important an admissible solution is to this particular search problem remains unanswered as well. Chattergy at the University of Hawaii has explored some of the heuristics that can be applied to mobile robot navigation [34].

One interesting approach to path generation developed by Thorpe at CMU is based on a path relaxation process [89,126]. It develops a better path from an original first cut solution initially put forth by operations performed on the underlying grid representation. Subgoal nodes, (produced from an A* search of the grid), are displaced according to specific constraints and the cost function recomputed. This process is repeated until convergence is observed. The net result is the removal of jagged edges as the path settles into a relaxed state. This idea of an iterative relaxation process for path refinement is extendible to other representations as well, and is used in AuRA (in a limited way) for multi-terrain path planning (Chapter 3).

In at least two cases, dynamic programming techniques were exploited for path generation instead of pure heuristic search [104,121]. In the first case cited, dynamic programming is used in conjunction with standard AI search techniques. Based on the claims made by the authors, dynamic programming seems a viable alternative to pure search, but it still appears as the exception rather than the rule in exploring paths for the mobile robot's world.

§3. Summary of vision research in mobile robotics

Vision research in mobile robotics has tried to accomplish the very difficult. Working in a partially known and uncertain environment, with a camera that is in motion and subject to bouncing, the vision system must provide information to the robot that is both

useful and accurate. Nonetheless, in order to provide general robot abilities, vision must be exploited to the fullest. Several different tasks for the mobile robot are well-suited for vision. These include:

- obstacle avoidance
- path (road) following
- goal (target) identification
- landmark recognition for vehicle localization
- dynamic map making

Many different research groups are exploring these areas. One way of greatly simplifying the task is by restricting the domain. The discussion that follows first describes the approaches that are domain-specific and then looks at those research efforts that are less restrictive. As this chapter is concerned primarily with representation and control, the vision systems discussed below should be viewed as a sampler, rather than a comprehensive survey of current vision research in mobile robotics.

§3.1 *Man-Made Environments*

Restricting the domain of a mobile robot to a known environment significantly reduces the number of possible interpretations of visual data. Several research vision systems have been constructed to exploit these constraints. A few are described below.

Obstacle avoidance using vision is one of the major areas of work. Tsuji [129,130] has developed a working system for a mobile robot operating in a hallway environment. It is based on two assumptions: first, there are many visible vertical lines in the scene; and second, the floor is almost flat. Although the second assumption is reasonable, the first may be brought into question for any area other than a hall. If that hall has smooth walls, it may be impossible to navigate even there. What is of note is the rapid sampling of data (1 image/second) while the robot is in motion (velocity 0.3m/sec). Optic flow analysis is used to provide information regarding collision avoidance and obstacle detection (the depth-from-motion algorithm described in Chapter 6 is also optic flow based). A 3 by 3 Sobel operator is used to detect vertical lines. Features are then extracted using Moravec's

interest operator in the neighborhoods surrounding these lines. Errors of only 10% in the distance to objects near the robot are claimed. This algorithm is also capable of detecting objects moving in directions other than the direction of motion of the robot. The inapplicability of this system to other environments makes it useless for the general case.

Another approach used for indoor obstacle avoidance is described in [89,125]. This system, developed at CMU, uses 2-eyed stereo, a marked improvement in computation time over the initial 9-eyed system previously used [93]. A thorough study of the constraints affecting processing speed is provided. These constraints involve both the imaging geometry (camera and robot) and motion geometry (predicted position). Nonetheless, the system is still too slow, taking in excess of 30 seconds per step. One second per step is claimed as the required speed for real-time obstacle avoidance.

§3.2 *Following Roadways*

Several research efforts have been directed to the task of following roadways. Four are reviewed below.

Researchers at the University of Maryland [74,135,136,137] have looked at the recognition and following of roadways specifically with the DARPA autonomous land vehicle in mind. Their process for road following is broken down into two distinct phases: Bootstrap-Image Processing and Feed-forward Processing. The purpose of the bootstrap system is to find the road's location without prior information about the vehicle's position. Once the road is identified and the robot is in motion, a feed-forward strategy is employed that relies heavily on the inertial guidance system for dead-reckoning. This restricts the possible location of features to smaller windows that can be processed more rapidly. The constraints based on error studies have been set at 0.25 meter and 1 degree of orientation - fine for inertial guidance but impractical for most lower cost systems. In order to maintain these tolerances, an expensive pan and tilt mechanism would be required as well. The control system proposed is hierarchical, consisting of a pilot, mission planner and navigator. The navigator is the sole interface to the three visual processing modules. Line extraction, by combining evidence from multiple image windows to yield the long parallel lines of the roadsides, is the principal example cited in the paper.

A similar strategy is used for the same task, but at higher speeds, by Dickmanns

and Zapp [39]. This system also operates using a window concept to meet the real-time processing constraints. Greater consideration is given to vehicle dynamics and the groundwork is presented for the eventual high-speed (up to 65 km/h) guidance of ground-based vehicles by computer vision. A dedicated microprocessor is assigned to each feature tracked in its own window. Anticipatory control is utilized via a "preview" window (based on vehicle dynamics). Stationary obstacle recognition is considered as well.

The ALV group at CMU has implemented a road-following system on two working robots, Neptune and Terregator [133]. Although the approach is simpler in concept than either of the above two systems, a working robot plant using these concepts exists, not merely simulations. Initially, the only environmental sensor used was a single black and white TV camera [133]. This has since been extended to color [132].

More recent work at CMU on the NAVLAB [134] has demonstrated the ability to follow roads that are streaked with shadows and poorly registered in the color spectrum. A pattern classification scheme based on pixel values on a color surface distinguish sunny and shaded road from non-road regions. The image pixels associated with the road are grouped into a single region which is then used to servo the vehicle.

MELDOG [121] tracks road edges with a CCD camera whose field of view can be changed to detect the road edge. The velocity requirements are much less stringent than for the ALV, but real-time processing is still required and "lurch-mode" unacceptable. This road-following algorithm is supported by other vision algorithms that determine landmarks and handle obstacle avoidance based on ultrasonic sensor detection.

Road following is essentially as domain-specific as the man-made environments described above. A case could easily be made stating that roadways *are* man-made environments. Consequently, by understanding the nature of the roads to be traversed and restricting the robot's motion to those roads, it is realistic to expect that reasonable real-time speeds could be attained. Off-road navigation (or on poorly defined roads or trails) would then be an entirely different problem.

§3.3 *Goal (Target) Identification*

Goal identification, primarily of military importance at this stage in development, is a means to determine targets in a changing environment. There is no absolute reason why this must be the only explored domain. A child running to hug its mother in a crowd

is an example of navigation being used to accomplish more pacific goals. It can also be argued that goal identification is the complement of obstacle avoidance; seeking objects rather than avoiding them. A goal/target can be defined as an object or collection of features of an object or environment which is to be recognized and approached. The goal has a stored representation for either the general class of the object in question or a specific instance of it.

Many of the issues in goal recognition are similar to those found in landmark identification; the context in which the results are to be used marks their distinction. Landmark identification is used principally for orientation of the robot, helping it to establish and maintain its position en route to a predetermined goal. Goal identification, however, must be carried out before final planning can be accomplished. The planner cannot determine an optimal path if it does not know where the goal is. Certainly exploratory path strategies could be employed to survey the area until the goal is encountered. However, if the robot has no means to determine that its goal is in view, it could not attain it.

Thus far, infrared imaging seems to be the dominant sensor modality for target identification. This regresses to the desire of the military to target hot objects (such as a tank or aircraft) in a relatively cold environment. Work done at Honeywell [20] involves optic flow and a partially rule-based system for obtaining range data from passive infrared sensors. This project utilizes the speed of an aircraft (or missile) and the sensed imagery to yield a dense range map (85-95% accuracy is claimed). Although currently the system is geared towards missile guidance, the designers claim it can be extended to ALV research for ranging and obstacle avoidance.

Goal recognition is important if only approximate knowledge of the final destination of the robot is available. If *a priori* knowledge indicates that the goal is in a certain room, the planner can march the robot into that room and then have goal identification algorithms determine the exact location of its target. This mimics the way in which humans might search for something when told it is in an approximate location. For example, if instructed to attend class in room GRC 301, an *a priori* map of a campus can get us to room GRC 301. Finding a vacant seat to sit in is another story. Although goal identification for autonomous machines is currently dominated and sullied by destructive applications, its necessity for truly autonomous robots cannot be underestimated. Little work has been done towards implementing a practical mobile robot system functioning

in an unstructured environment using this concept.

§3.4 *Landmark Recognition*

This task actually subsumes the role of goal/target identification, the fundamental difference being that motor behavior is not necessarily a consequence of landmark recognition. Landmark recognition for vehicle localization is a more ambitious and more difficult task than just identifying a road or lines in walls. What constitutes a landmark and how to identify them can involve techniques as simple as spectral analysis or template matching to as complex as the full problem of natural scene analysis. Currently, real-time constraints will tend to favor the simpler approaches.

Landmark recognition in a highly simplified situation has been implemented in MELDOG [121]. White painted lines of a specific length and width at known locations have been painted on roads to assist in localization of the vehicle. A specific landmark sensor system, using two sensors, (one each at the front and rear of the vehicle facing down) provide landmark detection. This is extremely limited. The designers discuss the possibility of utilizing real-time landmark recognition with ultrasonic data (principally walls), but do not extend vision as a practical means of handling the localization problem any further than using the simple "white-stripe" approach described above.

An ambitious system encountered for landmark recognition has been developed at the University of Maryland [3]. It relies on a Hough transform based template matching algorithm to identify landmarks whose representations have been stored in a landmark database. After matching has been accomplished, a maximally consistent set of landmarks is constructed which is then used to provide information, by techniques based essentially on triangulation, on the vehicle's position. It can also be used to request the visual system to find additional landmarks that would improve the position's certainty by repositioning the camera. The full navigational system is hierarchical, involving long range, intermediate range, and short range planning.

The system consists of three major modules: the Matcher which establishes expectations for likely landmark positions; the Finder which directs camera angle and focal length to attempt to find the landmarks in the positions put forward by the Matcher; and the Selector which chooses a set of landmarks that would be most appropriate for localization purposes. Although it remains to be incorporated into a working robot, this

project is the first system that takes visual landmark recognition in the more general case seriously (see Chapter 7 for AuRA's approach and a comparison to the Maryland system).

§3.5 *Dynamic Map Making*

Having a robot explore a region where it has never been before requires special capabilities. The vehicle must be able to acquire a model of its surroundings in order to determine the proper trajectories for it to navigate through. Vision has not been fully exploited as a sensor modality for localization to date. Most practical work has been concerned instead with sonar or laser rangefinder data [36,43,131]. Of the systems that rely heavily on visual data for dynamic map making, the results are quite sketchy.

Brooks dogmatizes important issues in visual map making in [25]. His approach is restricted to an indoor environment with the intent of developing a "rubbery, stretchy" relational map. The algorithms for vision expressed in the paper have not yet been implemented nor tested on real data. In several instances, only the general issues are described without tackling a specific computational approach for vision. Brooks' control paradigm espoused in [24] provide the framework for the implementation of these concepts.

Moravec, as previously mentioned [93,125], used stereo to obtain a sparse depth map to represent the location of obstacles in order to plan a path around them. In the later work, CPU times of 30 seconds to one minute per step is required. Sensed obstacles are represented as circles and path planning is conducted within the confines of "tangent space" - the paths that connect the tangents of the enclosing circles. Although obstacle avoidance is feasible with this approach, much work remains to be done in order to develop any semblance of higher level knowledge acquisition through vision. Moravec in [91] philosophizes on the roles of vision and locomotion in mobile systems (biological or otherwise).

HILARE [52] complements vision with a laser rangefinder as a means for successfully developing a world model of polygonal obstacles and walls. A low-level vision system detects planar surfaces of objects by contour extraction and filtering, serving as a guide for the laser range finder. The obstacle's slope is obtained which then is used to produce a 2D ground projection fit for incorporation into the world model.

§4. Summary and Conclusions for AuRA

This chapter has presented several yardsticks against which to compare representational strategies. These include:

- Representation of uncertainty
- Efficiency
- Support for multiple frames of reference
- Sensor independence
- Robot vehicle independence
- Support for semantic information
- Facilitation of parallel processing
- Support of localization

Chapter 3 will introduce the AuRA architecture. AuRA incorporates many of the best features of existing systems. The use of a regular grid in short-term memory to reflect uncertainty, while providing a hybrid vertex-graph free-space (meadow map) model to facilitate intermediate level navigation, plus embedded landmark and semantic information for mission-level decisions, affords across the board support for navigational and spatial reasoning issues. A hierarchical planner is used to develop a global path, while actual path execution, monitored by a distributed control system, uncouples reactive/reflexive piloting from map-navigation. A cartographer separately maintains the global data structures used to support the map-navigation. Robot vehicle independence is guaranteed by providing a standardized vehicle interface for communication with the robot. Sensor independence is maintained through a sensor-independent short-term memory level wherein all relevant sensor data is fused. An egocentric frame of reference is maintained within the perception and motor subsystems, while a global model based on *a priori* knowledge exists within the cartographer.

Much credit must be given to those researchers whose previous and ongoing efforts are reviewed in this chapter. Without their advances into new territory, (some advancing more blindly than others), many of the mistakes and difficulties that were made were

doomed to be repeated by others. The diversity of their representation and control efforts reminds this author of the early days of aviation. Perhaps a clearly superior approach to the problem of intelligent sensor-driven mobile robot navigation will eventually become apparent. Until that time, best wishes to those daring young men and women and their walking, crawling, creeping, rolling, and hopping machines.

CHAPTER III

THE AURA AUTONOMOUS ROBOT ARCHITECTURE

AuRA (Autonomous Robot Architecture) is a system architecture that provides extensions to the UMASS VISIONS system that are primarily concerned with safe mobile robot navigation. The VISIONS group at the University of Massachusetts has an extensive and ongoing research project in the interpretation of real-world images [41,42,102,140]. These extensions include the addition of representations specific to navigation, the incorporation of motor schemas as a means of associating perceptual techniques with motor behaviors, and the introduction of homeostatic control utilizing internal sensing as a means for dynamically altering planning and motor behaviors.

This chapter is divided into the following sections. Section 1 presents an overview of the AuRA architecture. Section 2 describes how navigation is accomplished within AuRA, specifically the roles of long-term and short-term memory and the operation of the navigator, pilot and motor-schema manager. The issue of spatial uncertainty is addressed in Section 3. Section 4 discusses the theoretical motivation for AuRA. The hardware and system implementation issues are described in Section 5. A summary concludes the chapter.

§1. Architecture Overview

A block diagram of AuRA is presented in Figure 2. AuRA consists of five major components: the planning, cartographic, perception, motor and homeostatic subsystems. The planner consists of the motor schema manager, pilot, navigator and mission planner and is described in Section 2 and in more detail in Chapters 4 and 5. A cartographer, whose task is to maintain the information stored in long- and short-term memory and

supply it on demand to planning and sensory modules, provides the additional functionality needed for navigational purposes. Long-term memory (LTM) contains the *a priori* knowledge available to the system, while short-term memory (STM) contains the acquired perceptual model of the world overlaid on an LTM context. The cartographer is also responsible for maintaining the spatial uncertainty in the vehicle's position.

A perception subsystem, (in the future consisting of the VISIONS system, sensor processing and sensors), is delegated the task of fielding all sensory information from the environment, performing preliminary filtering on that data for noise removal and feature enhancement, then extracting perceptual events and structuring the information in a coherent and consistent manner, and finally delivering it to the cartographer and motor schema manager. It is also the subsystem, in conjunction with the cartographer, where expectations are maintained to guide sensory processing.

The motor subsystem is the means by which the vehicle interacts with its environment in response to sensory stimuli and high-level plans. Motors and motor controllers serve to effect the necessary positional changes. A vehicle interface directs the motor controllers to perform the requested motor response received from higher level processing.

The homeostatic control subsystem is concerned with the maintenance of a safe internal environment for the robot. Internal sensors provide information which can dynamically affect the decision-making processes within the planner, as well as modify specific motor control parameters. Homeostatic control is to be implemented only after the motor schema manager is moved from simulation to real-time implementation and the vehicle is equipped with the necessary internal sensors. The mission planner is currently rudimentary and has a low priority for development.

The first pass implementation of the perceptual system does not draw on the entire VISIONS system. Although the VISIONS system is ultimately expected to fuse multi-sensory data to yield a rich 3D model of the perceived world, presently relevant vision algorithms are extracted from the VISIONS environment and used outside of its context. The algorithms used are simplified versions, gaining speed at the expense of robustness. The real-time needs of mobile robotics can be handled by this strategy as the vision algorithms are not yet developed on parallel hardware. (When the UMASS Image Understanding Architecture is available, parallelism will then be exploited). Thus the cartographer assumes greater responsibility than might be needed in future designs

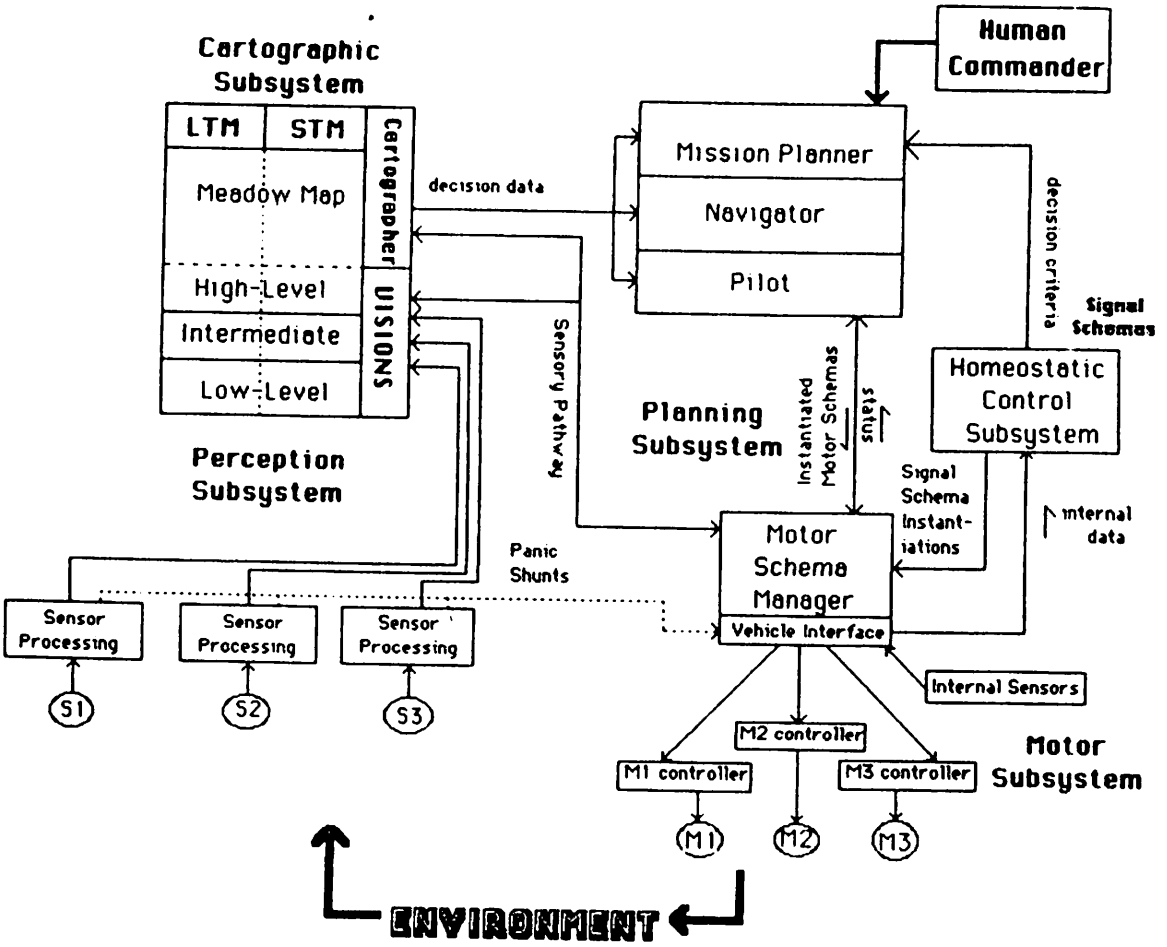


Figure 2: System architecture for AuRA

Block diagram of the system design for the Autonomous Robot Architecture (AuRA).

when many of the cartographer's chores are subsumed by the VISIONS system. Figure 3 shows AuRA's initial implementation strategy.

The subsections that follow describe briefly the ultimate roles of the various AuRA subsystems (with the exception of the planning subsystem which is discussed in Section 2.)

§1.1 *Cartographer*

The cartographer is the manager of the non-VISIONS representations and high-level controller of the map maintenance processes. Its responsibilities include:

- Preservation of the integrity of the perceived world model, reconciling temporally conflicting sensor data.
- Initiating and scheduling processes whose duty it is to:
  - incorporate data from the perception subsystem into short-term memory
  - instantiate models from LTM into STM
  - provide sensor expectations and to guide schema instantiations
- Maintenance of uncertainty at all levels of representation
  - spatial uncertainty map maintenance for robot localization
  - STM environmental uncertainty handling (object location)
- Initial LTM Map building (i.e. knowledge acquisition)

Additional information on the operation of the cartographer appears in Chapters 4 and 7.

§1.2 *Perception Subsystem*

Environmental sensor processing occurs within the confines of the perception subsystem and consists of three submodule types: sensors, sensor processors, and the VISIONS system. Currently, simplified versions of low-level vision algorithms that are tuned for real-time performance, at the expense of robustness, are used, until a full real-time scene interpretation VISIONS environment becomes available.

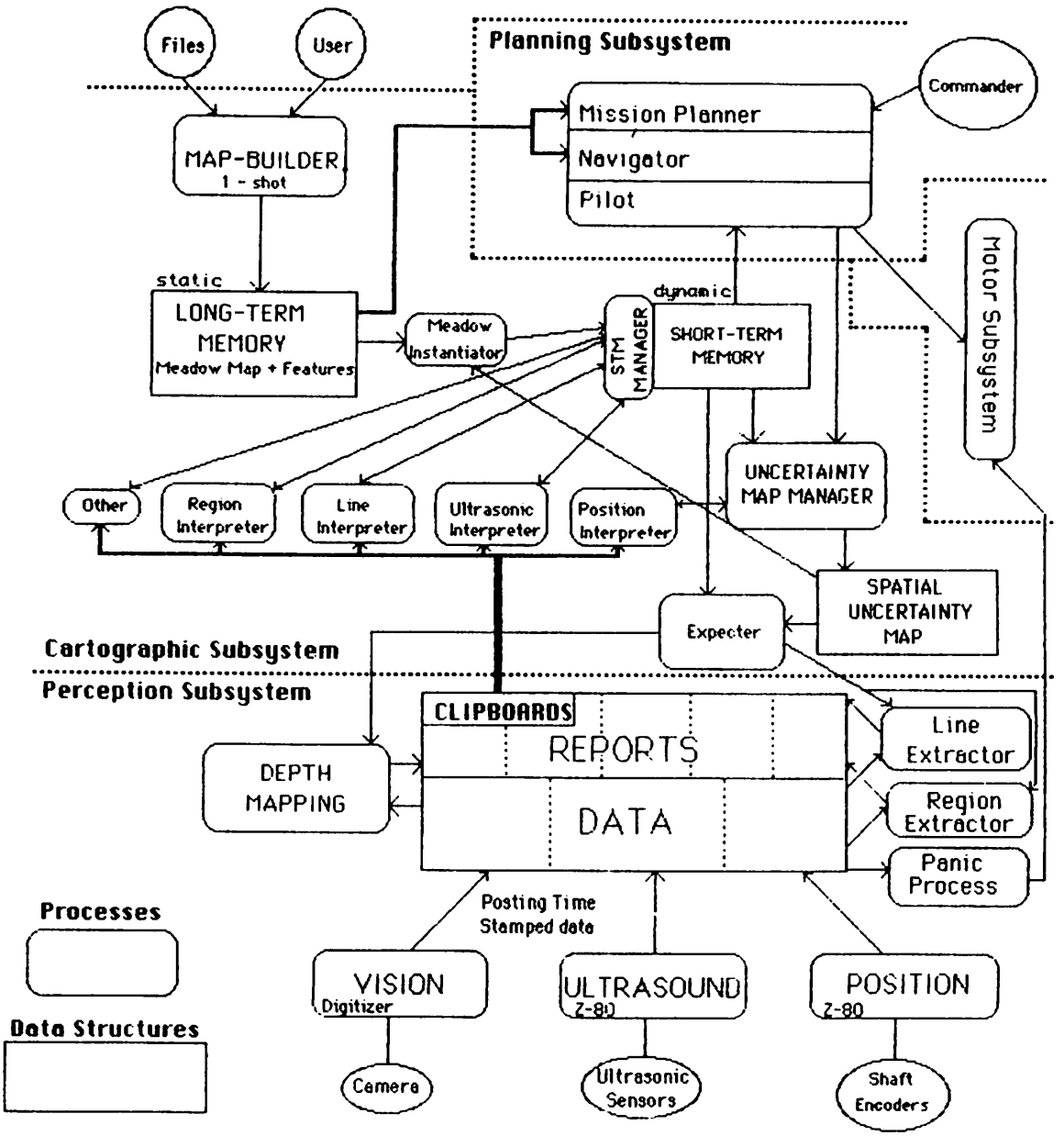


Figure 3: First pass implementation of AuRA architecture

Sensor processors preprocess the sensor data into a form that is acceptable to the receiving modules. The principal goal for these sensor-specific *filters* (e.g. for vision, ultrasonic or dead-reckoning sensors) is to simplify the job facing the remainder of the perception subsystem by converting relevant data from diverse sensors into a more useful form. The conversion of time-of-flight for a sonar echo to feet or the temporal averaging of images are typical tasks.

The VISIONS system is ultimately intended to be the heart of the perception subsystem. Multiple levels of processing acting on the sensor data and their associated interpretations are present. Perceptual schemas are instantiated and maintained within this system. The net result is a collection of plausible hypotheses and interpretations for sensor data with associated confidence levels that reflect their uncertainty. Data can be drawn off by the planner at any representation level within the VISIONS system, ranging from low-level pixel data and intermediate-level lines and surfaces, to high-level full scene interpretations.

Information foretelling imminent danger will pass directly to the vehicle interface from the sensor processors via panic shunts without the mediation of the cartographer, VISIONS system or motor schema manager. These panic shunts are intended to emulate reflex arc activity, bypassing higher level processing.

Chapter 6 describes the perceptual strategies used in AuRA in detail.

§1.3 *Motor Subsystem*

The motor subsystem is delegated the responsibility of effecting the commands of the motor schema manager. In the case of the UMASS Denning Research Vehicle, it consists of three major components: motors, motor controllers, and vehicle interface. The steering motors, drive motors and motor controllers are provided by the vehicle's manufacturer.

The vehicle interface, in its most general version, translates the commands from the motor schema manager into the specific form required for the vehicle. This module is the one component of the overall architecture most profoundly influenced by the specific robot vehicle chosen. Vehicle independence is a design goal for all other AuRA modules. Refer to Chapter 5 for a more detailed view of the motor subsystem.

§1.4 *Homeostatic Control Subsystem*

In order for robots to be truly autonomous, not only must they be capable of intelligent action, but they must be self-sustaining. Placing robots in environments that are unsafe for humans has been a longstanding aim of robotics. Little concern has been devoted to the maintenance of the routine functions that are essential for the ongoing "survival" of a robotic system. Most of these functions fall into an entirely different classification than high-level planning. The homeostatic subsystem of AuRA is concerned with homeostasis - the maintenance of a safe internal environment for the robot. This aspect of autonomous robot design deals with survivability issues. How can a robot best utilize its limited energy resources in light of changing environmental conditions? In high temperature environments, what actions can the robot take to minimize its risk? How will dangerous situations and limited resources affect planning? These concerns (and others) are addressed by the homeostatic control subsystem.

Concern for behavioral changes in planning due to the internal state of the robot has not been encountered elsewhere in the literature. Most systems assume optimal conditions at all times, others (e.g. [117]) operating in hazardous environments simply determine whether it is safe or not to enter a particular location, while still others (e.g. [104]) make plans based on fuel reserves and other factors but don't consider the robot's dynamic behavior.

In the homeostatic control subsystem, internal surveillance of the robot is constantly maintained by appropriate sensors. "Life"-threatening conditions such as excessive temperatures, corrosive atmospheres, or low energy levels, can dynamically alter variables in the motor subsystem and affect decision-making within the planning subsystem. This extended functionality will provide the robot with enhanced survivability through a greater capacity to respond to a changing environment.

Issues in the design of the homeostatic control system are discussed in [14]. Several significant features include:

- Information is transmitted via a non-hierarchical broadcast mechanism.
- Controllers are targeted by the presence of specialized receptor schemas that are used to accept and then indicate as to how the information received should be processed.

- Negative feedback control is managed by procedures embedded in the transmitter schema.
- Sensor inputs can trigger the transmitter schema, but maintenance levels are handled by the transmitter schema after initiation without additional intervention.
- The types of information to be handled are primarily concerned with the regulation of the robot's internal condition; in other words, homeostasis.
- Ongoing motor schemas rates (or other processes) are affected through the parameters specified in the receptor schemas.

Although initial system designs will assume optimal conditions for the homeostatic control system, its design considerations will be dealt with from the start to simplify the integration of this concept into later versions.

§2. Navigation

This section provides an overview of the process of navigation for our system, concentrating particularly on the relationship of visual perception to the robot's path choice and successful path completion. The detailed roles of the navigator, long-term and short-term memory are described in Chapter 4, and the motor schema manager's function is presented in Chapter 5.

There are two distinct levels of path planning available: map-navigation, based on *a priori* knowledge available from the cartographer and embedded in long-term memory; and sensor-data-driven piloting conducted by the motor-schema manager upon the receipt of instructions from the pilot. The motor schema manager is perhaps best viewed as the execution arm of the pilot, responding to the perceived world in an intelligent manner while striving to satisfy the navigator's goals. First, let's examine the hierarchical planning component of the planning subsystem.

A hierarchical planner, consisting of a mission planner, navigator and pilot (Fig. 4), implement the requested mission from the human commander. The functions of the three hierarchical submodules are described below. It should be remembered that communication is two way across the submodule interfaces, but is predictable and predetermined, a

characteristic of hierarchical control.

§2.1 *Mission Planner*

The mission planner is given the responsibility for high-level planning. This includes spatial reasoning capabilities, determination of navigation and pilot parameters and modes of operation, and selection of optimality criteria. Input to this module is from three sources: the cartographer, the homeostatic control subsystem and the human commander. The cartographer provides current world status, including both short-term and long-term memory structures. The homeostatic control system provides data regarding the robot's current internal status: energy and temperature levels and other relevant safety considerations that have a bearing on the robot's ability to successfully complete a plan. No assumptions should be made by the planner that the robot has the necessary resources available to complete any plan that is developed. This is crucial for reliable long-range planning capabilities.

Mission commands are entered by the human commander through a user interface. The exact structure of these commands will be dictated by the task domain (domestic, military, industrial, etc.).

Real-time operation is not as crucial for mission planning as it is for lower levels in the planning hierarchy. Nonetheless, efficient replanning may be necessary at this level upon receipt of status reports from the navigator indicating failure of the attainment of any subgoal.

The output of the mission planner is directed to the navigator. It consists of parameters posted on the blackboard and modes of operation that determine the overall behavior of the robot. Additionally, mission specifications and commands (subgoals) for the current task are provided.

The mission planner, although a significant component of the overall architecture, has a relatively low priority for complete implementation at this time. Chapter 4 describes the rudimentary mission planner used for the purposes of this dissertation.

§2.2 *Navigator*

The navigator accepts the specifications and behavioral parameter lists from the mission planner and designs a point-to-point path from start to goal based on the current *a*

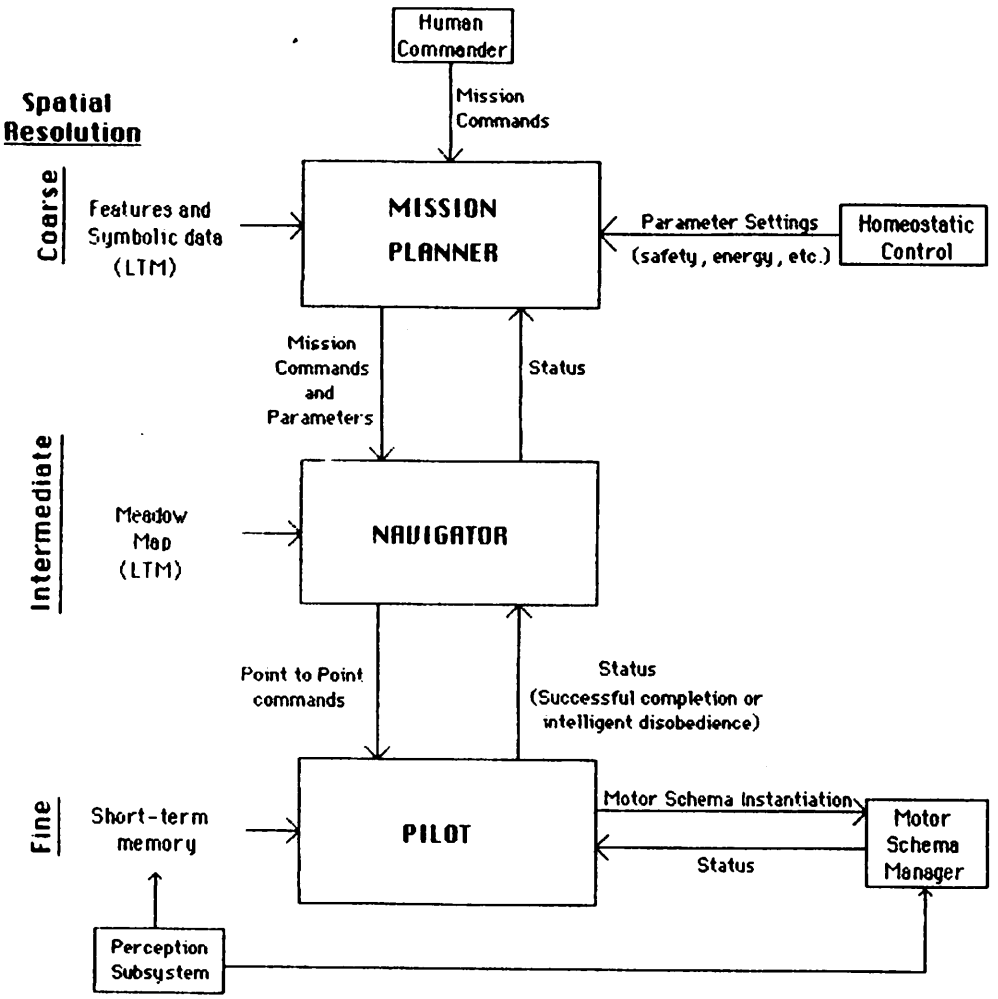


Figure 4: Hierarchical planner for AuRA

priori world model stored in LTM. The representation level used by the navigator is the "meadow map": a hybrid vertex-graph free-space world model. Status reports are issued back to the mission planner either upon successful completion of the mission specifications (subject to the behavioral constraints) or upon failure to meet the requisite goals. If failure results, the reason for failure is reported as well.

The meadow map's basic structure is an outgrowth of work by Crowley [36] and Giralt and Chatila [33,51,52]. Our work is distinguished by the incorporation of multiple terrain types, the use of specialized map production algorithms, the availability of several search strategies and the ability to easily embed perceptual and navigational knowledge. Data stored at this level reflects geometrically and topologically the robot's modeled world. A polygonal approximation of all obstacles is used to simplify both map building and path planning computation. The necessary visual representations (feature map) for path execution and uncertainty management are tied to these polygonal ground plane projection models. The meadow map serves as the basis for the robot's short-term memory context. Specific components are instantiated in STM based upon the robot's current position and the current navigational subgoal.

The feature map can be viewed as a facet of the associated meadow map. Data pertaining to the distinctive features of terrain, obstacles, landmarks, etc., constitute the feature map. The information stored here contains the attributes of the meadow map's vertices, lines, and polygons and their associated obstacles or free space.

Depending upon the robot's current position, meadows from long-term memory are moved into short-term memory. These contain information on landmarks currently visible, features of known obstacles, terrain characteristics, and the like. This data is available for prediction by the perception subsystem or for use by the pilot for schema instantiation. All meadows the robot is expected to traverse during path execution plus a limited number of adjacent meadows are made current in STM.

Output of the navigator is directed to the pilot. This output consists of a point-to-point path and other parameters that will affect the pilot's overall behavior. Essentially, the navigator is model-driven, (the model being the meadow map), passing off its goals to the data (sensor)-driven pilot. Status information is received by the navigator from the pilot indicating either the successful completion or failure of the established goals of the pilot. Upon pilot failure, the navigator may initiate replanning without reinvoking the

mission planner. Time constraints are more critical for the navigator than the mission planner, but are not as stringent as those needed for the real-time requirements of the pilot and motor schema manager. See Chapter 4 for a complete description of the navigator and meadow-map representation.

§2.3 *Pilot*

The pilot accepts a point-to-point path from the navigator and provides the robot with suitable motor behaviors that will lead to its successful traversal. The pilot selects appropriate motor schemas from a repertoire of available behaviors (based on the current long-term memory context), passing them (properly parameterized) to the motor schema manager for instantiation. From that point on, path execution is turned over to the motor schema manager. During actual path traversal, the cartographer concurrently builds up a short-term memory representation of the world based on available sensor data. If, for some reason, the motor schema manager fails to meet its goal within a prescribed amount of time, the pilot is reinvoked to find an alternate path, based on both the LTM context and STM. Approximating polygons representing sensed but unmodeled (i.e. unexpected) objects are inserted into the local ground plane instantiated meadows and the convex-decomposition algorithms (used by the cartographer to build LTM) are run upon them. These "fractured" meadows serve for short-term path reorientation by the pilot and the basis for the instantiation of new motor schemas.

Associated parameters for the slot-filling of motor schemas are provided by the mission planner, navigator and LTM. The commands issued by the pilot result in motor schema instantiation within the motor schema manager.

Typical motor schemas include:

- **Move-ahead**: Move in a specified direction.
- **Move-to-goal**: Move to an identifiable world feature.
- **Avoid-static-obstacle**: Avoid collision with unmodeled stationary obstacles.
- **Stop-when**: Stop when a specified sensory event occurs.
- **Stay-on-path**: Remain on an identifiable path (road, sidewalk, etc.).

Associated perceptual schemas (run in the context of the motor schema manager) include:

- **Find-obstacle**: Identify potential obstacles using a particular sensor strategy.
- **Find-landmark**: Detect a specified landmark using sensory data (for managing the robot's positional uncertainty).
- **Find-path**: Locate the position of a path on which the robot is currently situated using a specified sensor strategy.

The pilot requires more timely data than do either of the two higher levels in the planning hierarchy. Sensor data is passed in two ways: through the short-term representation provided by the cartographer or, in limited instances, by panic shunts which serve as reflex arcs issuing directly from the sensory subsystem.

The concept of a reflexive pilot is not novel, although this implementation is. Nitao and Parodi [98,104] describe the importance of such a pilot. The essential fact is that the pilot operates in virtually a memoryless manner, maintaining little or no information about former subgoals. The pilot is basically concerned with getting from one point to the next and reporting failure if it is unable to do so. Success or failure is based on judgment criteria passed down from the navigator. Reflex arc activity is strongly dependent on local sensory processing that is carried out within the perception subsystem.

Finally, the pilot monitors the motor subsystem status to determine if indeed the specified motor actions have been carried out as desired. The pilot reports its own status regarding the implementation of the navigator's specified plans back to the navigator.

See Chapter 4 for a description of the structure and operation of the pilot.

§2.4 *Motor Schema Manager*

Distributed control for the actual execution of path travel occurs within the confines of the motor schema manager. Multiple concurrent schemas are active during the robot's path traversal in a coordinated effort to achieve successful path transition. A potential field methodology [68,69] is used to provide the steering and velocity commands to the robot. An overall velocity vector is produced from the individual vector contributions of each active motor schema. This vector determines the desired velocity of the robot relative to its environment. When each motor schema is instantiated, at least one relevant

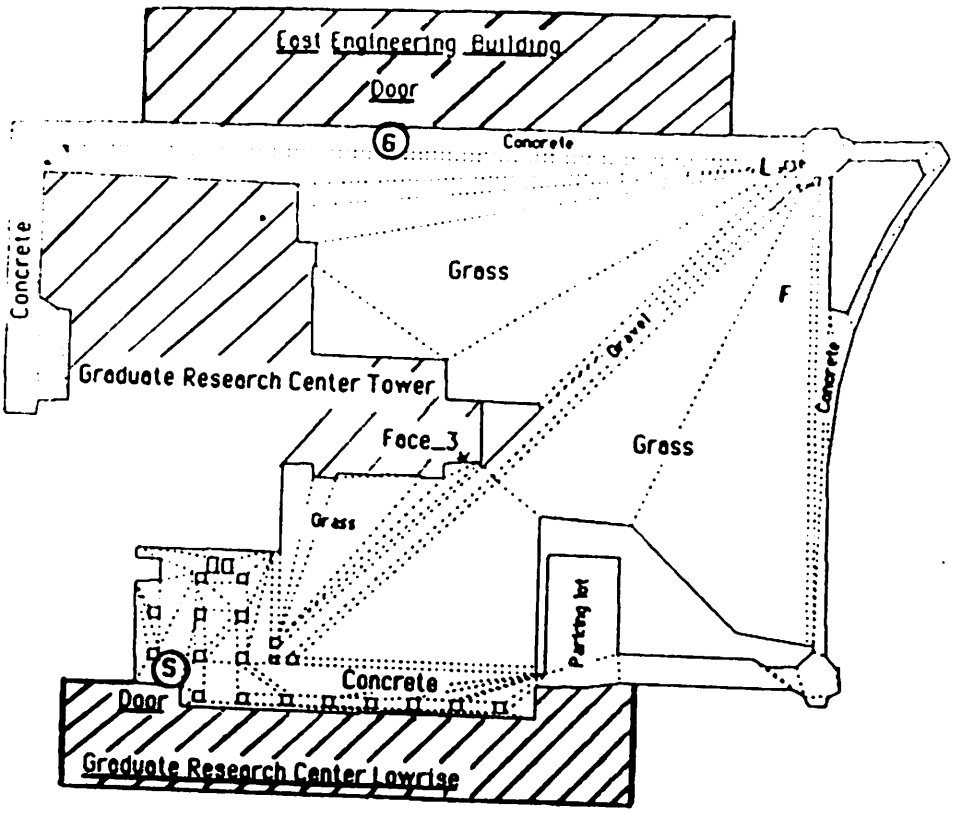
visual algorithm or perceptual schema is associated with it. Additionally, various perceptual schemas are instantiated to identify available landmarks (as predicted by long-term memory and the current uncertainty in the robot's position). These are used to localize the vehicle without necessarily evoking motor action. The role of the motor schema manager, the potential field representations used, and the underlying motivation for its use are presented in Chapter 5.

§2.5 *Navigation Scenario*

Perhaps the best way to convey the navigational process within AuRA is by example. Figure 5 represents an LTM meadow-map model of the area outside the Graduate Research Center at the University of Massachusetts. Embedded within this map, (although not visible in the figure), is additional data regarding landmarks, building surfaces, terrain characteristics, etc. This includes specific visual cues to assist the robot during its path traversal.

Suppose the robot is given the command to go from its current position (outside the GRC low-rise) to meet Professor X. Available weather data indicates that the grassy regions are currently impassable (the ground is muddy due to rain), and the robot must restrict its travel to the concrete sidewalks or the gravel path. The mission planner, recognizing this, might carry out the following: set the traversability factors for the grassy regions to IMPASSABLE, locate the fact that Prof. X's office is in the East Engineering (EE) building, determine that he is likely to be in his office at this time (by referring to the current time of day and the day of week) and then invoke the navigator to determine a path from the robot's current position to the door of the EE building. We'll ignore the indoor navigation issues here. (The current implementation of the mission planner is only rudimentary and the above discussion is presented to indicate its ultimate role as opposed to the current state of development).

The navigator, based on the instructions from the mission planner, determines a global path (Fig. 6) that satisfies these goals using an A* search algorithm through the meadow boundaries (Chapter 4 will provide the details of how this is accomplished). This path consists of 5 legs, the individual piecewise linear components of the path. Let's look particularly at leg 3, where the robot is to follow the gravel path (i.e. assume the robot has successfully traversed the first 2 legs of this path). The pilot receives the message to



Key
S - Starting Point for robot
G - Goal for robot
F - Fire Hydrant
L - Lamppost

Figure 5: Outdoor meadow map

This map represents the area outside the Graduate Research Center when viewed from above. The detail level of this particular map is low so that small objects are treated as unmodeled obstacles for global path planning purposes.

travel from point M, representing the center of probability of the robot's current position, to N, the end of the gravel path.

The pilot now has available in short-term memory "instantiated meadows" (i.e. those LTM meadows over which the robot is expected to pass during this particular leg of the journey, and several additional visible adjacent meadows, all provided by the cartographer). From this LTM data, the pilot extracts the following relevant facts:

1. Path - The robot is to travel over a gravel path bordered on either side by grass.
2. Landmark - At the end of the path, near where the robot is to turn, is a lamppost.
3. Landmark - Off to the right of the path there appears a bright red fire hydrant (a readily discernible landmark).
4. Landmark - To the left of the path, the robot will pass the GRC tower, a 16 story building (another good landmark).
5. Obstacles - It is possible, as always, that people, cars or unmodeled obstacles may be present on the path (either stationary or moving).
6. Goal - At the end of this path there is a change in terrain type, from gravel to concrete.

1 is useful for a path-following strategy, **2** and **6** are useful for goal recognition, **1,2,3,4,6** are useful for localization purposes, and **5** is necessary for obstacle avoidance.

From this information, the pilot, (see Chapter 4), determines that appropriate behaviors for this particular leg (travel across the gravel path) include:

A. **Stay-on-path(find-path**(gravel))
B. **Move-ahead** (NNE — 30 degrees)
C. **Move-to-goal**(right(**find_landmark**(LAMPPOST_107),3))
D. **Move-to-goal**(**find-transition-zone**(gravel,concrete))
E. **Find-landmark**(HYDRANT_2)
F. **Find-landmark**(GRC_TOWER(face_3))
G. **Avoid-obstacles**

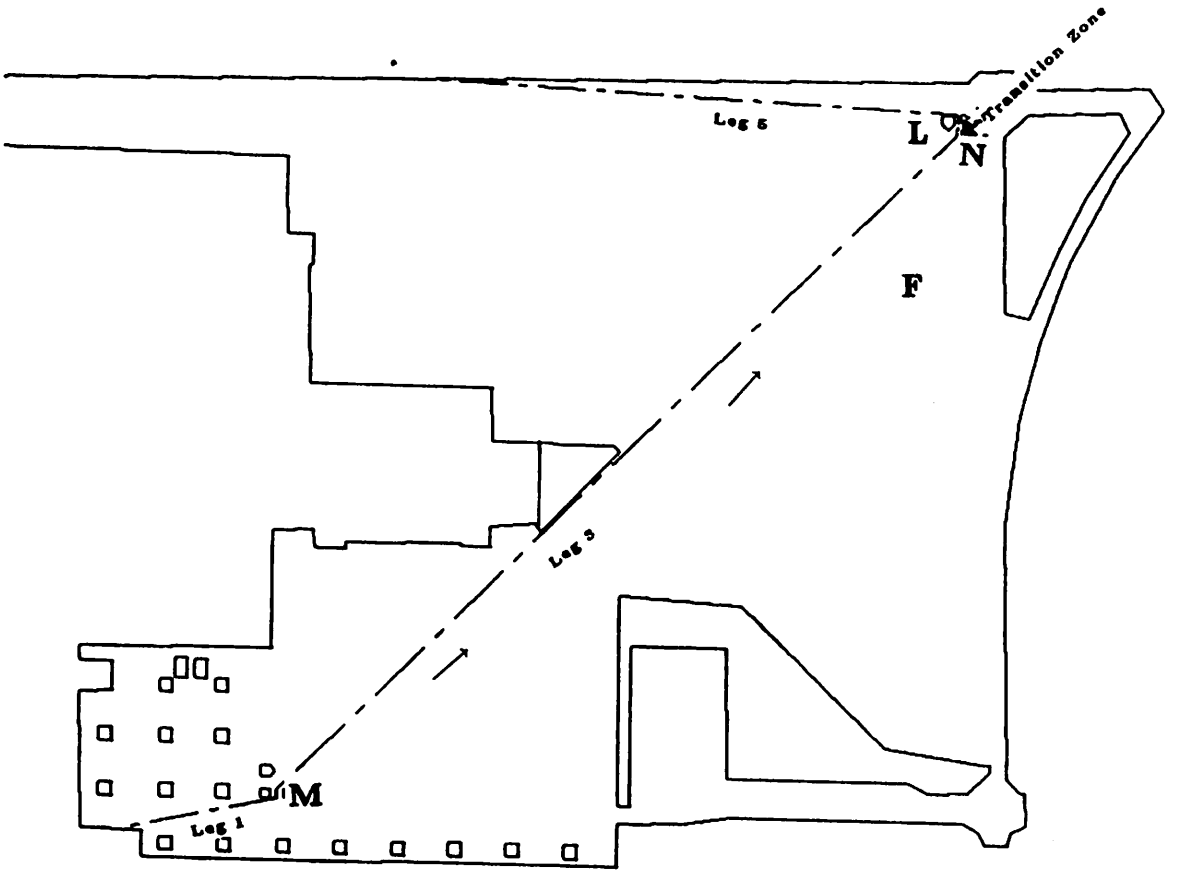


Figure 6: Global path constructed by navigator

An A* search algorithm is used to search the midpoints and edges of the bordering passable meadows to arrive at the global path.

The vector outputs of all active motor schemas are summed to produce the robot's velocity. Each component vector is computed from the robot's position relative to the sensed environmental feature. Chapter 5 describes the control issues for motor schema based navigation in detail.

Motion is first initiated by the **move-ahead** schema, directing the robot to move in a particular direction in global coordinates, in response to the pilot's need to satisfy the navigator's subgoal to move to point N. This heading is based on information contained within the spatial uncertainty map that reflects the uncertainty in the vehicle's position and orientation relative to the world map as well as the specific direction of this particular path leg. It is not critical that the heading be exactly correct; indeed significant error can be tolerated due to the presence of the **stay-on-path** motor schema. As soon as a **move-to-goal** schema becomes active (due to the recognition of the goal – the lamppost and/or terrain type transition zone), the **move-ahead** schema is deinstantiated in favor of it. Motor actions produced by the **move-ahead** schema and **move-to-goal** schema are mutually exclusive.

Stay-on-path(**find-path**(gravel)) yields 2 perceptual subschemas for one motor schema: **find-path-border** - using a line-finding algorithm to detect the position of the path's edges, and **segment-path**, a perceptual schema that uses region-based segmentation to locate the spatial extent of the path. Through the combined efforts of these cooperating schemas the position of the path relative to the robot is ascertained. As a result of the posted path position, the **stay-on-path** motor schema produces an appropriate velocity vector (based on the robot's current position within the field generated by the path) moving the vehicle towards the center of the path (Fig. 7a). This vector is summed with any other vector outputs of active motor schemas (e.g. **move-ahead**) to yield the overall velocity vector for the robot.

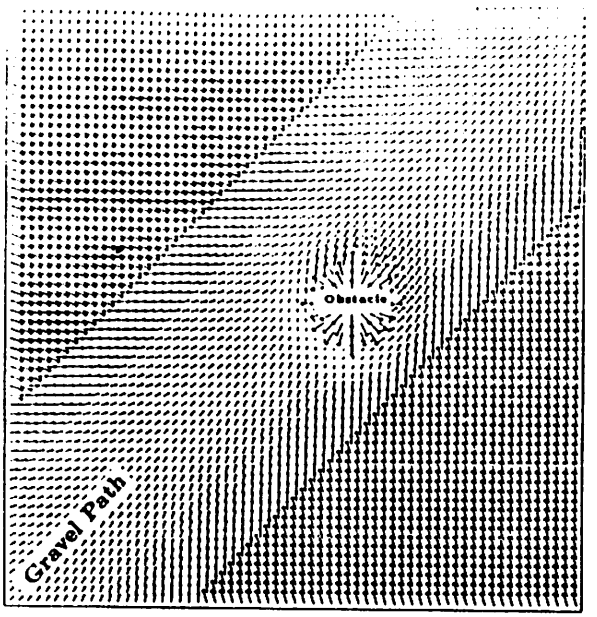
We define a schema instantiation (SI) to be the activity of applying a general class of schemas to a specific case [6,7,140]. The fact that a lamppost is present at the end of the path results in the creation of a **find-landmark** SI dedicated to finding LAMPPOST 107, whose model is extracted from LTM via the instantiated meadows in STM. This **find-landmark** schema directs the sensor processing by instantiating a VISIONS perceptual schema and/or looking for particular strong vertical lines in a given portion of the image and/or utilizing any other relevant sensor algorithm. Every time a potential

Figure 7: Potential fields produced during leg traversal

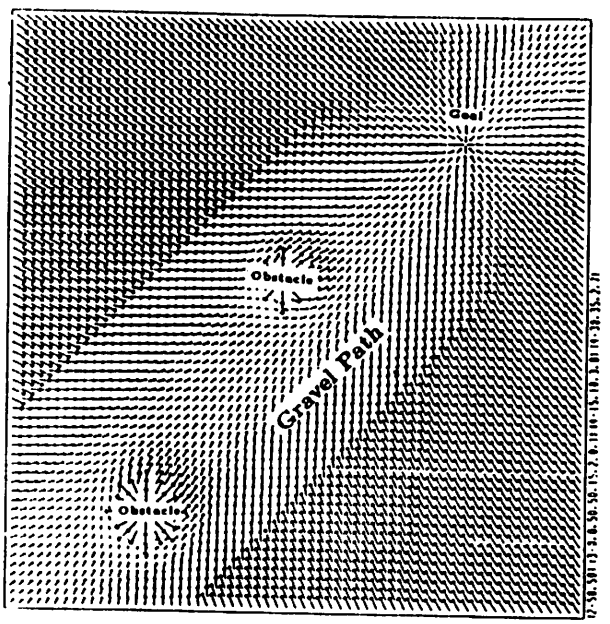
The arrows represent the desired velocity vectors that constrain the robot's motion, indicating the velocity the robot should undertake based on its position within the field. The primitive schema potential fields that are summed to yield this composite behavior appear in Fig. 47.

a). Before the goal is identified, the move-ahead and stay-on-path SIs conduct the robot on its way. A single obstacle SI is present.

b). After the goal is identified, the move-to-goal SI replaces the move-ahead SI. Two obstacle SIs are shown as the goal is approached.



(a)



(b)

LAMPPOST_107 is found in the image (perhaps evidenced by a pair of strong parallel long vertical lines in an appropriate window of the image) a new LAMPPOST_107 SI is created and monitored independently of all other similarly created LAMPPOST_107 schema instantiations. When sufficient supportive data is available confirming that one of the SIs is highly probable to be the landmark desired, all other LAMPPOST_107 SIs are deinstantiated (or placed into hibernation) and the appropriate motor schema (**move-to-goal**) starts producing a velocity vector directing the robot to a point 3 feet to the right of the identified lamppost. If the certainty in the current LAMPPOST_107 drops below a certain threshold, other SIs may be activated or created in response to particular visual events that correlate to the lamppost's model. Additionally, output from the **find-landmark** schema is used to update the robot's spatial uncertainty map, independent of any motor action that may result from the **move-to-goal** SI.

The **move-to-goal**(**find-transition-zone**(gravel,concrete)) SI is handled in a similar manner, but different perceptual schemas are instantiated and the image is searched in different regions. Texture measures for gravel are of value as well as the presence of a strong horizontal line within the boundaries of the path. The **move-to-goal** schema contains an implicit **stop-when** schema, so when the target is reached the pilot is notified that the goal has been achieved and the next leg can be undertaken.

The **find-landmark**(HYDRANT_2) schema might involve a color-based segmentation, tagging all bright red blobs in a particular portion of the image as a potential fire-hydrant. Ultimately size and shape from a model of the hydrant would be brought into focus to confirm the hypothesis to prevent incorrect identifications (e.g. a red car, or a person with a red coat). Once identified, this hydrant is then used to reduce the uncertainty in the robot's position (i.e. localization). The same kind of operation would be involved in the **find-landmark**(GRC_TOWER(face_3)) SI, but instead of using color as the primary agent for hypothesis formation, a strong vertical line (the building is 16 stories high!) or a corner silhouetted against the sky would be more suitable as the main strategy.

The **avoid-obstacles** schema is actually active most of the time. The image is windowed in the direction of the robot's motion and if any unusual events occur in that area (e.g. change in texture, color, strong line, etc.) an **obstacle** SI is associated with that particular event. That portion of the image is monitored over time by the

obstacle perceptual SI to try to confirm or disprove the hypothesis that the visual event is truly an obstacle. Concurrent with the instantiation of the **obstacle** perceptual schema is the instantiation of an **avoid-obstacle** motor schema. If the monitored obstacle's certainty becomes sufficiently high and the robot enters within the sphere of influence of the obstacle, then a repulsive velocity field is produced by the **avoid-obstacle** SI, altering the robot's course. If, on the other hand, the hypothesized obstacle eventually is determined to be a phantom and not a real obstacle at all, both the perceptual and motor schemas are deinstantiated. When an active obstacle passes outside of the influence of the vehicle, its SIs are deinstantiated as well. Nonetheless, information about the obstacle's position is maintained in STM by the cartographer at least for the duration of the leg traversal.

Figure 7 shows a potential field simulation representative of the robot traversing a path studded with obstacles as above. More details regarding the interaction and operation of the motor schemas in AuRA can be found in Chapter 5.

§3. Uncertainty Representation

Treatment of uncertainty must occur at several levels throughout the system. Estimates of positional and orientation uncertainty are crucial to accurate determination of a path. The robot not only needs an accurate representation of the world, it must also model its position relative to the world.

A new strategy for representing the positional uncertainty is accomplished through the use of a spatial uncertainty map. This map reflects the plausible limits of the robot's position within the world itself, beginning with an initial amount of uncertainty in the robot's starting position. Each "turn and run" motion of the robot will be accompanied by a possible difference between the actual amount of distance traveled and the actual amount of rotation accomplished from those amounts commanded the robot through the vehicle interface. This error will depend on several factors, not least of which is the terrain. A spatial uncertainty map, representing both the center of probability of the robot's position as well as the probable limits of the robot's position, is maintained and updated on every "turn and run" move.

An uncertainty transform is performed upon the previous spatial uncertainty map

for each move, based on the distance traversed, angle of rotation and characteristics of the terrain. Experimental data has been obtained regarding the mean error, standard deviation, etc., for both translational and rotational motion for each of the terrain types the robot is expected to encounter. These include concrete, grass, gravel and tile. These data, in conjunction with the commands fed to the pilot, are used to determine the predicted spatial occupancy areas of the robot.

The chief significance of this approach lies in its ability to use this data to restrict sensor interpretation. The robot's position is known sufficiently well to enable us to restrict the possible interpretations of sensor data or to window the visual images fed to the perception subsystem, thus decreasing processing time. If no plausible interpretation was found within these limits, special procedures can be invoked calling for additional sensor data to re-establish the robot's bearings.

If no feedback was provided by the sensors, the spatial uncertainty map would grow without bound, eventually occupying the entire world model. Consequently, sensory information and subsequent landmark recognition serve to prune this spatial uncertainty map. As correct interpretations are found within the limits of the map, a reduction in its size is made. This feedback between internal model and sensor data helps meet the real-time demands of a mobile robot system.

Chapter 7 describes in detail the maintenance of the spatial uncertainty map, its relationship to landmark perception and its overall role within AuRA.

§4. Theoretical motivation

AuRA has many characteristics which distinguish it from previous work. Theoretical consideration of cybernetic issues provides the impetus for most of the concepts employed in the AuRA architecture. It is believed that insights drawn from the existing autonomous mobile control systems, animals, can provide powerful tools in any implementation of autonomous robotic based vehicles.

Arbib [5] states that a robot requires a minimum of the following four components to function effectively in a complex environment:

1. A set of receptors and algorithms to interpret the data into meaningful relations through scene analysis.

2. A set of effectors and algorithms to act upon the environment and reposition the sensors.

3. An internal world model reflecting the results of scene analysis and robot actions.

4. A problem solver that uses scene analysis output to both update the world model and provide commands for courses of actions. It must also be able to interrupt activities when necessary in order to update and replan as necessary.

The AuRA architecture implements all four of these functions. The perception subsystem subsumes item one, just as the action subsystem does item two. The internal world model is contained within the multi-level representation scheme and maintained within VISIONS and the cartographer. Both short-term memory and a hierarchical long-term memory are present as well as a means for their modification. The problem solver's functionality is distributed between the cartographer and hierarchical planner. Replanning is initiated upon subgoal failure in the hierarchical planner, by danger signals from the homeostatic control subsystem or by reflex arc activity passed through the panic shunts directly from the sensor subsystem.

The action-perception cycle, described in cybernetic context in [6], is reflected in the overall structure of AuRA. Arbib forwards the idea that perception should be viewed as potential action. Succinctly stated "Perception activates . . . and planning concentrates" [6, p. 1459]. Robotics has been described by Brady as the "intelligent connection of perception into action" [23]. To that end, the action-perception cycle was considered as an important design model upon which to subdivide functionality within the overall system.

The action-perception cycle essentially involves perception of the world via a sensory system resulting in the modification of a cognitive map of the world. This map then serves as a basis for the direction of locomotion and other actions that in turn alter the current perception of the world. Arbib [6] and Hanson and Riseman [53] have advanced schema theory as an approach to describe the interactions involved. Although the use of all the forms of schemas in AuRA may not be true to the form that these authors report,

the interpretation presented below does model a functioning mobile robot system. It is not intended to emulate the operation of the animal brain.

Schema usage involves multiple concurrent processes, each posting hypotheses, possibly several times, guiding the overall system to converge on a valid interpretation of the perceived scene or implementation of the invoked action. The VISIONS group at the University of Massachusetts has developed a scene interpretation system implementing these ideas [140]. Ongoing work within that group has led to the development of a "schema shell" – a control and maintenance system for schemas. Other work [58,77,110] from the University of Massachusett's Laboratory for Perceptual Robotics performed in the context of distributed control of a robot hand affects our formalization of motor schema theory.

Schema theory is implemented at three locations within AuRA: the sensor subsystem (perceptual schemas - VISIONS), the action subsystem (motor schemas) and the homeostatic control subsystem (signal schemas). The perceptual schemas and motor schemas will be discussed first. Signal schemas, concerned with maintaining internal control of the robot, will be elaborated upon separately.

The computational responsibility and power residing within both the sensor subsystem and motor schema manager is not apparent upon inspection of the block diagram (Fig. 2). Each of these units contains fully independent distributed processing units requiring significantly more computational resources than both the planner and cartographer combined. Consequently, much of the work needed to implement these units in their entirety has to be deferred until the basic requirements necessary for a practical operating robotic system have been completed. Development of these components as a real-time control method is a long-term goal, but their design must be considered at an early stage to prevent an unnecessary system design change at a later date. Simulation work (Chapter 5) and experimental results (Chapter 8) demonstrate the validity of this approach.

§4.1 *Perceptual schemas (VISIONS)*

A perceptual schema has been defined as the "unit of knowledge, the internal representation of a domain of interaction, within the brain" [6]. Although this architecture is not specifically concerned with brain theory, the subdivision of perceptual realization

into discrete and manageable units is important.

Hypothesis formation, confidence (or activation) levels, and multiple concurrent processes are characteristic of all of AuRA's schema forms (perceptual, motor and signal). VISIONS perceptual schemas however, are geared specifically for the interpretation of images and ultimately the building of a sensor-independent 3D world model. As such they are not dedicated to the production of action in a robot. It is expected that VISIONS schemas, used in the context of sensor fusion, will be extended to produce sensor-independent representations drawing ultimately on multiple sensory sources. This digested sensor data can be readily integrated into the motor schemas. Since this extension is a long-term activity, the implemented version (working demonstration system - see Chapter 8) of this robot system of necessity requires some design compromises.

Operating within the perception subsystem (i.e. VISIONS system) this collection of concurrent processes posting hypotheses regarding interpretation (at different levels) of a scene is used to build an understanding of the nature and relationships of the objects perceived. When sufficiently high confidence is achieved in an interpretation of part or all of the scene, the information is forwarded to the cartographer and drawn upon by the motor schemas.

The motor schemas described below are concerned only with obtaining the information necessary to produce an action. Based on the premise of action-oriented perception, these schemas operate on the representations provided by the perceptual schemas. Specific needs or expectations regarding sensory data can be communicated via the sensory channel from the motor schema manager to the VISIONS system. This serves as a focus-of-attention mechanism based on specific action requirements.

§4.2 *Motor Schemas*

Motor schemas operate in a manner analogous to the perceptual schemas. It should be noted that the motor schemas run within the framework of the planning subsystem under the control of the motor schema manager and have a decidedly different flavor than the perceptual schemas. The motor schema manager consists of a separate schema shell in the form currently being used by the VISIONS interpretation group. Appropriate motor schema representations are employed which bear only superficial resemblance to the VISIONS perceptual schemas.

Motor schemas were a principal part of Overton's dissertation [101]. His definition for a motor schema is quite apropos. He states that a motor schema is "a control system which continually monitors feedback from the system it controls to determine the appropriate pattern of action for achieving the motor schema's goals, (these will, in general, be subgoals within some higher-level coordinated control program)" [101].

As it is necessary that a large multiprocessor be used before the realization of real-time scene interpretation and sophisticated real-time motor schema control, much of the motor schema work for this dissertation is conducted via simulation (Chapter 5). The working robot demonstration system (Chapter 8) has several motor schemas (although not operating concurrently) to illustrate conceptually how these behavioral control systems can be used to advantage.

§4.3 *Signal Schemas*

Signal schemas are an outgrowth of schema theory as applied to homeostasis - the maintenance of a stable internal system. In order to ensure a robot's safety in hostile environments, its behavior must be modified in response to the changing conditions of its own internal variables. How much energy remains, its internal temperature, and other factors can and should affect both the decision making process and effector action.

The homeostatic control subsystem [14] is responsible for the activation of relevant signal schemas (transmitter and receptor), to ensure the survival of the robot in conditions where it might be jeopardized. The little work that has been done previously [e.g. 117] has treated the hazardous environment problem as a go/no-go binary decision. What the signal schema/homeostatic control subsystem affords is behavioral modification based on the current internal conditions of the robot. Our initial system implementation assumes optimal conditions at all times. Nonetheless, it is safe to assume that mobile robots will be expected to undertake tasks that are too hazardous for humans and indeed may be hazardous to their own existence. If this architecture is to support the more general case of mobile robot, signal schemas and their distributed control within the motor schema manager are needed. It may be that a realistic implementation of homeostatic control would require significant hardware communication changes as well, conceivably involving local area networks [14].

§5. System issues

System integration issues for a system as complex as AuRA are by no means trivial. This section will discuss the approaches used in the first pass implementation. Topics include supporting hardware, the communications link and global memory sharing.

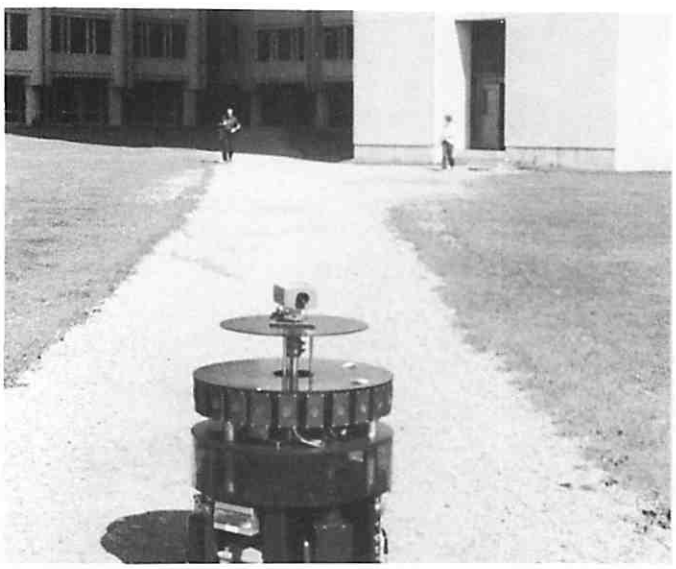
§5.1 *Hardware configuration*

The UMASS DRV (Fig. 8), aka HARV (short for HARVey Wallbanger), is a mobile robot manufactured by Denning Mobile Robotics. It is equipped with 24 ultrasonic sensors as well as shaft encoders for both the steering and drive motors. A single video camera (the VISIONS system has not yet utilized stereo images) is mounted on the vehicle and connected to a Gould IP8500 digitizer.

The basic hardware support is depicted in Figure 9. Most of the software runs on the VISION's group VAXen. The LTM mapbuilding and STM maintenance components of the cartographer, higher level components of the planning system, motor subsystem and the perception algorithms are coded in C. The pilot and spatial uncertainty subsystem of the cartographer are written in COMMON LISP. This development environment differs from future run-time environments which might contain multiple microVAXs, SUNs and of course the Sequent multiprocessor.

Code written for handling communications with the vehicle over a serial line was written in FORTRAN, drawing largely on a library of existing routines developed for such purposes. Graphics routines utilize COINS GUS device-independent graphic FORTRAN routines.

The MC68000 processor onboard the Denning Research Vehicle (DRV) handles terminal emulation using C code provided by the manufacturer. Sensor preprocessing of both the ultrasonic data and shaft encoder data is handled by 2 separate Z-80 "expert" microprocessors. The ultrasonic processor converts the time of flight for the sound wave to tenths of a foot, and coordinates the 24 sensors by alternately firing them in three banks of 8 interleaved sensors. The encoder Z-80 converts shaft revolutions for both the steering and drive motors into a cartesian coordinate system reporting in tenths of a foot and tenths of a degree. Motor controller board status can be polled directly by the VAX to detect the actual motion of the vehicle at any given time.



(a)

(b)

Figure 8: UMASS DRV (HARV)

(a) HARV outside the Graduate Research Center

(b) HARV inside the GRC

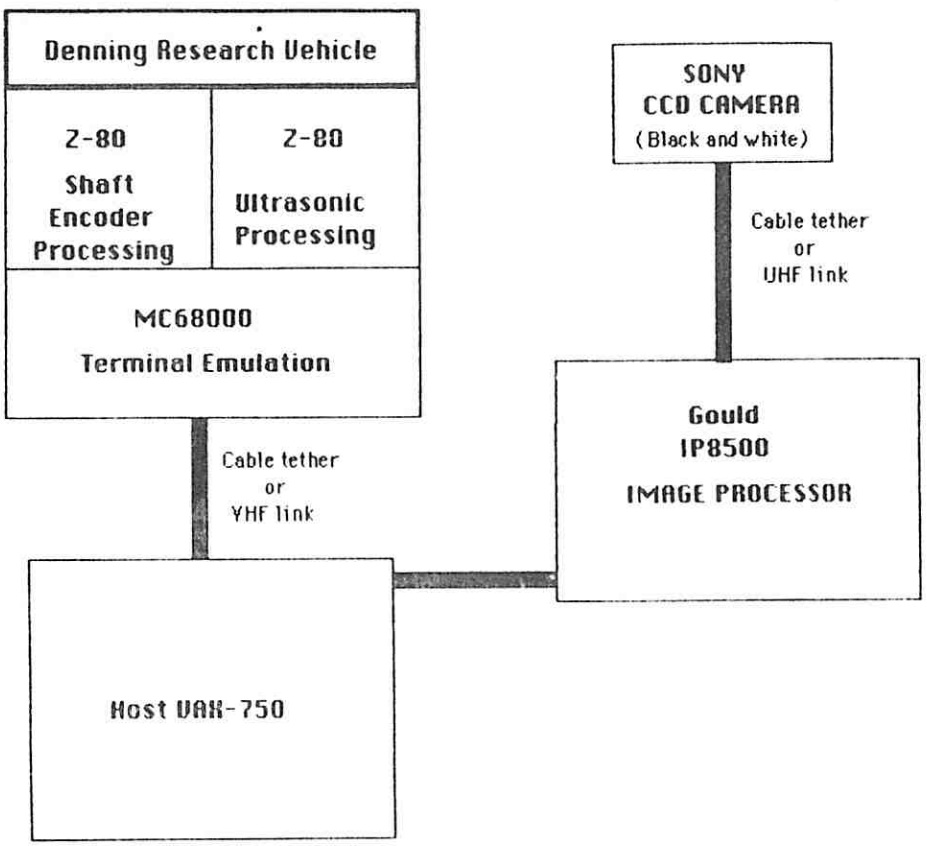


Figure 9: First pass AuRA Hardware

The Gould IP8500 image processor handles the image acquisition. Functions such as temporal averaging of several images to minimize noise as well as smoothing are available. Most of the code to accomplish this was obtained by stripping out relevant FORTRAN routines embedded in the LIPS operating system for the Gould. The use of the available lookup table facilities for the preprocessing of images for specific vision algorithms will be exploited where appropriate in the future.

A major problem for real-time performance is the limited bandwidth of image transmission from the Gould to the VAX. When preprocessing is done on the Gould, four image channel buffers are available. In some instances, if preprocessing is performed on the Gould, more than one image buffer needs to be transmitted to the VAX. The gain obtained by exploiting the parallelism available with the Gould is somewhat offset by the necessity of shipping multiple images back-and-forth between the two processors. For many of the experiments in Chapter 8, video processing was performed on a VAX after image acquisition on the Gould.

A Texas Instrument's Explorer workstation is the current home of the schema shell. Although the shell itself emulates concurrent processing, the multiple schema shell processes are scheduled round-robin on the single LISP processor of the machine. Additionally, a communication bottleneck between the TI Explorer and the VAXen occurs over the CHAOSnet link. For these reasons, the experimental schema system of Chapter 8 was implemented on the VAX. When the schema shell is finally implemented on the new 16 processor Sequent, it would be appropriate to port most of the VAX and schema shell software to that machine.

§5.2 *Communications link*

Two methods of communication with the robot are available: a visible tether and an invisible tether. As the computing power required to drive the planning and perception subsystems far exceeds the onboard capabilities of the vehicle, communications with stationary host processors is required.

The visible tether is just that: 500 feet of cabling. Actually two cables are present, one RS-232 serial communications link to transmit and receive data from HARV's onboard microprocessors to the VAX, and a video cable connecting the SONY CCD camera to the Gould.

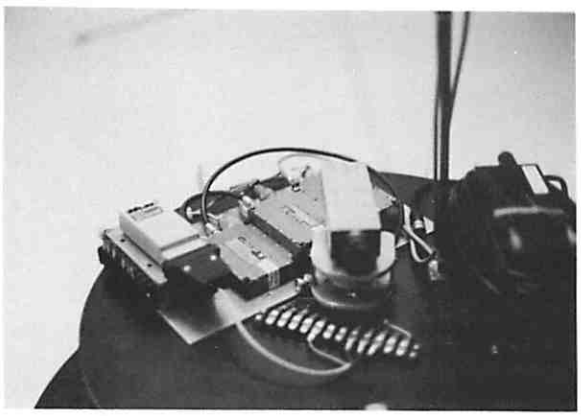
The invisible tether consists of a UHF/VHF TV-radio link broadcasting on channel 50 using satellite TV technology. A mobile station (Fig. 10) powered by separate batteries and not the robot's own power supply (to reduce noise) transmits video images while receiving motor and status commands on a separate frequency and antenna. The ground-based transceiver completes the connection between the TV-radio signals and the Gould and VAX.

§5.3 *Clipboards*

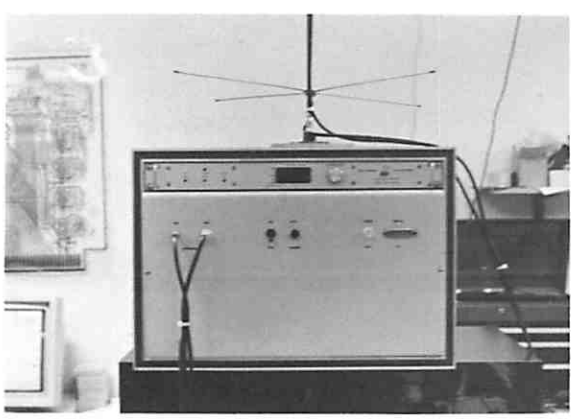
In order to effectively communicate the results from differing concurrent perceptual and interpretive processes, a global data structure termed clipboards has been developed. Clipboards are related to a heterarchical blackboard data structure, similar to the whiteboards used in the NAVLAB [115] and the blackboard in the schema shell [40].

The name "clipboards" was chosen based on an analogous situation found in meteorological stations for pilots at airports. Posted on the wall at these stations are multiple reports of sensor observations for a particular location that go back over time. New reports are added to the top of each of these clipboards as older ones are discarded from the bottom of the pile. Interpreted reports (e.g. forecasts or weather patterns) are available based on condensations of the raw data present. Depending on the level of detail required, a high level overview reflecting current or expected conditions can be obtained as well as the raw data that serves for the basis of these reports. There is a sharp and clear partitioning of these reports, and pilots can quickly find the information that is relevant to their particular needs.

Clipboards in AuRA provide a similar service. Time-stamped data arriving from sensor processes are posted in specific locations in shared global memory. (How this memory is created and accessed by multiple processes is described in the section following). Each clipboard partition of this global memory structure consists of a circular (ring) buffer. As new data becomes available it overwrites old data, but a fixed number of reports dating back over time are always available. Significant event reports can be locked onto the clipboard if so desired until released by another consumer process. The number of slots in the buffer for each report depends on their frequency and the amount of data required to store them. For example, 24 ultrasonic readings require a fraction of what a single 256 by 256 image would require. The current clipboard stores 5 raw sonar scans,



(a)



(b)

Figure 10: Communications hardware
a) Station on robot.
b) Ground base.

1024 interpreted sonar scans including position data (for STM), 10 encoder readings and 5 images.

Intermediate processes such as the line finder operate on the relatively raw data as it is received. The line finding reports are posted on the clipboard and are available for other higher level perceptual processes (e.g. perceptual schemas) associated with motor schemas running in the context of the motor schema manager. Several perceptual processes run solely within the confines of the perception subsystem, in some cases guided by expectations provided by the needs of higher level processing (e.g. tuning the buckets on the fast line finder as described in Chapter 6).

In summary, clipboards are global data structures consisting of a collection of circular queues. These queues are filled by sensor processes posting relatively raw data and by intermediate processes acting on this low-level data to produce intermediate results. For those familiar with the VISIONS system, the similarity to its hierarchical structure (low-level pixel data, intermediate tokens and high-level schema instantiations) should be apparent (Fig. 11). In both VISIONS and AuRA, each queue has its own space relative to the data it must produce, with each queue serving as a "temporal buffer". Remember that an AuRA design goal is to eventually hand off the sensor processing to the VISIONS system when it is capable of performing multi-sensor interpretation (review Fig. 2). This first pass implementation of AuRA using clipboards is consistent with that goal.

§5.4 *Memory sharing via global sections*

Sharing the global data structures present in AuRA is a primary system consideration. These structures include short-term memory, long-term memory, clipboards, and a command buffer for communication with the vehicle. As the first pass implementation consists of multiple processes in concurrent execution, a means must be available for the sharing of data. Although C typically affords pipes for interprocess communication, the bandwidth was deemed too small to be of value for the large amount of data that needs to be shared.

VMS, the operating system used on the VAX, offers extensive system services for the sharing of data. The principal technique exploited is the creation and mapping of global sections. A process initially creates a global section, which consists of a specific virtual address space mapped to a user-specified pagefile. The cartographer's creation of LTM is

a good example. Other processes can map this region to their own virtual address space via a system service call. As a result, the planner processes can independently access the exact data being used and managed by the cartographer. Interprocess synchronization is handled via a semaphore-based method. Locking of the data structure is first performed whenever it is to be modified. Other processes can access this data freely when mapped and unlocked.

It is also possible in principle to use multiple VAX processors by taking advantage of the VAXCluster architecture. By mapping multiple processes on different processors to the same shared disk pagefile, the data becomes available to all. The major problem in using multiple VAX processors is associated with dynamically changing data. Whenever data is changed on one processor, it must be write-page-faulted back to the disk to be available to the other processors. If the data may have changed since the last read by a given processor due to other processors modifying it, a read-page-fault must be made to ensure that the modified data is brought in. Although this approach would work well for static data structures such as LTM, it is generally undesirable due to the increased amount of page faulting required. Perhaps more significantly, VMS offers no easy way to force page faults from the disk when the page is already resident in physical memory. This can be accomplished by unmapping and remapping the section, but this is a costly process. Perhaps future releases of VMS will give the systems programmer even more flexibility regarding page fault control. Until that time, the multiple processes will operate on a single VAX processor at a significant computational penalty.

§6. Summary

AuRA is a mobile robot system architecture that provides the flexibility and extensibility that is needed for an experimental testbed for robot navigation. By allowing for the incorporation of *a priori* knowledge in long-term memory, a variety of different perceptual strategies can be brought to bear by the robot in achieving its navigational goals. In particular, the individual motor schemas and their associated perceptual schemas can be added to or deleted from the overall system without forcing a redesign.

A hierarchical planner determines the initial route as a sequence of legs to be completed over known terrain with predicted natural landmarks. Typical objects encountered in extended man-made domains (the interior of buildings, and outdoor settings with buildings and/or paths present) provide the information necessary for localization. The information gleaned from LTM is used to guide the pilot in the selection and parameterization of appropriate motor schemas and their associated perceptual schemas for instantiation in the motor schema manager. Actual piloting (sensor-driven navigation) is conducted by the motor schema manager. Positional updating occurs concurrently with the actual path traversal. Positional uncertainty is managed through the use of a spatial uncertainty map and related uncertainty transform processes.

AuRA is a system in development and thus is not yet complete. Most of the components concerning navigation and uncertainty management are already in place. It is anticipated that AuRA will undergo evolutionary changes as new components are added (e.g. a more complete mission planner). The schema shell implementation of the motor schema manager exists on an isolated Texas Instrument's Explorer and awaits the shell's migration to the Sequent multiprocessor before it is added to the rest of the system. In the meantime, the schemas are evaluated sequentially in the experimental motor schema testbed (Chapter 8). Indeed, integration issues are a major concern for a system consisting of as many processes as does AuRA, and much remains to be resolved.

AuRA approaches the problem of robotics in a manner different than previous efforts. By drawing on cybernetic models, such as schema theory and the action-perception cycle, significant progress is made towards the development of intelligent systems. Action-oriented perception restricts the amount of computation to tractable levels as dictated by the robot's task of the moment. High level knowledge guides selection of primitive motor behaviors in a new approach to the problem of navigation. Schema theory serves as a basis for the design of the AuRA architecture. Motor, perceptual, and signal schemas are the mechanisms for the connection of perception to action in the quest for intelligent robotic behavior. AuRA itself provides the integrated framework for these components.

AuRA's approach to navigation itself is perhaps most significant, with the robot no longer heavily relying on positional sensors to guide its path execution. Motor behaviors, instead of specific motor commands, accomplish the task of navigation. In a domain as open as a "cosmopolitan" robot's, this behavioral approach is of crucial importance. The

robot is afforded the freedom to react in a reflexive manner to its environment instead of going through a sequence of preprogrammed steps. This flexibility, in our estimation, is absolutely essential for successful navigation in a changing world.

CHAPTER IV

NAVIGATIONAL PATH PLANNING

Obtaining intelligent travel has long been a concern for AI and robotics researchers. Many different issues are involved in the production of such travel. These include spatial reasoning, heuristic search, motor control, representation of uncertainty and environmental sensing of various types, particularly vision.

This chapter is concerned primarily with path construction and navigation in a partially modeled environment. The Autonomous Robot Architecture (AuRA) incorporates a hierarchical planner consisting of a pilot, navigator, mission planner and motor schema manager (the execution arm of the pilot). This chapter addresses specifically the role and operation of the navigator and its associated world model representations upon which the navigator bases its decisions.

In the introduction, relevant work will be cited followed by a description of the UMASS environment. Section 2 will describe the representation used by the navigator and the cartographic software that builds this map. The operation of the navigator will be described in Section 3. The extension of the representation to include diverse terrain types will be related in Section 4. A summary and conclusions will complete this chapter.

§1. Introduction

Early in the days of artificial intelligence, Amarel showed that a good representation is essential for the solution of a problem [1]. AuRA's representation was chosen based on an analysis of the existing representational strategies used in this domain (Chap. 2). A hybrid vertex-graph free-space representation was chosen for global path planning.

The UMASS VISIONS [53,102,140] system encompasses a multi-level representation scheme (Fig. 11). Previously, VISIONS has not maintained representations specifically

addressing navigational path planning. While provision is made for 3D representations of objects and their 2D projections in the vertical plane, no express representation of horizontal projections to the ground plane has been available. AuRA extends and complements the VISIONS system by adding representation levels that specifically deal with the issues involved in navigational path planning.

Navigation path planners come in many forms, simple [130], hierarchical [63,66,90,100, 108] and distributed [51,120]. Unfortunately, in developing hierarchical planners, such as the one used here (Fig. 4), no clear guidelines demarcate what should be delegated to each of the different components (pilot, navigator, etc.). In many cases functionality in one component for one given system is significantly greater than in another (e.g. Nitao and Parodi's reflexive pilot [98,103] incorporates much of what would be considered a navigator in other systems). The navigator in AuRA serves a role analogous to the navigator in a road rally: to provide a piecewise linear path to the vehicle pilot (driver) for execution. Instructions might be: proceed 1.2 km on this road then turn right 90 degrees at the traffic light. The navigator operates from a relatively static map and is not concerned with unrepresented obstacles unless the pilot expressly requests an alternate route.

The pilot is considerably more short-sighted. It is concerned only with satisfying one subgoal from the navigator at a time (although future subgoals may affect its decisions). The pilot additionally accepts constraints from the navigator such as criteria for failure to attain a subgoal. If any of those criteria are met, the navigator is informed and navigational replanning initiated. Local alterations in the route can be made without reinvocation of the navigator as long as these alterations fall within acceptable limits. The pilot need not utilize the same representation the navigator does, as it assumes that the navigator has correctly produced a path that avoids any modeled obstacle. Consequently the pilot is concerned solely with avoiding unmodeled obstacles (subject to certain constraints). Other work [63,66,98] describes the use of similar hierarchical planning systems.

§1.1 *UMASS environment*

The two arenas in which the robot is operated include both indoor and outdoor environs. The first is within the confines of our building, the Lederle Graduate Research

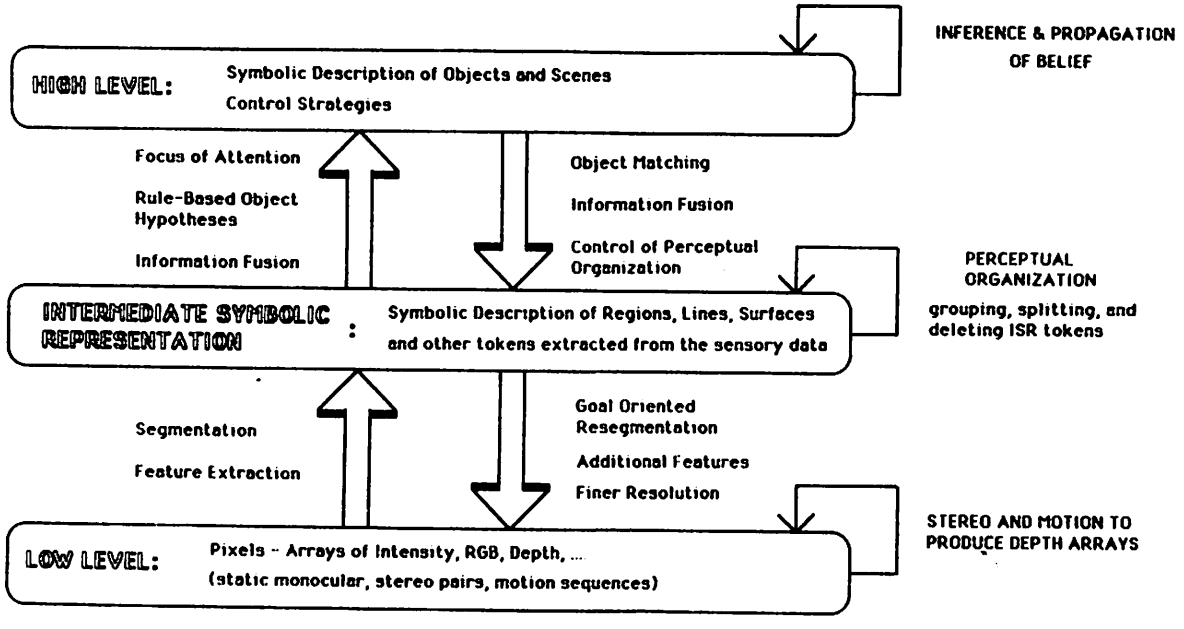


Figure 11: Representations and Processing in VISIONS

Representations ranging from schemas to low-level visual data are organized in a multi-level system. The reader is referred to [55] for additional information. (Reprinted from [55])

Center (GRC). The navigator assumes significant but incomplete *a priori* knowledge of the world. Blueprints for the building (Fig. 12) constitute the basis on which the initial indoor representation is built. A digitizer is used to enter the relevant map features.

The second arena is the grounds surrounding the GRC. This model is derived largely from a map made via aerial photography (Fig. 13). Figure 14 shows two photographs of the same area. Multiple terrain types are present including concrete sidewalks, grassy regions and a gravel path, all of which are available for navigation by the robot.

It should be noted that although the ground plane assumption is made (i.e. the free space is flat) as a simplification for these early phases of the research, there is nothing inherent in the representation that precludes the use of surface models (e.g. planar patches) to represent topographic features within the free space regions.

§2. Representation

To address the issues of path planning, a static representation and a dynamic representation have been developed (the reader is referred to [140] for visual interpretation representations). The static representation, or long-term memory (LTM), is where all *a priori* knowledge is embedded. It is static only in the sense that learning has not yet been incorporated into the system. Although a variety of sensor interpretation strategies also access data stored in LTM, the navigator is the prime consumer of this representation within the hierarchical planner.

The dynamic representation, short-term memory (STM), is a layered representation consisting of the robot's current perception of the world based upon a long-term memory context. Of the planner components, the pilot (and motor schema manager, which is the execution arm of the pilot) is the principal user of the data stored here. Portions of LTM are instantiated in STM based upon the robot's current position and the navigator's instructions. As the robot traverses this path, sensor data (visual and ultrasonic) are incorporated by the cartographer to build up a dynamic model of the perceived world. This is then used to direct the pilot to appropriate action when the path is blocked or a short-cut makes itself apparent. Additionally, vehicle localization (increasing positional certainty) can be guided by available landmarks found in these regions of visibility. A discussion of the details of short-term memory for navigation appears in Section 5.

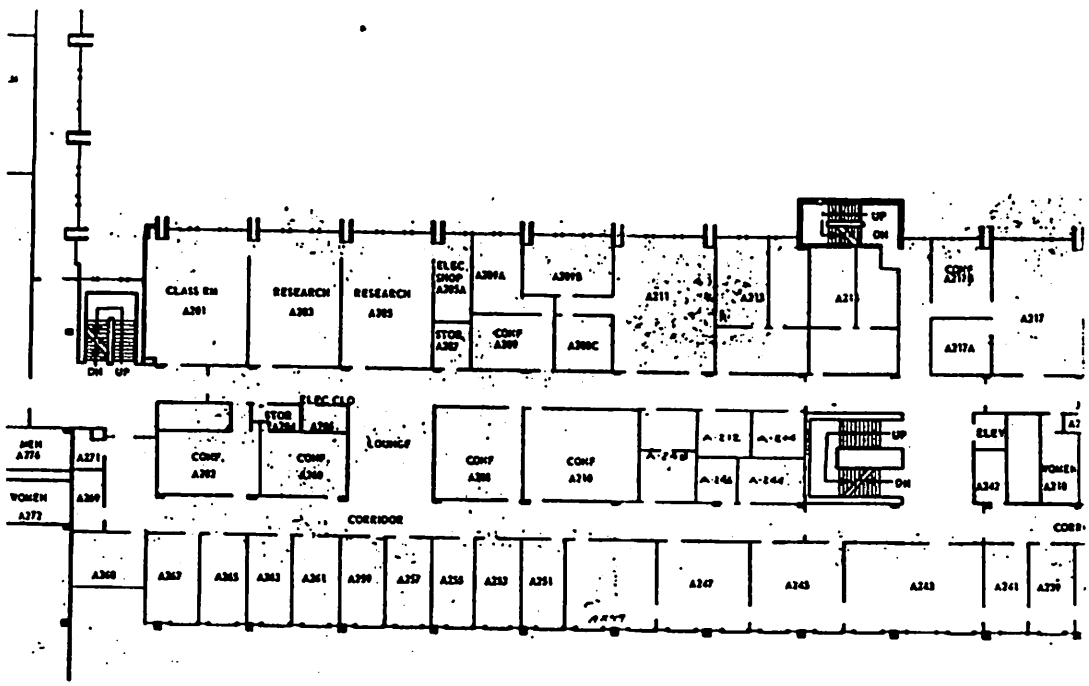


Figure 12: Indoor environmental model

This diagram, a partial blueprint for the GRC, serves as the basis for building the long-term memory model of the robot's world (indoor case).

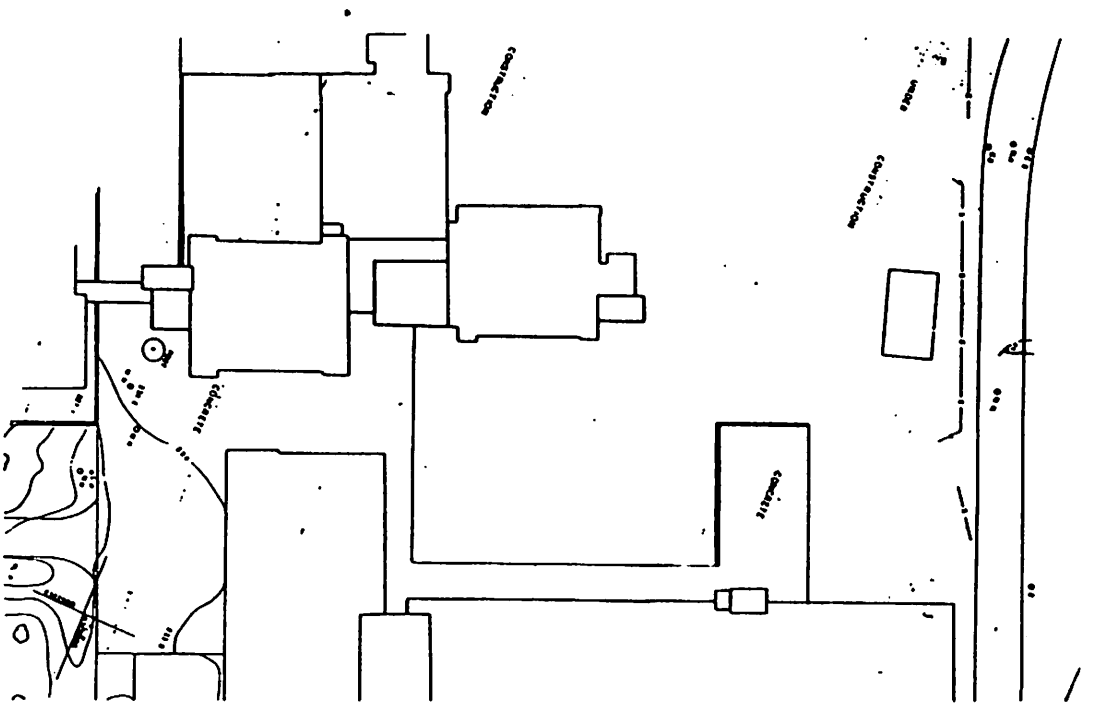


Figure 13: Outdoor environmental model

For the outdoor scenario, the long-term memory representation is built starting with an aerial map of the environment surrounding the GRC.

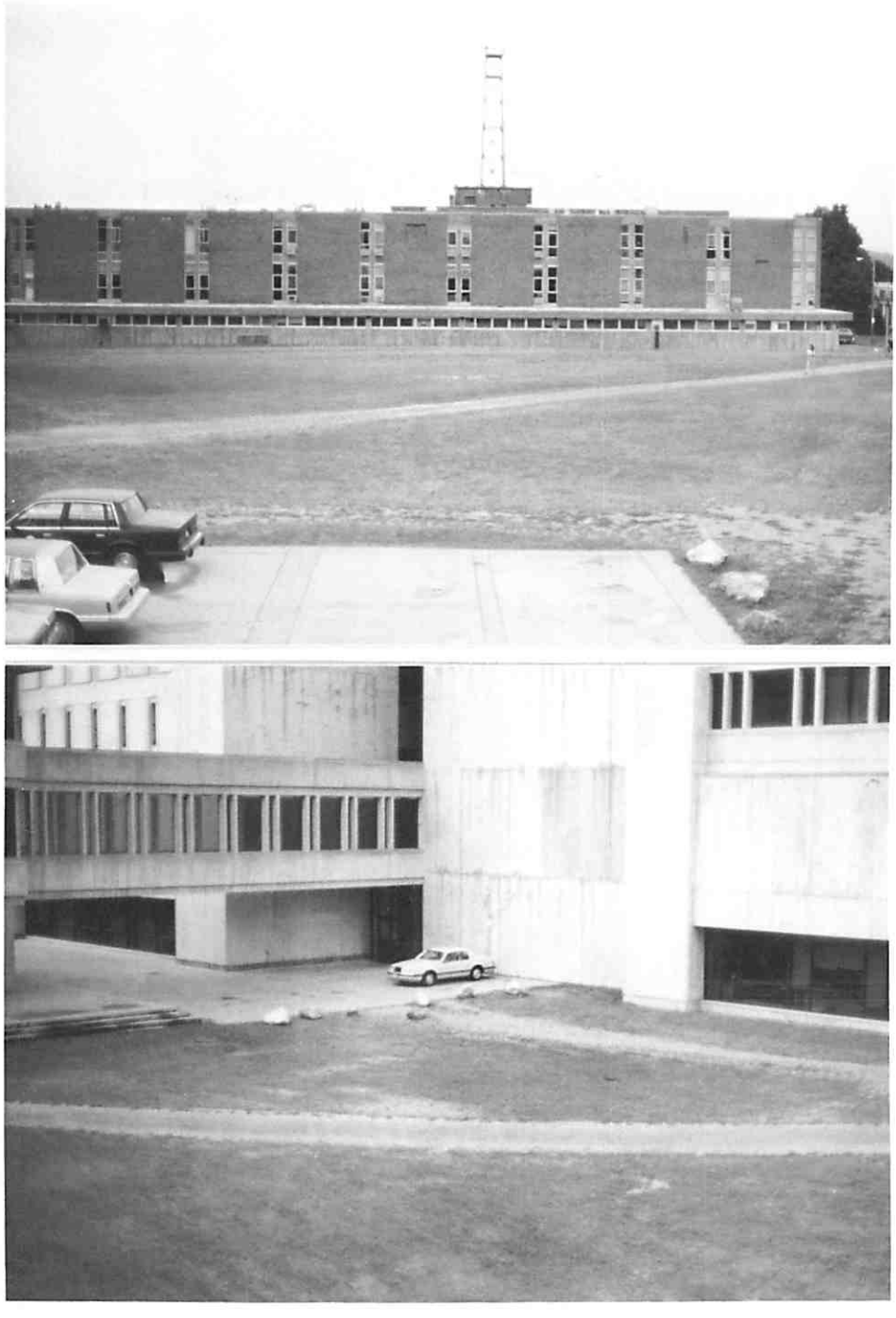


Figure 14: Outdoor photos

Two outdoor photos of the area depicted in Fig. 13.

§2.1 *Long Term Memory - Meadow Map*

The principal representation used by the navigator is a "meadow map" representation (a hybrid vertex-graph free-space model) based on previous work by Crowley [36] and Chatila and Laumond [33]. It models free space as a collection of convex polygons. Diagrams depicting an indoor scene and an outdoor scene appear in Figures 12 and 13 respectively. The rationale for using convex regions is that a line between any point within one convex region to any other point within that same region is guaranteed to be free of collisions with all known obstacles. Thus, the global path planning problem simplifies to finding an appropriate sequence of convex region traversals. (Actually finding a "good" path is more difficult - see the path improvement strategies described in Sections 3.2 and 4.2).

What distinguishes this representation from the efforts that preceded it is the ability to embed both terrain characteristics and data for establishing sensor expectations and its extension to allow navigation over diverse terrain types (see Section 4). Convex regions were chosen over a regular grid approach due to their ability to avoid digitization bias, a smaller search space, and a significant reduction in memory requirements. Voronoi diagrams (a set of polygonal regions, each representing an enclosed area in which all contained points are closer to one particular point in a given point set than to any other point in the set) were avoided due to their inability to relate landmark and terrain data readily and their perceived limitations on flexibility of path construction when compared to the strategy used in AuRA.

In this section, the map building algorithm used to construct the basic representation (not multi-terrain) is described. This is followed by a brief presentation on the role of the feature editor and its importance for guiding environmental sensing.

Meadow Map Construction

The algorithm for the construction of the LTM meadow map is described in Figure 15 and consists of the following phases: initialization, main map building and clean-up.

Initialization

In the initialization phase, a series of vertices in global coordinates describing the maximum reaches of robot navigation are accepted. In the case of the interior of a

FREE SPACE MAP BUILDING ALGORITHM

Initialization

Accept and shrink bounding region (*a lá*) configuration space)
Accept and grow modeled obstacles (configuration space)
Merge collided grown obstacles and border together
Attach obstacles to border (1 region results)

Main map building algorithm

IF region is convex (no concave angles present)
    done
ELSE
    Find (most,least,first) concave angle
    Connect it to (most opposite,leftmost,rightmost) clear vertex
    Apply main map-building algorithm recursively
        to the two resulting regions
ENDIF

Clean-up

Merge any regions together that will yield a convex region
Output list of connected convex regions

Figure 15: Free space map-building algorithm

building, this would be the bounding walls. In more open terrain, it might be boundaries of limiting paths or an imaginary polygon bounding the traversable region. There are no restrictions on the shape of the bounding region (and obstacles) other than that they be represented by a series of straight line segments. Curving surfaces must be converted to piecewise-linear segment approximations. The raw data is obtained from a map or blueprint of the region and the use of a bitpad digitizer.

After the actual coordinates of the bounding region are accepted, the region is shrunk by a distance equal to the radius of the robot plus a safety margin (Fig. 16) in the configuration space (C-space) manner as developed by Lozano-Perez [76]. This enables the robot to be treated as a point thereafter for path-planning purposes. Ties are maintained via pointers from the newly created C-space vertices to the original bounding vertex.

During the shrinking (and obstacle growing process described below) a deviation from standard C-space techniques was required in the case of concave vertices. Normally the circular robot would produce a curved C-space for a concave angle (Fig. 17a). Two alternatives are available to produce the required linear segments. First the resulting side line segments could be extended until they meet (Fig. 17b). This could result in significant and unnecessary loss of free space for very sharp angles, even resulting in the blockage of free space corridors (Fig. 17c). The second alternative is extending the line segments and then chopping the resulting C-space region, when a line normal to the bisector of the grown angle is intersected (at a distance from the vertex equal to the robot diameter) with the segments produced in the first method (Fig. 17c-d). Although this approach still wastes some free space, it is considerably better than the first case. The principal drawback is the formation of two grown vertices from the original ungrown one, which results ultimately in more regions being formed and thus more processing time during the later path planning phase. The decision at what degree of convexity to switch from straight extension to chopping mode is controlled by setting a program parameter. Highly cluttered areas would favor chopping for most of the concave angles to prevent passage occlusion, whereas relatively clear areas would prefer the straightforward extension method.

Known obstacles that are present in the environment (pillars, telephone poles, etc., – any static impediment to motion) are then added. These also are digitized in the manner above. These obstacles are then grown in the C-space style for the same reason that the

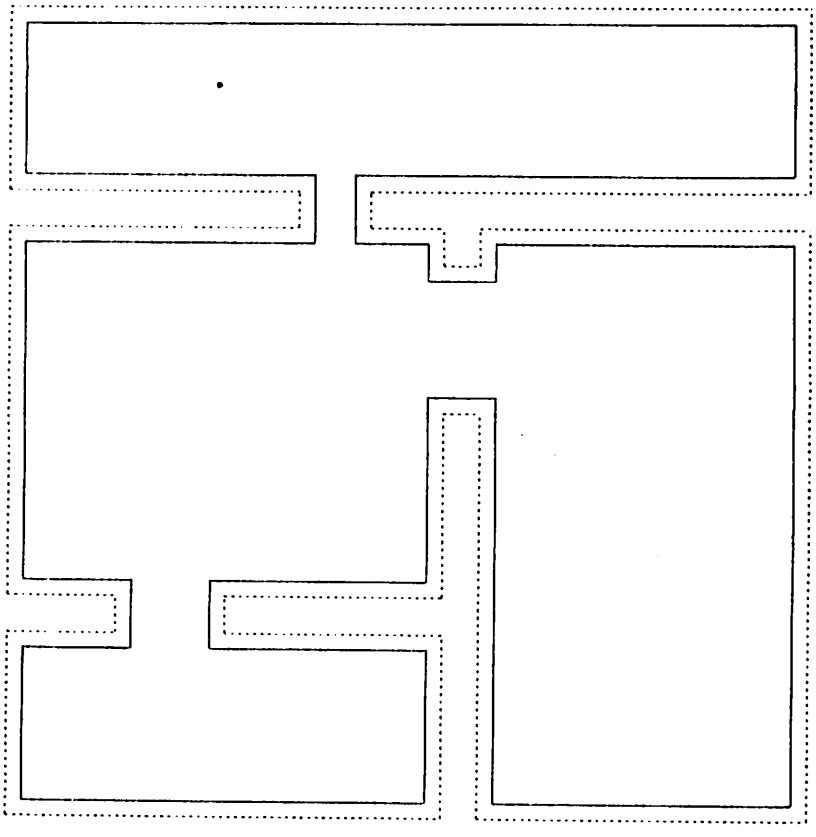


Figure 16: Grown border

A test indoor case, based on an example from [51], shows the original ungrown border, (dotted line), and the resulting shrunken region (solid line). (The shrinking is exaggerated in this figure for the sake of clarity)

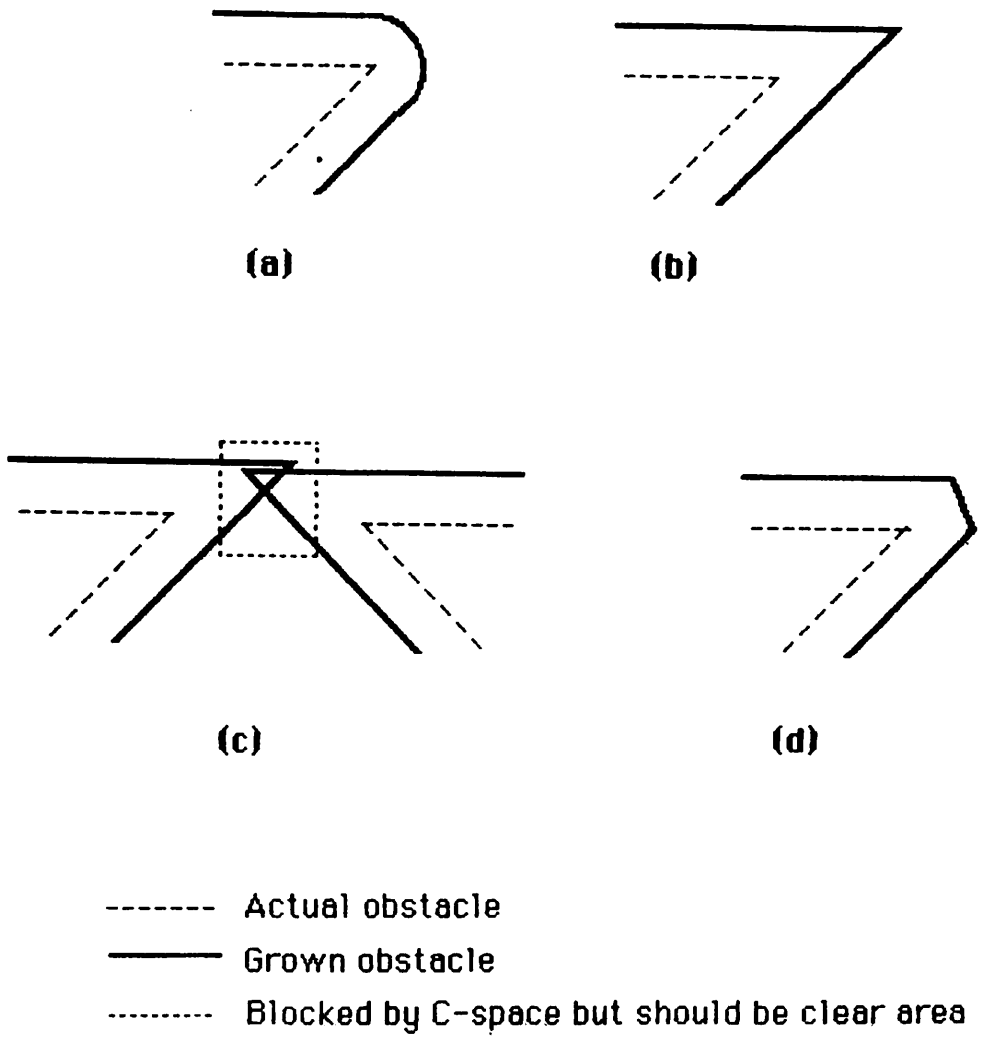


Figure 17: Chopping concave angles

(a) Traditional configuration space growing for a circular robot.

(b) Simple intersection method results in a single vertex.

(c) Simple method can occlude passages that should be clear.

(d) Chopping the angle reduces waste but results in 2 vertices.

bounding region was shrunk (Fig. 18). Any obstacles whose growth results in a collision with the bounding region, are merged into the border as they no longer can be completely circumnavigated.

Finally, the remaining obstacles are attached to the bounding region as follows. The obstacle vertex that is closest to a bounding obstacle is attached to the bounding region by two passable links; one going out to the obstacle and the other returning. This is repeated until all the obstacles are connected. In essence, a single region, consisting of the border and the perimeters of all the grown obstacles, is produced (Fig. 19). To the mathematical purist, this would not be a region as the two passable lines connecting the obstacle to the border are in identical positions. Nonetheless, this simplifies the recursive decomposition algorithm which appears in the step following.

Main Map Building Algorithm

This portion of the algorithm decomposes the region produced in the initialization phase by recursively splitting the area until all resulting regions are convex. Upon receipt of the initial region it is checked for convexity. If the region is convex, this portion of the procedure terminates. If it isn't, which usually is the case, a concave angle is selected from those available in the region. There is guaranteed to be at least one concave angle or the region would be convex. Three options were considered for selection of the vertex: the least concave, the most concave, or the first concave angle found can be chosen (Fig. 20). Intuition as well as empirical results indicate that choosing the most concave angle results in the fewest regions to be remerged during the cleanup phase below. Choosing the first concave angle would be more computationally efficient during this phase, but may require additional compute time in the clean-up phase, offsetting any gains here.

After an appropriate concave vertex is selected, the second (victim) vertex for splitting the region in two must be chosen. Again we have defined three choices (Fig. 21): the leftmost clear vertex, the rightmost clear vertex, or the most nearly opposite vertex (right of center). A connecting edge, labeled as passable, is completed between the concave angle and its victim and the initial region is split in two. The algorithm is then applied recursively to each of the resulting two newly formed regions. Pointers within the newly produced edges are maintained indicating the adjacent passable region (Fig. 21a). Thus a graph of convex regions and their traversability is produced, facilitating search during

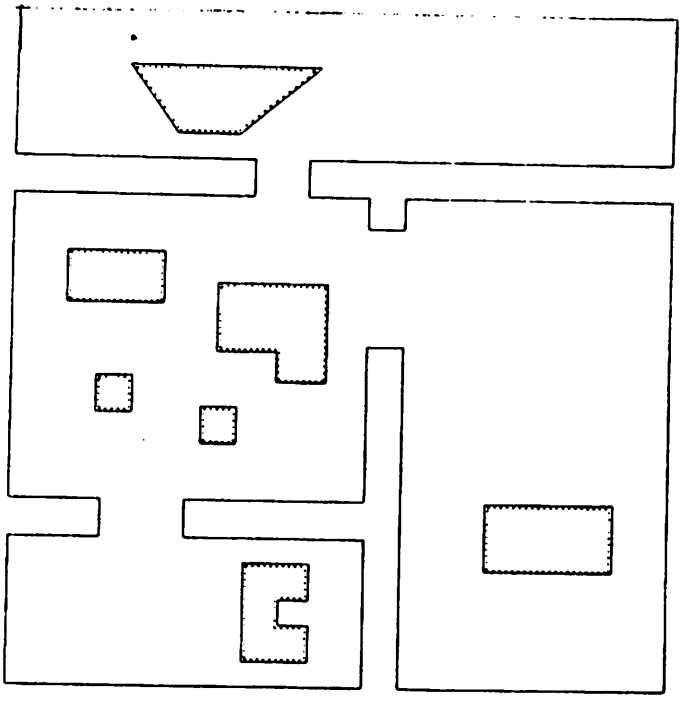


Figure 18: Obstacles

The obstacles have been added to the case shown in 16. The dotted line represents their original position and the solid line their grown area. Only the shrunken border region is shown (solid line), not the original border.

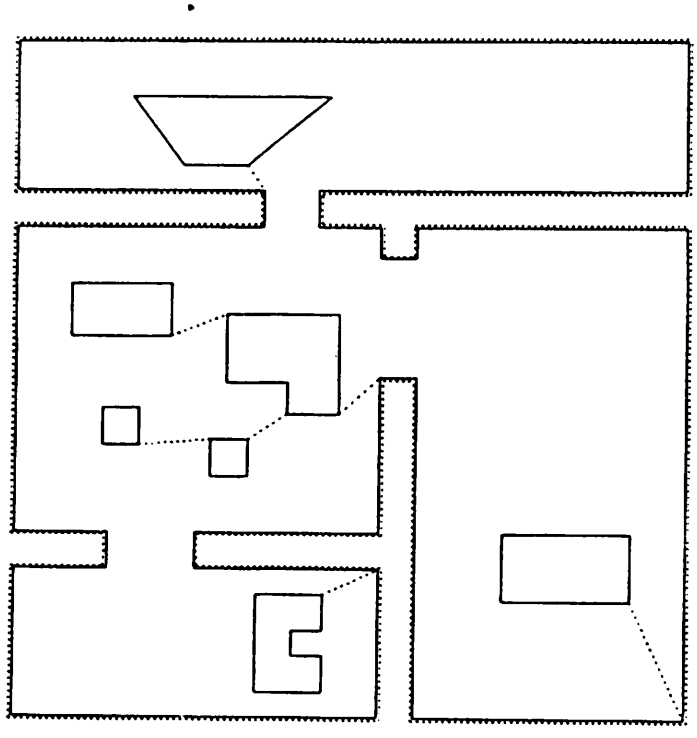


Figure 19: Attached obstacles

The grown obstacles have been attached to the surrounding shrunken border. The resulting single region is now ready to undergo convex decomposition.

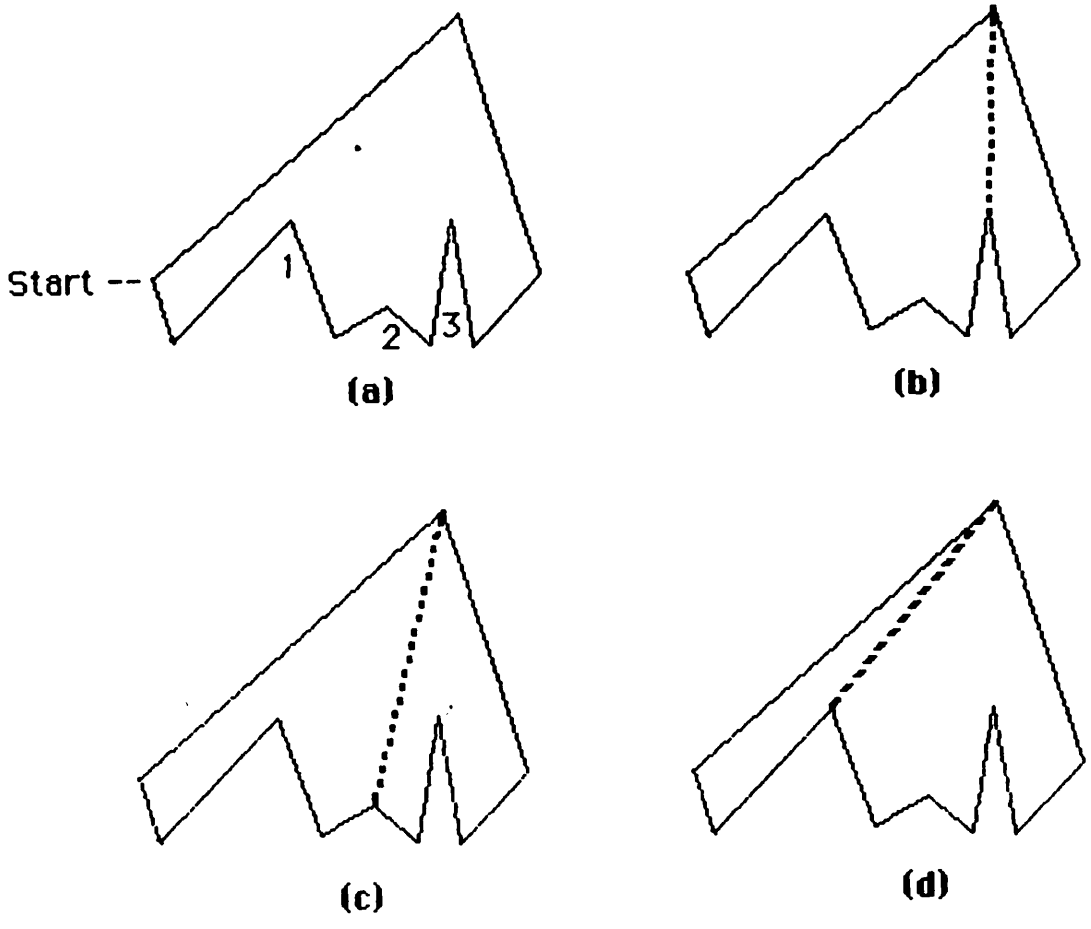


Figure 20: Effect of vertex selection on decomposition

a) A region to be decomposed that contains 3 concave vertices (numbered 1,2,3). The list representing the region begins at start and proceeds counter-clockwise. Figures b-d show how the vertex can affect the decomposition. For each case below, the most opposite vertex is selected as the victim.

b) Most concave vertex selected.

c) Least concave vertex selected.

d) First concave vertex selected.

the path finding process. This decomposition continues until all of the regions produced by this algorithm are convex.

The efficiency of each mode and their impact on the path planning computation, and data regarding map-building times appears in Appendix A. Considerable experimentation was carried out, trying to determine which of the concave selection modes and victim selection modes (of the 9 possibilities) yields the "best" results. Figure 22 shows 5 different decompositions on the same region. Just how to define what constitutes the best result is nebulous. Shortest Euclidean distance as a path length metric (which might appear to be the most obvious choice) may result in significant problems with the clipping of modeled obstacles during travel due to the inherent positional uncertainty found in the mobile robotics domain. Fewest overall legs in a particular path is another possible choice. In one instance [36] an algorithm producing the maximally large convex region is used. This might actually work against the path optimization strategies described below, although conceivably improving overall search time for the coarse "raw" path.

When the path search was restricted to the midpoints of the bounding regions, (A*-1, see Section 3.1), the experimentation indicated, even based on the shortest distance metric, that the results obtained were more strongly influenced by the shape of the initial bounding region and the choice of start and goal points of a particular path than by any predetermined choice of vertex selection modes for decomposition. That is not to say the choice of decomposition method did not produce significantly different paths in certain circumstances for the midpoint search; rather, information that is dependent on a particular initial region and the most likely paths to be taken within that region should appropriately influence the vertex selection process. An expanded search (A*-3) through 3 points on each passable meadow boundary (the midpoint and one point near each endpoint) largely decouples the dependency of the path cost on the decomposition method. Consequently, it becomes less significant which map-building strategy is chosen if this more costly search methodology is used.

When not guided by other factors, this author would choose the most concave angle and most nearly opposite angle as selection modes, as it generally results in the fewest merges in the clean-up phase while yielding a more aesthetically pleasing result (aesthetics are not forwarded as a metric however). It also is intuitively appealing as a "natural" decomposition strategy.

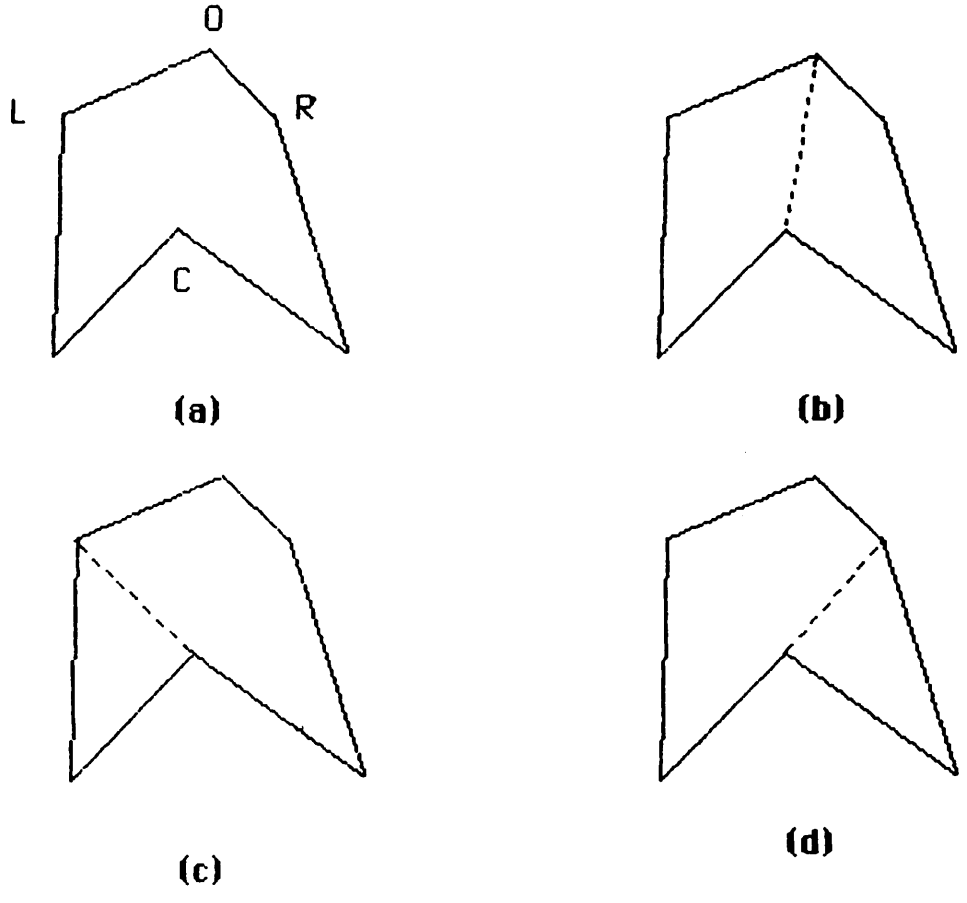


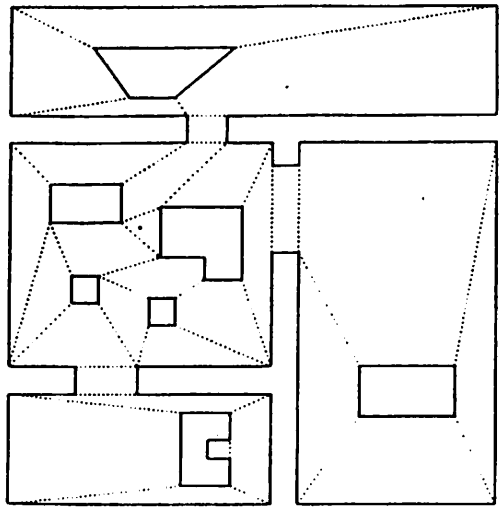
Figure 21: Effect of victim selection on decomposition

a) This figure contains one concave vertex (C) with three possible victims. O is the most opposite victim, L is the leftmost and R is the rightmost.

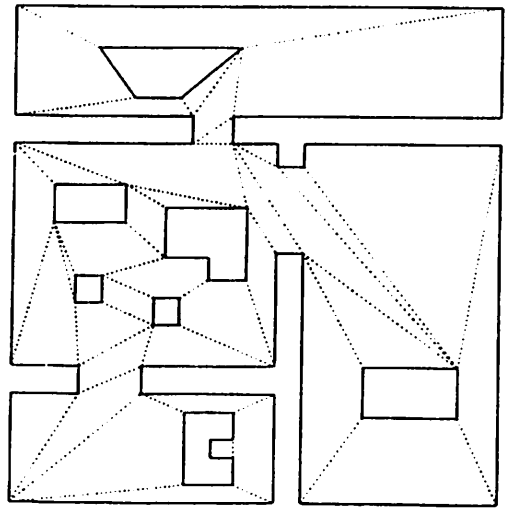
b) The decomposition resulting from the most opposite victim selection mode.

c) The decomposition resulting from the leftmost victim selection mode.

d) The decomposition resulting from the rightmost victim selection mode.



(a)



(b)

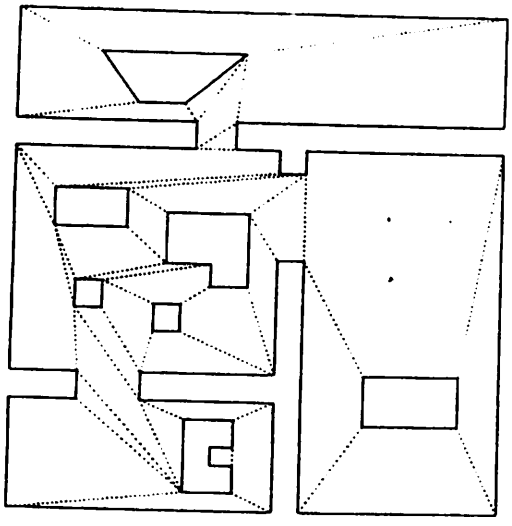
Figure 22: Different convex decompositions

Figure 22 shows 5 of the nine different decompositions available for the region shown in Fig. 19. Solid lines represent impassable obstacles and borders, dotted lines - passable boundaries between meadows.

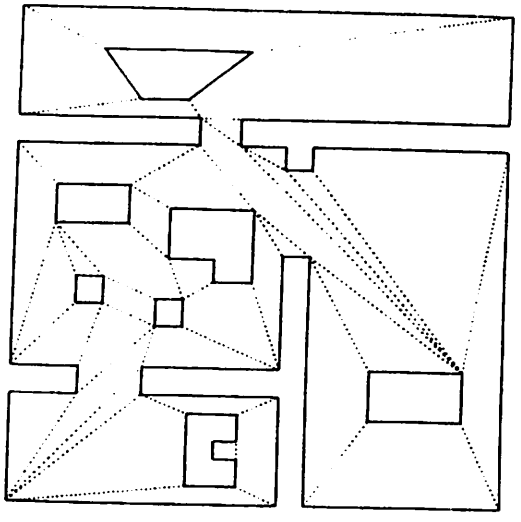
a) Selection modes: most concave angle, most opposite victim.

b) Selection modes: most concave angle, leftmost victim.

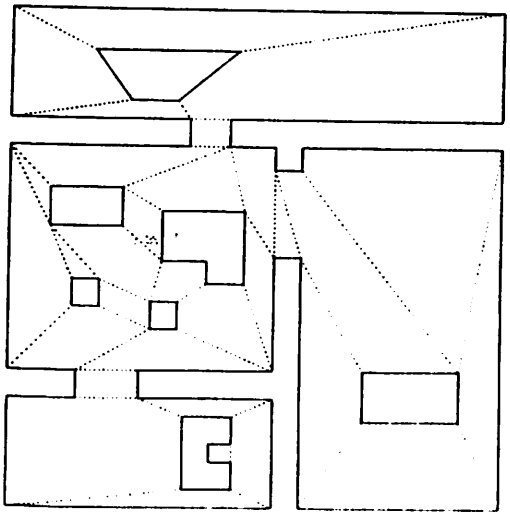
(Figure continued on following page).



(c)



(d)



(e)

Figure 22 continued.

c) Selection modes: most concave angle, rightmost victim.

d) Selection modes: least concave angle, most opposite victim.

e) Selection modes: first concave angle, most opposite victim.

One other note: although the algorithm is recursive, for efficiency in implementation, instead of using the system stack and system-provided activation frames found with standard recursive calls, a push down stack was managed by the map-builder process itself, avoiding the significant system overhead that would be required for the decomposition of large and complex free space areas. The net result in any case is a list of the resulting convex regions along with an embedded connectivity graph maintained by pointer links in the passable edges connecting a region to its adjacent passable regions.

Clean-up

The resulting number of convex regions produced by the main decomposition algorithm is not always minimal. In other words there may be some regions which can be merged together that still result in a convex region. This is a consequence of the local nature of the decomposition technique; no checks are run to determine the global consequences of a region splitting. Although this could be built into the algorithm, the backtracking that would be required is believed to be considerably more expensive than the simple merging step. In some instances (e.g. most concave - most opposite vertex selection modes) merging is relatively rare, while in others it is relatively common. During this phase, a pass is made on the convex region list that merges together any regions that would result in a single convex region. On merge completion, the map-builder process then invokes the feature editor.

It should be recalled that this is the algorithm for the simplest case, involving only one terrain type. Multiple terrain types require additional processing which will be described in Section 4.

§2.2 *Feature editor*

The feature editor was created to allow data pertinent to sensor guided navigation and uncertainty management to be added to the meadow map. It serves as an interface to assist in the knowledge acquisition and representation processes.

A major advantage of the meadow map representation is the ease with which representations of objects, landmarks, terrain features, etc., can be expanded. In an experimental system this is very important. The level of granularity at the point of attachment can vary. Obstacles or walls can be represented with full 3D models of the entire object;

2D planar representations of surfaces (pop-up views) can be associated with meadow sides (edges); or simplistic line models for individual corners, projected up into the image plane, can be attached to meadow vertices. For terrain, entire region characteristics (traversability data, statistical error data, data for guiding visual region segmentation algorithms, etc.) can be tied to the free space regions. Individual meadows can have topographical models (for non-planar surfaces) which represent contours in any way the designer of such a representation chooses. This flexibility for adding and modifying world representations is one of the prime factors in the choice of the meadow map scheme over other alternatives such as the regular grid or Voronoi diagram.

The mechanism for adding these representations is through the use of the feature editor. The concept is simple: a particular free space region, obstacle, obstacle edge, or vertex is chosen through the editor; data for the new representation is accepted by the editor; storage is allocated for it and a link is made between the new representation and the old. This is repeated until no more data is to be added. The data within the allocated representations can also be modified interactively if required.

The data stored in the representations can be acquired through sensors as well. For example in the case of visual data for region segmentation, by pointing the robot camera in the direction of a known region type (e.g. grass), and then acquiring the appropriate statistics through interactive use of the video digitizer and a histogram process, the robot can store the statistical features for a particular terrain type on a per run basis. This avoids the inflexibility that would be present if the statistics had to computed once for all weather and seasonal conditions. The result is more robust visual segmentation. In essence the robot can be trained quickly and efficiently to recognize certain terrain features. It would not be difficult to extend this to include data for landmark recognition and other necessities once appropriate representational strategies were chosen. Static representations can be input from data files, reducing the map-building time. It should be recognized, however, that changing lighting conditions and other environmental factors can render these statistics obsolete. Consequently, adaptive (feedforward) mechanisms are required by the perceptual processes to make the system more reliable (Chap. 6).

Some of the initial features included in the system are:

- Terrain data
  - Traversability factor (ease of passage)
  - Terrain-specific translational and rotational error data - (to guide the uncertainty map manager in handling positional uncertainty)
  - Data to guide terrain region labeling using visual region segmentation
  - Unmodeled obstacle density
- Obstacle data
  - 3D models of landmarks
  - Partial wire frame models of buildings
  - Vertical edge data for particular vertices (sides of buildings, doorways, etc.)

Currently this data is user-supplied. In the last stage of the map-builder, after the user exits the feature editor, the pointers for long-term memory are installed making LTM available to other processes. The map-builder process then terminates.

§3. Navigation (Global Path Planning)

After the map-builder process terminates, the planner process is initiated. The planner is hierarchical in design; consisting of a mission planner, navigator and pilot. The mission planner is delegated the responsibility for interpreting high level commands, determining the nature of the mission, setting criteria for mission, navigator and pilot failure, and setting appropriate navigator and pilot parameters. For example, if the mission is reconnaissance oriented, (e.g. searching for lost keys), the pilot mode of operation would be set to path seeking. On the other hand if it was target oriented, (e.g. delivering a pizza), the mode would be set to goal seeking. The mission planner, although part of the overall design, is not yet fully implemented, and has a relatively low priority. Section 6.1 describes the rudimentary mission planner used in the first-pass implementation of AuRA.

The navigator accepts a start and a goal point from the mission planner and, using the global map built by the map-builder, determines the "best" path to attain that goal. The

definition of optimality is determined by the mission planner. It might be the shortest, or the safest, or the fastest, or the least energy consuming path. In essence, the mission planner determines which cost functions and heuristics that the navigator will use in carrying out its role.

The remainder of this section deals with the search strategies used by the navigator, the path improvement strategies that convert a coarse, raw path into a refined one, and a presentation of results. The modifications necessary for multi-terrain navigation are presented in the section following.

§3.1 *Search*

The navigator's task is to search through the meadow map produced by the map-builder and derive a good path available for a specified start and goal. As stated earlier, "best" is difficult to define. Many different criteria can be used to affect the quality of a path. Parodi [104] used a weighted cost function and dynamic programming techniques to search through the solution space of a regular grid and arrive at the best path.

The A* search algorithm [56,96] is used in AuRA with heuristics that guarantee optimality. The A* cost function is defined as:

$$f = g + \hat{h}$$

where:

g = the measured cost of the path up the current point

$\hat{h}$ = the heuristic cost from the current point to the goal

(to be admissible, $\hat{h}$ must not overestimate the actual cost to the goal).

Two different search spaces are available for the search algorithm. The simplest and most efficient, A*-1, is built from the midpoints of the bordering passable regions (a concept derived from Crowley's adits [36]). The larger space, A*-3, is derived from a triad of points on the bordering regions; the midpoint and two points near each end of the passable edge (separated from the end by a specified safety margin). Although computationally more expensive (the space is larger), the advantage of A*-3 over A*-1 lies in a significant decoupling of the path planning from the map-building free space decomposition method (due to the expanded search space). The A*-3 method explores more alternatives, possibly resulting in a lower cost path than would be available with

the A*-1 search. Additionally, since it tests points in close proximity to obstacle vertices, the location of the boundaries of the adjacent meadows themselves become less important (especially for short paths). In either case (A*-1 or A*-3), the search space is smaller (and consequently faster) than that of a regular grid or pure vertex graph representation. Finding the initial coarse path is a fairly rapid operation (see Appendix A). It is guaranteed to be the best path available (subject to the cost function chosen) within the specified search space. This space, however, is not strictly analogous to the physical world.

The choice of A*-1 or A*-3 is made by the mission planner, differentiating between the two on the basis of whether it is more important to compute a path rapidly (A*-1: faster) or more important to traverse the path rapidly (A*-3: can yield a lower cost path). In many cases the paths resulting from both A*-1 and A*-3 are identical.

In order to ensure admissibility, the heuristic function $\hat{h}$ of A* must never overestimate the cost remaining to the goal. The easiest $\hat{h}$ heuristic function to guarantee an underestimate (or the exact cost) is the straight-line Euclidean distance on the plane from the current position to the goal assuming the best terrain. Since the search space is relatively small, no effort has gone into finding better heuristics. The computation time required to produce the path (in A*-1) is somewhat dwarfed by the time required to convert this initial raw path into a refined and reasonable path (see Appendix A).

The cost function used takes into account the traversability factor of a given terrain type, the actual distance traversed, and can readily incorporate other factors such as threat measurement, topographical grades, etc. This cost function is used in the g component of the A* algorithm. Other factors might include unmodeled obstacle density (perhaps a function of time of day - e.g. high obstacle density between classes on a sidewalk, low otherwise), and ease of localization (based on numbers of readily discernible landmarks within a given region).

At this point in our research, it is impossible to say just what constitutes a "best" path. If a path is very short but the robot gets lost due to inadequate landmarks for localization, or it gets mired in poor terrain and its dead reckoning sensors become grossly misleading, little has been achieved. The only effective metric is the robot's ability, under the pilot's control, to successfully complete the path. If it can, only then can we take into account additional yardsticks such as time, distance, etc.

Consequently, the ultimate goal of the navigator is to arrive at a "reasonable" path rather than a claimed "best" path. By reasonable we mean a path that appears plausible from a human's perspective - i.e. it is conceivable that a person would take a similar path. In any case even if an optimal path (by whatever definition) was attainable by the navigator it could only be based on partial information (i.e. the modeled world). Since the robot's environment is subject to unmodeled and even moving obstacles, there is no *a priori* guarantee that any path produced by any navigator is the best path, given only incomplete world knowledge (although the path is optimized relative to the current world model). Reasonableness seems an acceptable criteria.

In a dynamically changing world, which can quickly invalidate preformulated plans, how do we gauge "reasonableness"? Only successful completion of the robot's experiments can be the judge. This is especially the case for multi-terrain navigation. When would you, as a human, take a short-cut over the grass in lieu of the sidewalk? Robot's have different locomotion systems so this example is not fully extendible, but the navigator can still be judged in this light. One more point in defense of the premise of reasonableness: do people really choose an "optimal path" when traveling from one point to another? I think not, except under rare circumstances. The time required to compute the path, (referring to a map etc.), might take longer than the completion time of the path itself. Therefore, no claims are made for the optimality of the paths produced by the navigator, only that the resultant paths are reasonable under all observed conditions.

In summary, the navigator algorithm (shown in Fig. 23) accepts two points from the mission planner. It then searches, using the A* algorithm with a cost function based on terrain factors and traversability, the space of midpoints (A*-1) or triads (A*-3) of connecting adjacent passable meadows, outputting a coarse path consisting of a series of piecewise linear segments connecting the start, the edges of bordering meadows and the goal. This approach generally expands fewer nodes than would a comparable pure vertex graph of the obstacle edges (and obviously much fewer than a regular grid) as the number of passable meadow boundaries is less than the number of vertices. The pure vertex graph (although guaranteed to produce the shortest path) also suffers from an inability to readily produce safe paths (paths that minimize the danger of the robot clipping an obstacle), since the free space is not directly represented. A Voronoi diagram can readily produce safe paths, but also lacks the flexibility afforded by this representation

to change its strategies (safe to short to fast) when deemed appropriate by the mission planner. The Voronoi diagram discards information regarding the obstacles themselves too soon, making it necessary to reconstruct that data when needed for alternate path finding strategies.

The key, however, lies in the path improvement techniques described below. Without these techniques the raw path produced in many cases would appear to be haphazard and unreasonable even to the casual observer (especially for A*-1).

§3.2 *Path Improvement Strategy*

Path improvement techniques are relatively common in the use of regular grids. Although the representation used for AuRA's long-term memory is a meadow map and not a regular grid, the precedent of refining a coarse path into a better one exists. Thorpe uses a relaxation based approach on a coarse grid [126] while Mitchell and Keirsey use a compensation technique [88] to minimize inefficiency due to digitization bias.

The algorithm for path improvement for the simple single terrain case is presented in Figure 24. A graphic illustration of the process appears in Fig. 25. The "raw" path is first received from the navigator. Beginning at the start path node and proceeding to the end node, each node on a passable meadow border is tested at three locations; slid all the way to the left (leaving a specified safety margin clearance), slid all the way to the right (minus safety margin), and unchanged at the middle. The lowest cost solution is chosen and the path modified accordingly. This can be visualized as pulling on the ends of the path thus tightening the path around the obstacles and walls. This is considerably less costly than a relaxation algorithm requiring multiple iterations over the entire path. (A limited relaxation algorithm involving only the transition zones is required for the multiple terrain case - see Section 4). The A*-3 search method can bypass this initial tautness processing as its search strategy has already effectively accomplished it.

For eliminating unnecessary turns, if deemed appropriate by the mission planner, a straightening algorithm is utilized. Any unnecessary turns in the path are removed. Beginning with the start path node, all further path nodes are checked against the current path node to see if a path exists that does not intersect with any of the known environmental obstacles. If such a path exists, all intervening path nodes between the two connectable nodes are deleted from the path. This process is repeated for all nodes in

PATH FINDING ALGORITHM

Accept start and goal from mission planner.

Check for validity (located in free space).

Search

Apply A* search algorithm through convex region connectors.

(A* - 1: midpoint only)

(A* - 3: midpoint + two points near endpoints of connectors)

Output Raw Path.

Path Improvement Techniques

If specified

A. Tighten path by sliding towards side vertex
by some given amount.

B. Straighten path by removing any turns that are not essential
for a clear traversal.

(details for both parts A and B appear in Fig. 24)

Return "reasonable" piecewise linear path through world model.

Figure 23: Path finding algorithm

PATH IMPROVEMENT ALGORITHM

(single terrain type)

Accept the coarse path from the search component of the navigator

Tautness part

Accept safety margin (clearance from side)
Get first border midpoint of coarse path
DO WHILE Not at end of path
    compute length of path for three cases
        a. Midpoint unchanged
        b. Midpoint slid to right (maintaining safety margin)
        c. Midpoint slid to left (maintaining safety margin)
    choose lowest cost path from a, b or c.
    modify path if necessary and mark path node as moved
    Get next path node
ENDDO

Straightness Part

Get start of path
DO WHILE not at end
    IF clear path is available to any path node ahead of current node
        delete all intervening nodes
    ENDIF
    Get next path node
ENDDO

Clean up

Slide towards edges again as in tightening part above if path was
straightened (only for unmoved path nodes still at midpoint)
Output refined path

Figure 24: Path Improvement Algorithm

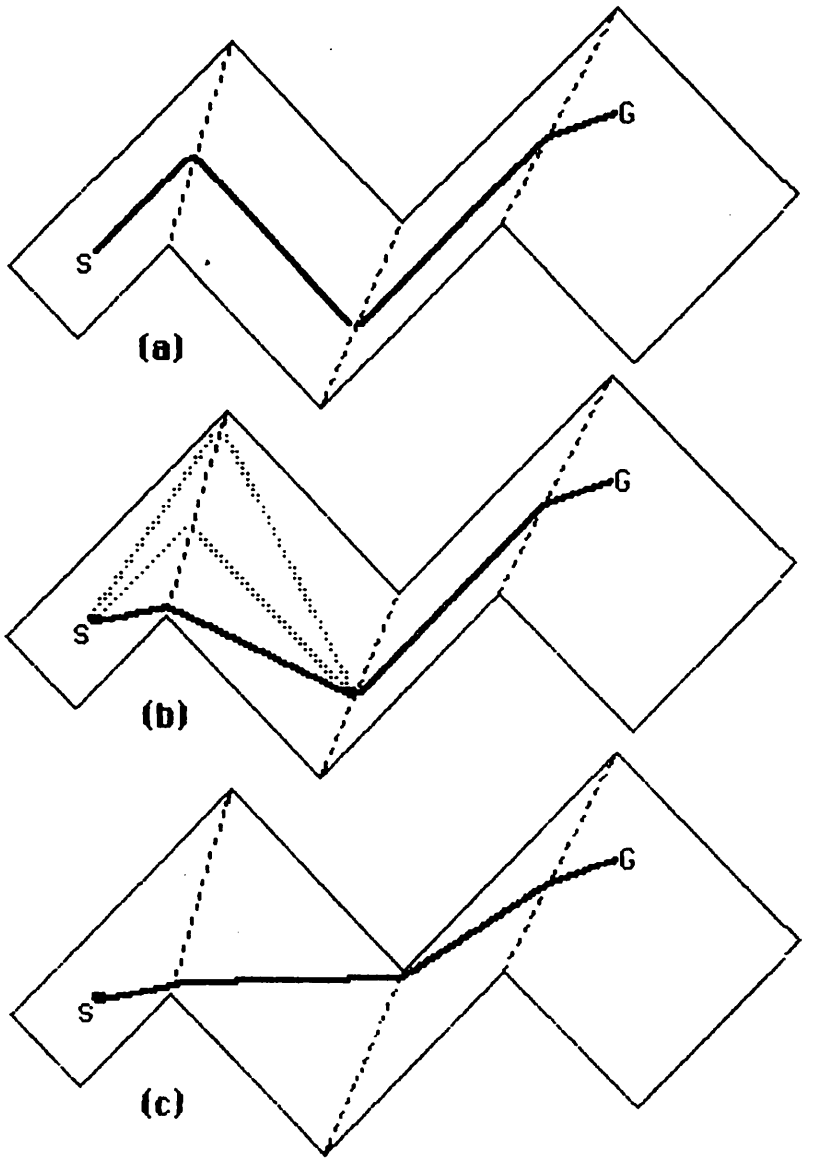


Figure 25: Path improvement - tautness component

a) The initial raw path passed from the A* search strategy to the tautness component. S denotes the start and G the goal.

b) The first passable boundary is tested at three locations, the midpoint and the two endpoints (minus a safety margin). The lower cost result is shown as the dark line.

c) The process is repeated at the other two passable boundaries. This results in a lower cost but as yet unstraightened path. (The kink from the final boundary to the goal will be removed by the straightening component).

the path.

If the path is straightened, a better path may now be obtained by sliding some of the previously unmoved path nodes. Before exiting, the algorithm checks all these unmoved nodes, if there are any, to see if a lower cost path can be obtained by sliding them along their meadow boundaries (basically the same procedure as in the tautness part above but checking only a subset of the remaining path nodes). The resulting refined path is output from the navigator and stored in short-term memory for use by the pilot.

Reasonable paths have been observed in extensive testing of all cases presented. Unnecessary detours around obstacles are removed by the straightening component of the path improvement strategies, while overall cost minimization is ensured by the tightening approach.

§3.3 *Results*

The results are presented in Figures 26-32. The straightening component of the algorithm can be observed to remove unnecessary detours around obstacles, while the tightening component reduces the overall path cost. The cost function used is the same cost function used in the search algorithm (for these figures, Euclidean distance is the cost).

A significant advantage in deferring the path improvement strategy until after the search (rather than being an integral portion of the search algorithm) lies in the ability to alter the path, if necessary, without re-searching. Additionally, embedding the straightening portion would be awkward at best within the search algorithm.

For A*-1, (Figures 26-29), the actual paths produced from the navigator are a function not only of the start-end points and improvement strategies used, but are also dependent on the modes used during the map-building. Considerable experimentation was conducted trying to determine which if any of the nine modes available to the map-builder resulted in consistently better paths. No clear connection could be made between the cost of the path, the start and end points of the path, and the nature of the convex region decomposition. In some decompositions, for a given start and end point a better path (A*-1) could be obtained using one decomposition approach over another (Fig. 28 and 29). For another set of start-goal points, however, the same approach that performed poorly in the first case did better than the one that previously performed well. For all

Figure 26: Single terrain path planning example - (A*-1)

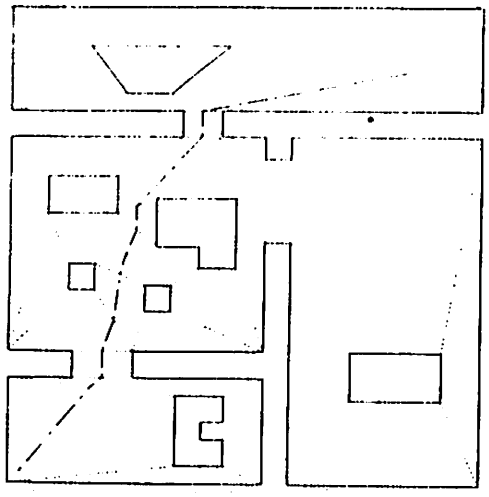
This sequence illustrates the path finding process of the navigator for a single terrain type. The convex decomposition of Fig. 22a is used for Figs. 26-28. The initial start in this case is in the lower left corner, while the goal is in the upper right. Solid lines represent grown obstacles and borders, dotted lines represent passable meadow boundaries and the dot-dash line is the path. The safety margin was specified as 1 foot (on an overall scale of approximately 400' by 400').

a) The initial path produced by the A*-1 search algorithm through the mid-points of the passable bordering meadows.

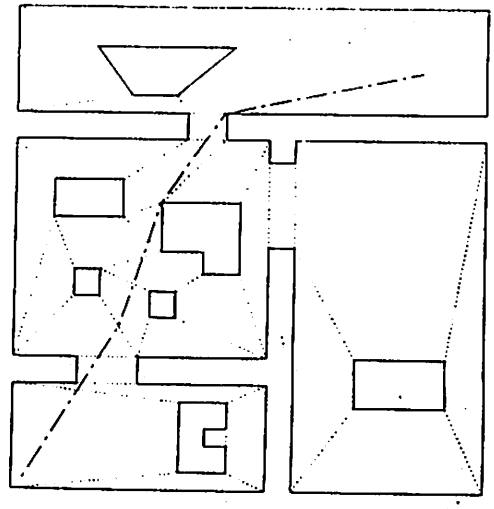
b) The path after undergoing path improvement strategies.

c) The same path as in (b), shown without passable boundaries for clarity.

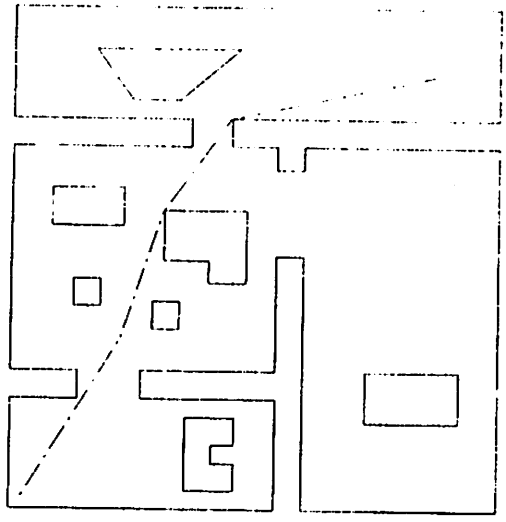
d) A safer path (safety margin 10 ft). Note that a safety margin of 10 feet does not guarantee 10 foot clearance of all obstacle vertices. It serves only to limit the tightening (sliding) along the passable border to within 10 feet of the vertex. It is NOT a measure of path distance from the vertex as one should note in the last corner in the upper right portion of the path.



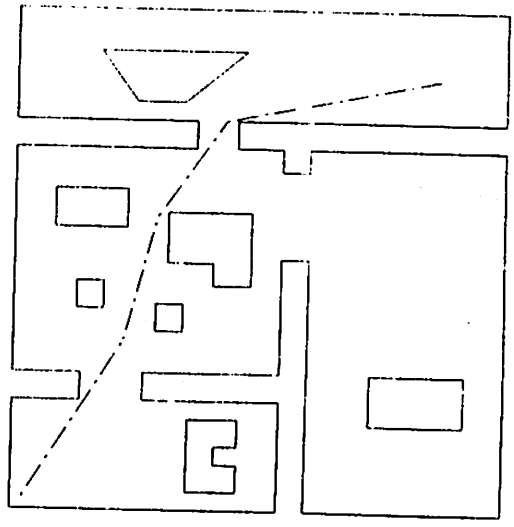
(a)



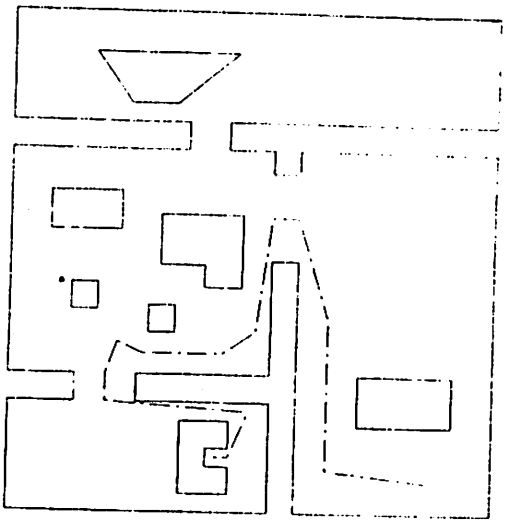
(b)



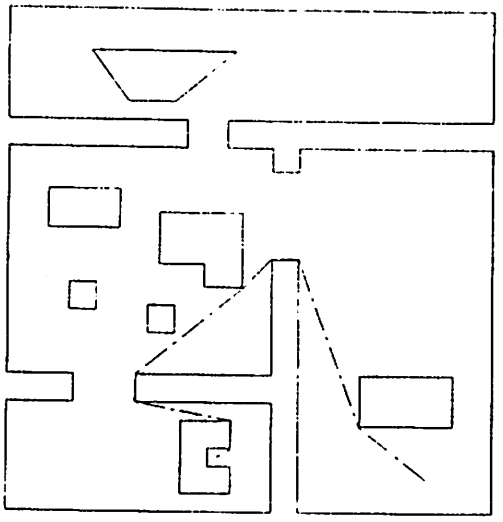
(c)



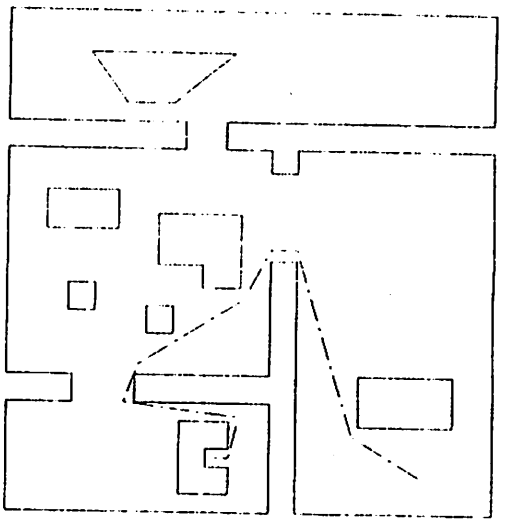
(d)



(a)



(b)



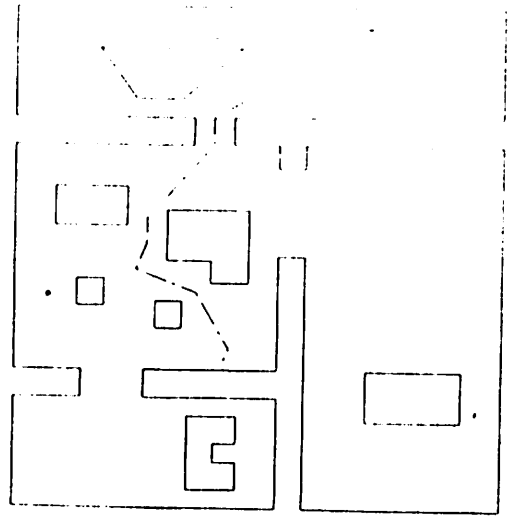
(c)

Figure 27: Another single terrain planning example (A`-1)

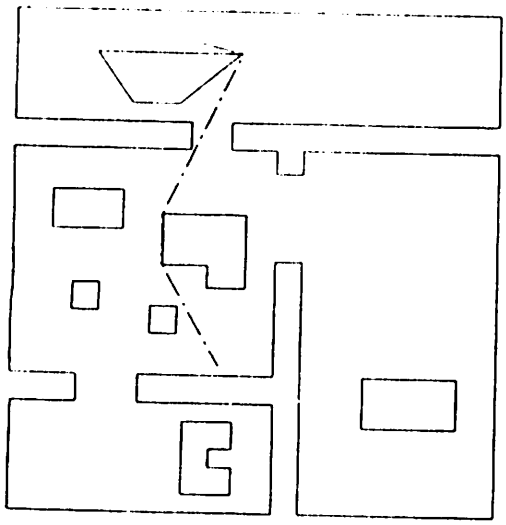
a) Initial coarse path.

b) Improved path (safety margin 1 foot).

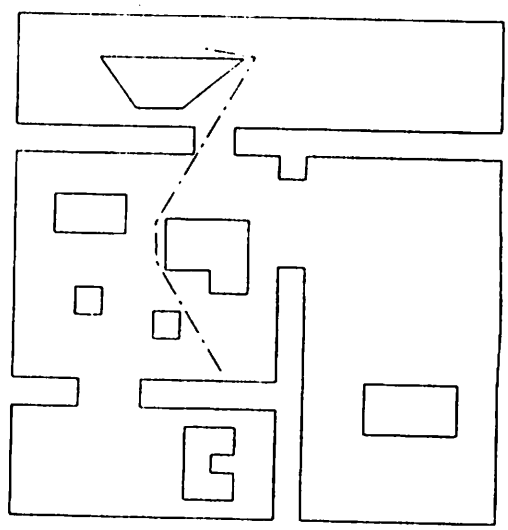
c) Improved path (safety margin 10 feet - see note for Fig. 26d.)



(a)



(b)



(c)

Figure 28: Yet another single terrain planning example (A`-1)

a) Initial coarse path.

b) Improved path (safety margin 1 foot).

c) Improved path (safety margin 10 feet - see note for Fig. 26d.)

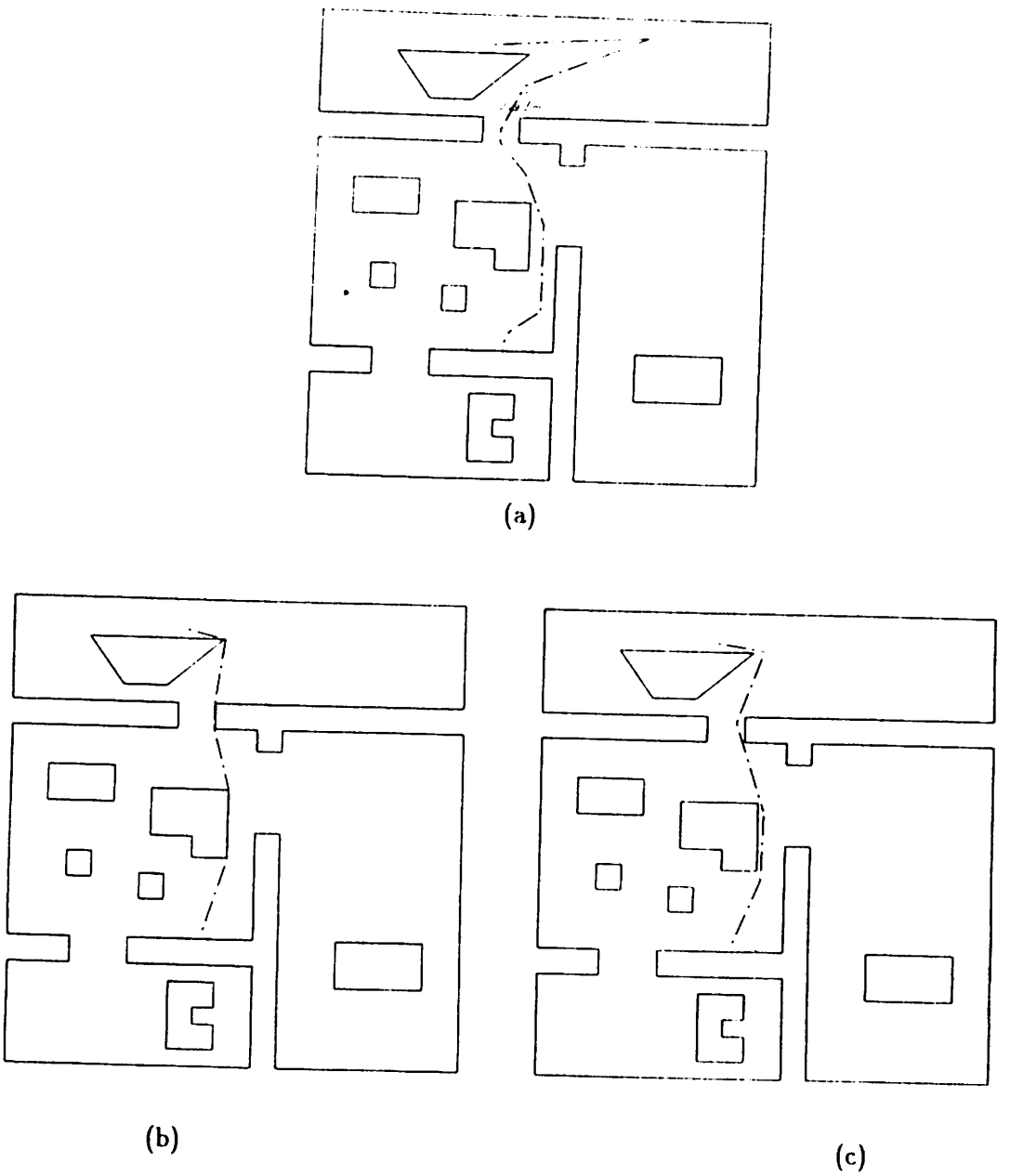


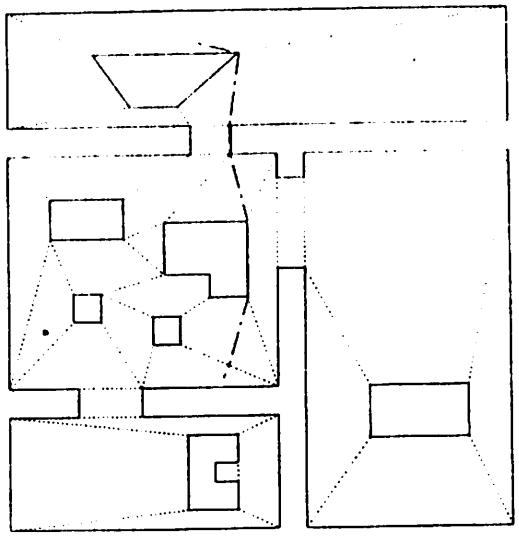
Figure 29: Dependency on decomposition method (A*-1)

In this case, the decomposition method of Figure 22b (most concave vertex, leftmost victim) was used, not that of Fig. 22a (most concave vertex, most opposite victim) as in the previous cases. Although the start-goal points and path improvement techniques are identical with Figure 28, the path produced here is of lower cost. This is a consequence of the decomposition strategy used.

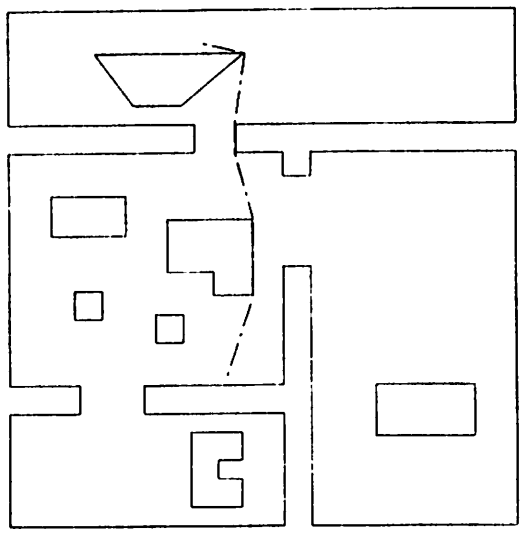
a) Initial coarse path.

b) Improved path (safety margin 1 foot).

c) Improved path (safety margin 10 feet - see note for Fig. 26d.)



(a)



(b)

Figure 30: A*-3 path planning

Contrasting this figure against Fig. 28 (which uses the same decomposition method as is used here), A*-3 search provides the same lower cost path as was seen in Fig. 29 (although a different decomposition strategy was used in Fig. 29). A partial decoupling of the decomposition method and path finding strategy is in evidence.

a) Initial A*-3 search path. Note the selection of points near edges as well as midpoints.

b) Final improved path. Almost identical to raw path (a) in this case.

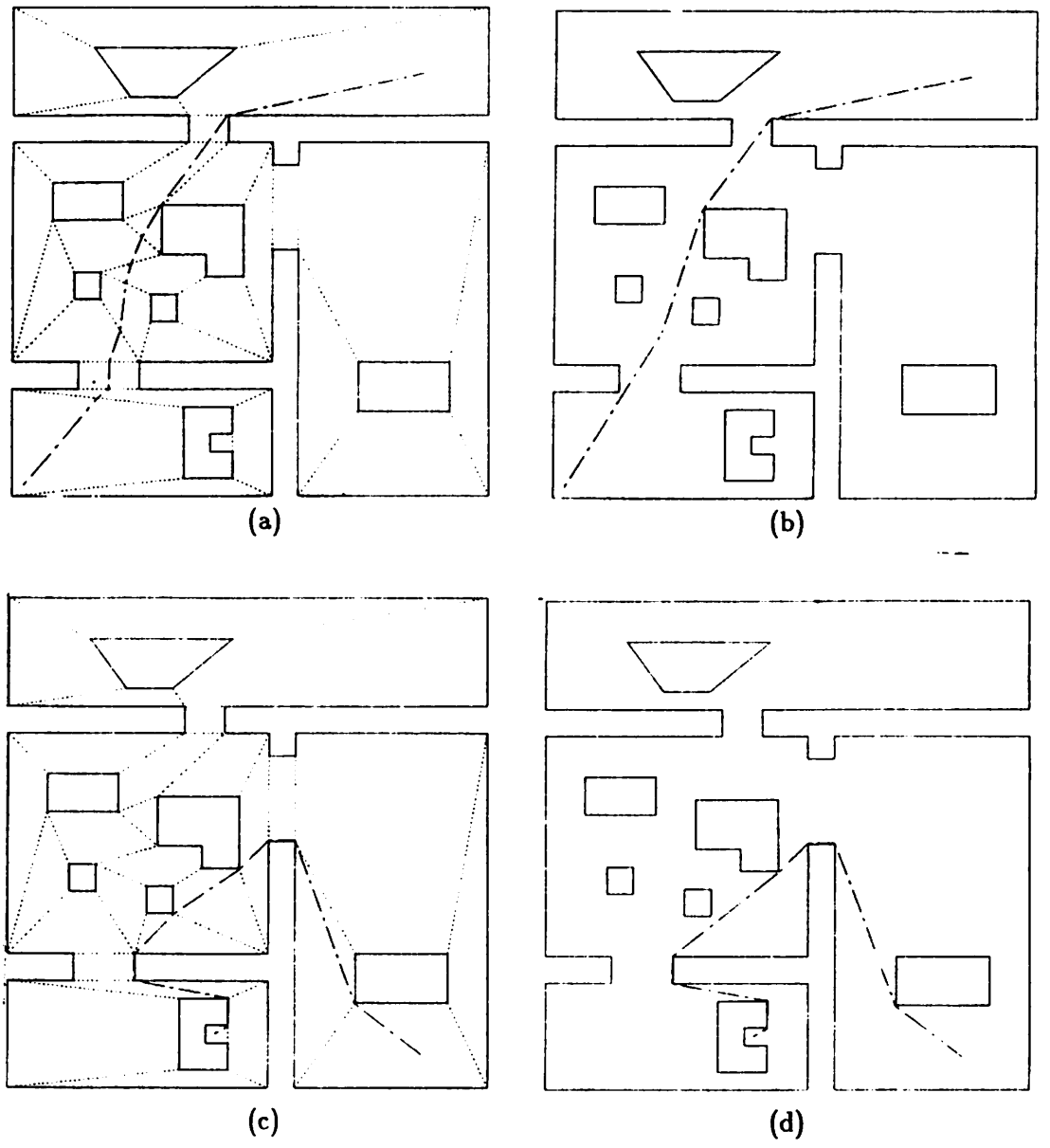


Figure 31: More A*-3 path planning

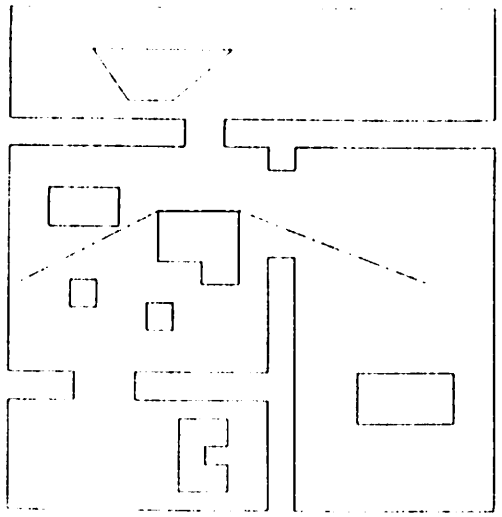
Here it can be seen that the A*-3 method gives no improvement over the A*-1 method for the cases in Figs. 26 and 27. The extra computational cost proves unnecessary.

a) Same start and goal as in Fig. 26 but using A*-3. The initial raw path is shown as a dashed line in relation to passable meadow boundaries (dotted lines).

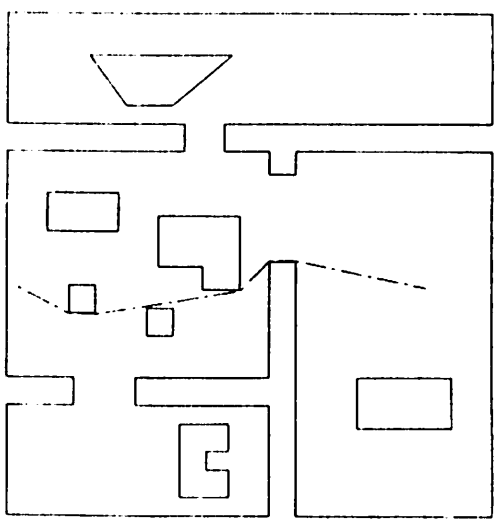
b) Final improved path for (a).

c) Same path as in Fig. 27 but using A*-3. Initial raw path.

d) Final improved path for (c).



(a)



(b)

Figure 32: A*-1 versus A*-3

In tight quarters, A*-3 can make a slight difference. The gain in this example, however, is small (on the order of 2% of the overall cost).

a) A*-1 final improved path.

b) A*-3 final improved path.

possible start and end points within any given map, no single map-building strategy was clearly superior.

For A*-3 noticeable improvement occurred. Figure 30 clearly shows the ability to produce a better path using the same convex decomposition than was the case with the A*-1 method (Fig. 28). Figure 32 confirms these results. The raw paths of A*-3 are generally close to or the same as the final path, whereas this is much rarer in the case of A*-1. The computational penalty, however, can be significant as the search space is considerably larger and is discussed in Appendix A.

Hopefully the diagrams (Fig. 26-32) give a feel for just what a reasonable path is. There are no unnecessary or unexpected turns. When two or more choices are available, if one is significantly more advantageous than the others, the better one will be chosen. If there is only a slight advantage (you might need a ruler to tell as in Fig. 32), one of the best will be chosen. No claims for overall optimality are made, although if suboptimal results are produced for these cases they still qualify as reasonable. Restating that optimality is perhaps a misplaced notion in a dynamically changing world (without constant replanning), the value of spending high computational effort in ensuring absolute minimal costs in the mobile robot domain is unjustified.

The limitations for other representation forms show the advantages inherent in the use of the meadow map approach; these include:

- regular grid: high memory and search cost and digitization bias resulting in sub-optimal paths;
- pure vertex graph: optimal paths only in the context of shortest distance and not amenable to safe path production;
- Voronoi diagrams: more difficult to arrive at short paths and additional representational features not easily embedded.

A major advantage lies in the ability of the meadow map to incorporate virtually any additional representation desired to guide vehicle localization, sensor processing, etc. (see the discussion on the feature editor in Section 2.2). If the paths produced are sub-optimal in the global context and thus are only "reasonable", that is a small price to pay for the versatility and lower memory costs afforded by this representational strategy.

§4. Multi-Terrain Extensions

One of the principal contributions of this work lies in its extension to handle differing terrain types. Previously the regular grid has been the principal representation used to deal with diverse ground covers [88]. Certainly, for the planner to produce realistic paths in outdoor scenarios, a reflection of the different terrain types must be taken into account by the navigator. Some terrain types will be more costly to traverse than others (e.g. gravel or grass as opposed to concrete). We do not want to exclude these different terrains as navigable areas, but yet we don't want to lump them into one uniform terrain type. The traction of the vehicle will depend on the specific surface encountered and more slippage is expected to occur on gravel than on pavement. The cost in terms of positional uncertainty can be high on loose ground. On the other hand, if a significant reduction in the total distance to be traversed from start to goal can be obtained (and associated reduction in time cost), the tradeoff of increased positional uncertainty for greater time savings may be warranted. In some cases the total amount of positional uncertainty gained by traveling over poor surfaces may be substantially less than that garnered by traveling over a superior cover due to the much shorter distance the robot may travel by taking a rougher terrain short-cut.

Another sticky point lies in terrain borders where one ground cover type ends and another begins. If the robot keeps one wheel on one terrain type and the other(s) on a different cover, disorientation can be rapid. One of the goals of the representational strategy used here is to prevent the robot from straddling terrain borders. This is accomplished by the creation of transition zones which separate the ground covers and define clean traversal points. Forbidden zones are also produced which prevent the robot from navigating at the corners of terrain boundaries (regions which typically are expected to be very problematic in terms of maintaining proper localization).

This section first describes how the map-builder accommodates multiple terrain types through the construction of transition zones and their appropriate features. A description of how the navigator has been modified from the uni-terrain model to accommodate path-planning through this extended representation is then presented.

§4.1 *Multi-terrain map-builder*

The extended map-builder is built from the uni-terrain map-builder described in Section 2.1. The algorithm appears in Figure 33.

The input structure of a terrain region is identical to that of the previous map-building algorithm: a list of border and obstacle vertices. This region is decomposed in exactly the same manner as was done previously. Nothing labeling a terrain border is present to identify it as such to the algorithm. Initially, all borders of each terrain region and its enclosed obstacles (which may later turn into other terrain regions) are initially labeled as impassable.

Wherever two different terrain types are found to touch, rectangular transition zones are built allowing a limited type of traversability between them. As a side effect, forbidden zones (corners of intersecting bounding regions), are marked as off-limits for later path planning purposes. This restriction ensures that any path taken across a transition zone will result in a minimal distance path. The transition zone is tagged for recognition by the path planner and other components of the overall system dealing with long-term memory. The details of this process follow.

After the initial terrain area is decomposed, the map-builder algorithm keeps accepting new ones until none remain. After each terrain area is decomposed in isolation, a matching algorithm is run on each new terrain convex region to see if it shares any common edges with any of the previously decomposed regions. This match is performed on the ungrown vertices (or else they would never match). If a match is identified, evidenced by at least the partial overlap of any impassable edges of two different terrain types, a transition zone is built.

The transition zone is a special region connecting two differing terrain types. Most of the data for transition zone construction is already available from the matching process. Basically, the two grown edges, each representing the common border of each matched region, are used for two of the edges of the transition zone (Fig. 34a). This gives a distance across the zone equal to the robot's diameter plus two times any safety margin that was used in the growing (or shrinking) of the initial terrain regions (Fig. 34b). The initial zone consists of the four vertices of the two matched edges.

It is highly desirable to minimize the time it takes for the robot to cross a transition

MULTI-TERRAIN MAP-BUILDER ALGORITHM

```
DO WHILE no more terrain to add
    Run the uni-terrain map-builder (Fig. 15) on a terrain region
    Tag all resulting free space regions with a new terrain identifier
    Match borders of new free space regions against the
       terrain free space regions already produced
    IF matches exist
        Build transition zones connecting terrain types
        Add these transition zones to free space regions
    ENDIF
ENDDO
```

Figure 33: Multi-terrain map-builder algorithm

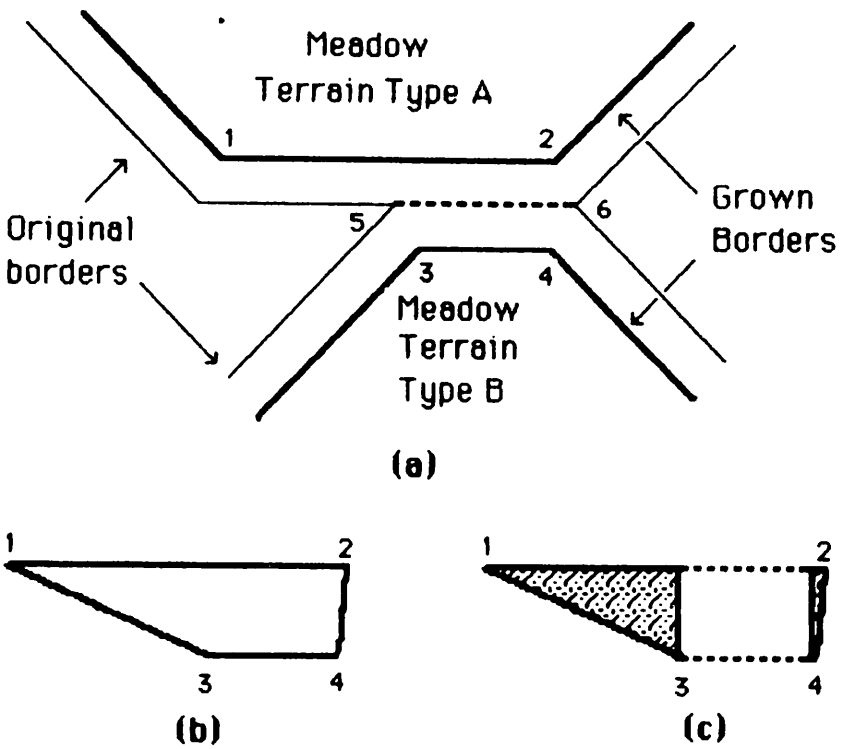


Figure 34: Transition zone construction

a) Initial bordering terrain types. Terrains A and B share a common edge from vertex 5 to 6.

b) The initial transition zone is built by connecting the four vertices of the bordering C-space lines.

c) The initial region is converted into a rectangle yielding the final transition zone. The resulting forbidden zones are shown as shaded areas.

zone, implying a normal straightline path. Consequently the initial polygonal representation is converted into a rectangle (Fig. 34c). The new edges produced (sides of the rectangle) are labeled as impassable, producing small forbidden zones which the planner construes as unnavigable. Any path that is produced by the path planner is guaranteed to be normal to the original matched edges, thus ensuring the smoothest and fastest transition possible from one terrain type to another. Finally, appropriate passable links are made to connect the new transition zone and the two bounding free space regions of the different terrain types.

The traversability factor (used for costing in path planning) should be high for transition zones due to the problems associated with terrain changes. Currently the traversability of a transition zone is defaulted to the sum of the traversabilities of the two bordering terrain types. This value can be readily changed if appropriate via the feature editor (Section 2.2).

§4.2 *Multi-terrain Navigator*

The navigator must be modified somewhat to ensure that the path produced is reasonable in the multi-terrain case. The only components of the navigator that must be changed are the path improvement strategies. This includes both the straightening and tautness components. No modifications whatsoever are necessary for the search component because the terrain cost is included in the cost function. As the transition zone is rectangular, any path produced by the A*-1 method through the midpoints of passable regions crossing over different terrain types is guaranteed to result in a straightline across the transition zone. The A*-3 case occasionally requires slight path preprocessing to ensure a perpendicular crossing of the transition zones prior to improvement. Unfortunately, modifying the path improvement strategies (for both A*-1 and A*-3) was non-trivial and required the implementation of a relaxation algorithm, limited to relaxing the terrain crossings only. Previously, terrain traversability costs were ignored during the path improvement process, yielding shorter but more costly paths over more difficult terrain. The relaxation approach readily ensured that traversals across the transition zone remained perpendicular to the border edges while still producing low cost paths dependent on the nature of the ground cover.

The algorithm for multi-terrain path improvement is shown in Figure 35. Actually,

the complexity is somewhat greater due to special case treatment (start or end within transition zone, entire path in transition zone, etc.). Sedgewick [114] states that "special cases ... are the bane of geometric algorithms", and I am in firm agreement with him.

The algorithm proceeds as follows: the previous path improvement strategy is first run within the framework of each terrain type in isolation. This is the identical algorithm as described in Section 3.2 but restricted to individual terrain types. To reduce the cost of the relaxation later, the transition zone crossings are then slid in the same manner as was done for the individual meadow border passages, with one exception. Both crossing points on the transition zones are slid in tandem, insuring a perpendicular passage across the transition region. This step generally reduces the overall distance the transition zone crossings will have to be moved during the relaxation phase, thus reducing computation time. Any previously unmoved vertices within the regions themselves are then retested to see if sliding will lower the overall cost. If necessary, additional path straightening is then performed.

Although avoiding a relaxation method for path improvement was an initial design goal due to perceived high computational costs, (as in relaxation on a regular grid), it eventually became necessary to resort to one. The cost associated with this relaxation (see Appendix A) is not particularly high however, due to the preprocessing on the path and, more importantly, only the transition zone crossings are relaxed, not all passable borders. The algorithm used is fairly standard: displace the transition zones an increment in both directions and measure the lowest cost. Use the new lower cost point as the starting point for the next displacement. Keep repeating until any displacement results in a higher or equal cost path. Convergence is guaranteed using this standard hill-climbing methodology. The time for convergence is determined to a large extent by the displacement size and on the number of terrain crossings. The results for the worst test cases in the lab have yielded times for the relaxation component that are not disproportionate (typically the same order of magnitude) with the other components of the algorithm (see Appendix A). The best case (no transition crossings) is virtually identical to that of the uni-terrain results (Section 3.3); the average case results in slight increases in path improvement time. Finally, path straightening is reattempted within the context of each terrain type before final release to the pilot.

MULTI-TERRAIN PATH IMPROVEMENT ALGORITHM

Accept a coarse path from search component of navigator.
Run tautness and straightness component of uni-terrain path planner
    on each part of path within a given terrain type (Fig. 24).
Slide only the transition zones as in previous tautness algorithm.
Run tautness and straightness component again on each part of path
    within a given terrain type (only on previously unmoved vertices).
Relax path by settling transition zone crossings into
    a minimal cost point.
Restraighten if necessary.

Figure 35: Multi-terrain path improvement algorithm

§4.3 *Results*

A schematic model of the environment outside the Graduate Research Center was used for the outdoor terrain examples. Five different terrain regions are present: concrete, two disjoint grassy regions, a gravel path and a parking lot. For the purposes of path planning: the traversability of the concrete and the parking lot was set to 1.0, grass 1.5, and gravel a factor of 1.2 (these are relative values: the higher the number, the more difficult to traverse). The gravel path, although rough, has the decided advantage of path borders, which make path-following strategies available that are not useful on grass. The terrain types and their associated transition zones can be seen in Figure 36.

In Figures 37 through 39, the results of the path planning algorithm are illustrated. The A*-1 search method was used for all these cases. Sub-figures 37a-39a shows the initial path through the search space. Note in this and all other cases the perpendicular passage through the transition zone is evident. Sub-figures 37b-39b show the improved path before transition zone relaxation. Sub-figures 37c-39c display the final path after relaxation and post-relaxation straightening.

§5. Cartographic subsystem

The details and construction of long-term memory by the cartographer's map-builder process has been described in Section 2. The uncertainty management subsystem is described in chapter 7. What remains to be discussed here is the structure and maintenance of short-term memory. The subsections following will present the STM structure, the STM manager's role in creating and maintaining the perceptual level of STM, and the cartographer's meadow instantiator process which provides the LTM context for STM that is drawn upon by the pilot in the event of motor schema navigation failure.

§5.1 *Short-term memory structure*

Short-term memory is a bi-level structure (Fig. 40). Its primary purpose is to provide information for navigational purposes (i.e. it does not serve the same purpose as VISIONS STM). At the base level, it consists of a group of meadows from LTM which define the context for the current pilot goal. Recall that LTM consists, to a large extent, of a

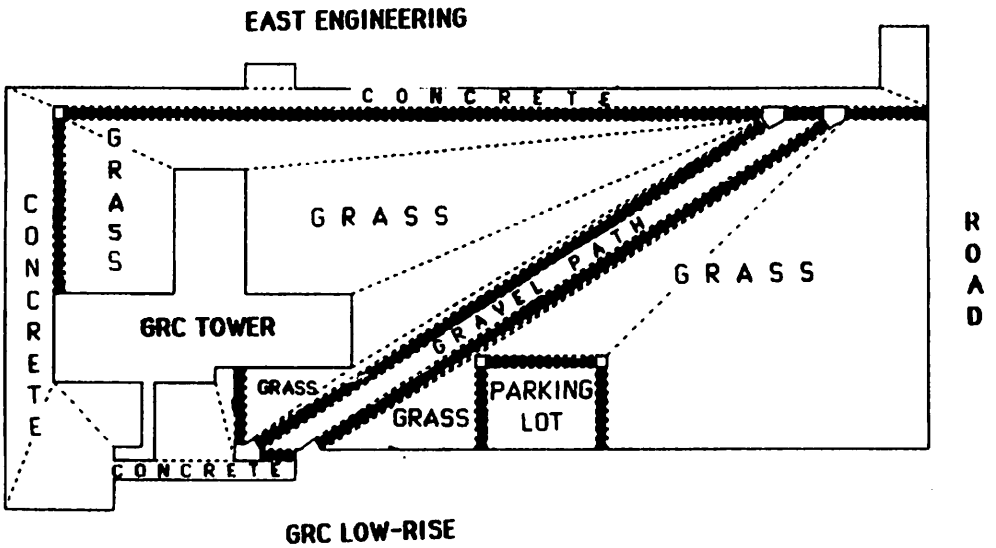


Figure 36: Multi-terrain map

A *schematic* diagram of the area surrounding the Graduate Research Center. All impassable regions are represented as solid lines, passable meadow boundaries as dotted lines. The passable transition zones are the solid rectangles. The different terrains (grass, concrete, parking lot and gravel path) are labeled. (Scale approximately 320' by 180'. This is much smaller than is actually the case, but necessary to clearly show the transition zone-path relationship in the figures to follow, i.e. the transition zones are larger than they would appear otherwise).

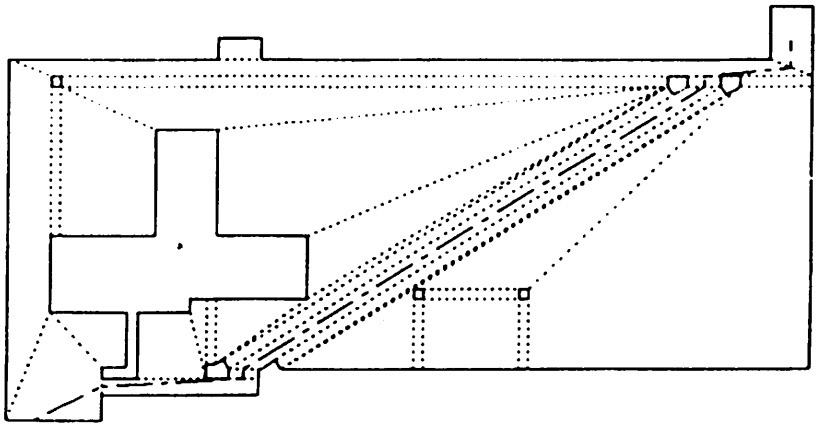
Figure 37: Multi-terrain path planning example

The start point is in the lower left corner on concrete with the goal in the upper right on concrete. The path planner decides it is more efficient to take the gravel path to achieve its goal, requiring the traversal of two transition zones.

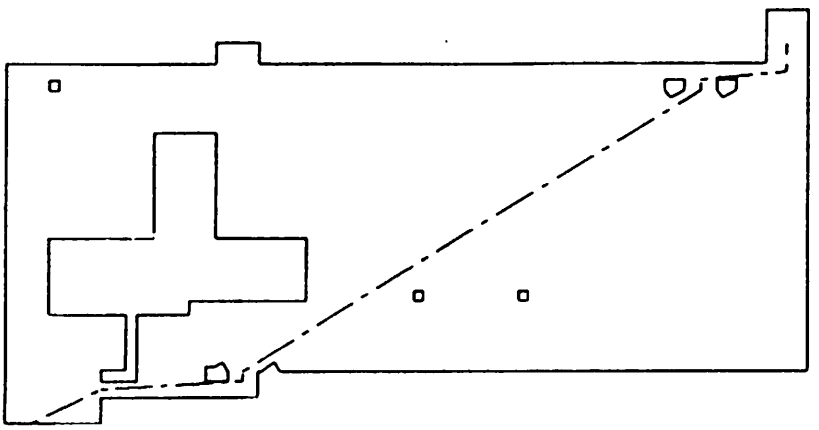
a) Initial raw path from A*-1 search through midpoints of passable regions. The cost function includes a traversability factor dependent on terrain.

b) The same path without the passable borders. Note the forbidden zones present at the edges of transition zones.

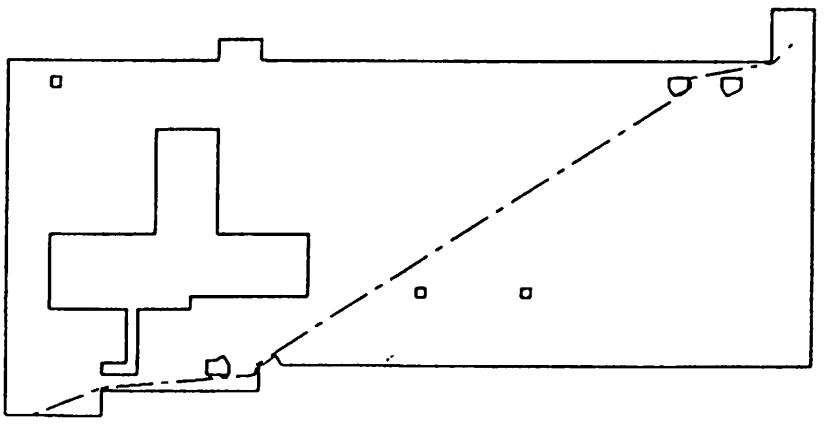
c) The final improved path. Note how the total length over concrete was lengthened while the distance over gravel shortened (safety margin 1 foot).



(a)

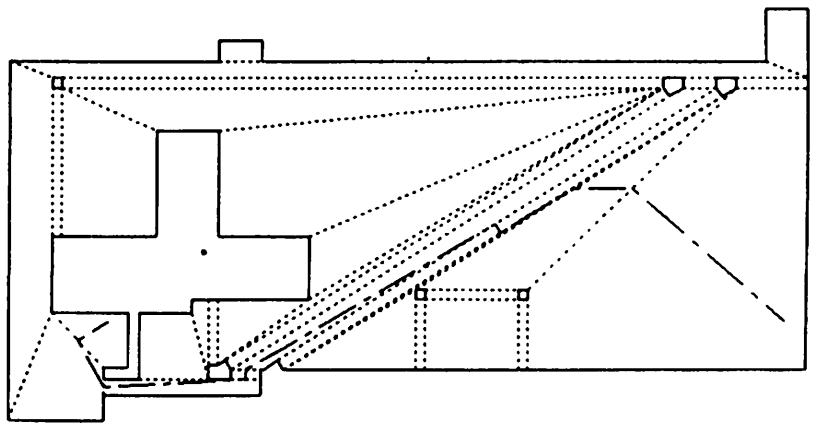


(b)

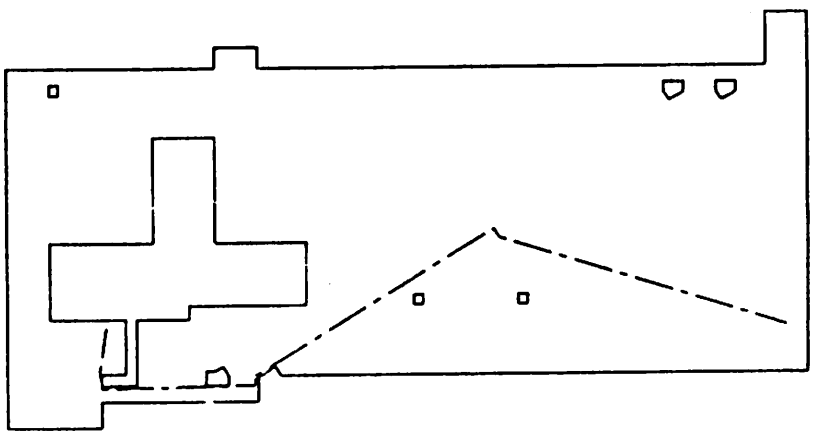


(c)

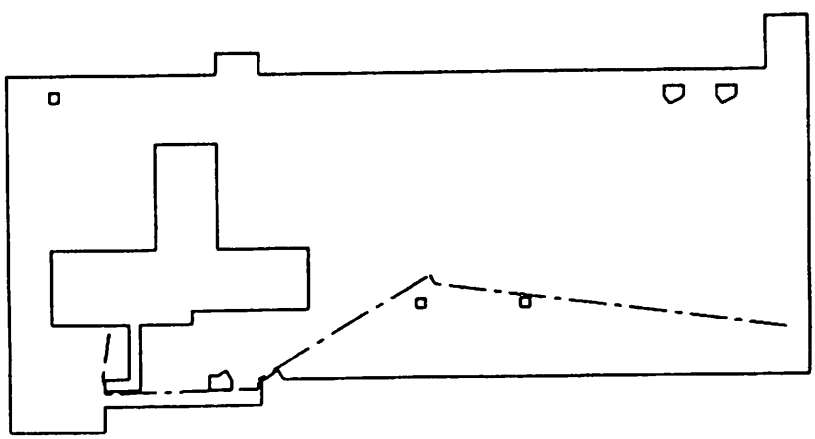
Figure 38: Another multi-terrain path planning example
Start in lower right, goal in lower left.
a) Initial raw path - crosses two transition zones, grass to gravel, gravel to concrete.
b) Improved but unrelaxed path. Grass to gravel crossing is at midpoint of transition zone.
c) Final relaxed path - note the repositioning of the grass to gravel crossing.



(a)

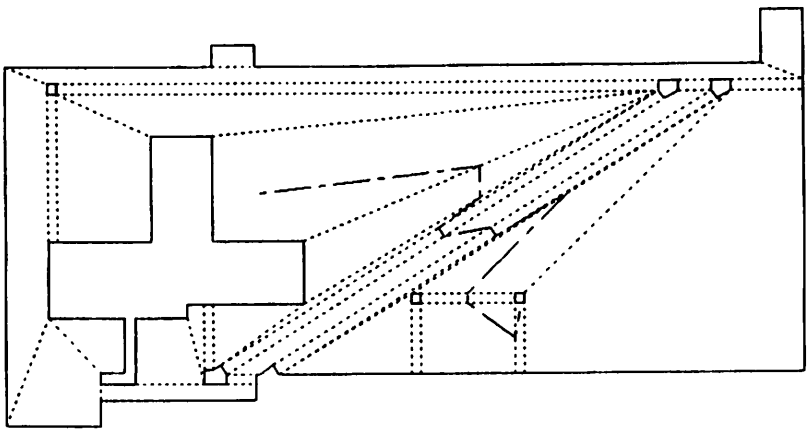


(b)

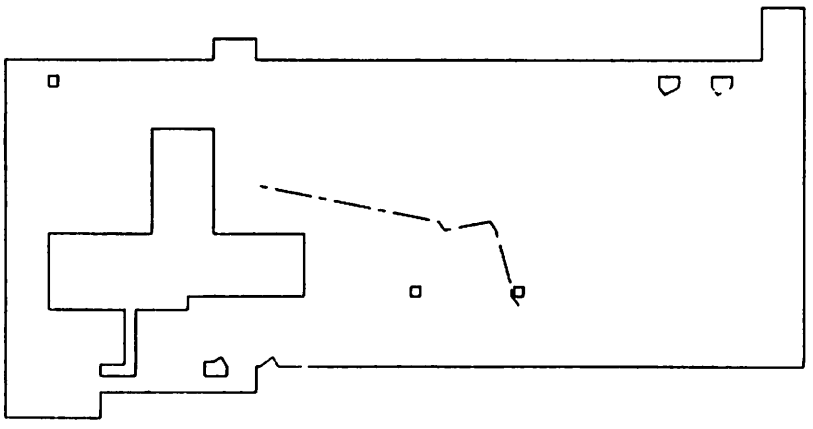


(c)

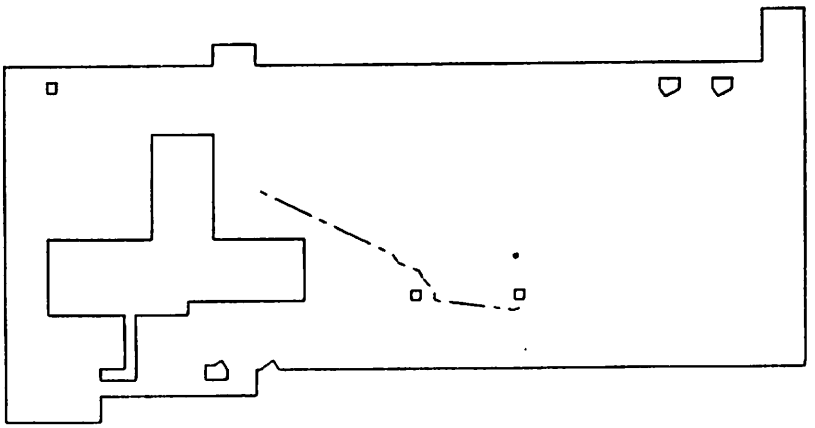
Figure 39: Yet another multi-terrain path planning example.
Start in lower right, goal in upper left.
a) Initial raw path.
b) Improved but unrelaxed path.
c) Final relaxed path.



(a)



(b)



(c)

connectivity graph of these meadows. The base level defines the context of the world map against which sensory acquired information to be applied. These meadows are moved into STM by the meadow instantiator process (Sec. 3.2). The overlying STM top level contains a sensory-based model built up by the cartographer as the robot travels through the world. A local frame of reference based on the current pilot leg is used for this level. The STM manager incorporates the sensor data as it is received into this top layer. The details of each level follow.

The instantiated meadow level is crucial for the correct selection of motor schemas by the pilot. Basically it contains pointers to the parent LTM meadows and tags indicating if the meadow is the start, end, on-path, adjacent, or other type of meadow. This enables the retrieval of terrain characteristics, potential landmarks and other information pertinent to the robot's current position in the world without a time-consuming search of LTM. The provision of context for the sensory built top-level of STM is also of major significance. The top level of STM, consisting of a regular grid embedded with sensor information, should be viewed as an overlay on these meadows (Fig. 40). Thus, when the pilot needs to reorient the path for whatever reason, spatial occupancy information from the sensor (top) level can be moved into the base-level instantiated meadows. The meadow "fracturing" process that is performed by the pilot, when the need arises (described in Section 6.2 below), recomputes locally the robot's path based on information from both sensor data (represented in high-level STM) and LTM models. This is more efficient than reinvoking the navigator to compute a global path anew.

The top level of STM uses a grid representation of space. It is based on extensions of the method used by Moravec and Elves [92,45] for interpreting sonar data. Navigational space is tessellated into a grid, with each square containing information regarding the occupancy of the area (whether it is free space or filled with an obstacle). This map is built up from sensor readings acquired from the robot as it travels. These readings taken from the robot's egocentric frame of reference must be combined into a robot position-independent representation. The frame of reference used for the grid representation is a "local-global" coordinate system. It does not correspond directly to the robot's egocentric vantage point nor does it match the LTM global world model. It is a local model of the world that is built solely from egocentric sensor measurements taken from the vehicle. The resulting grid must be correlated against the lower instantiated meadow level of

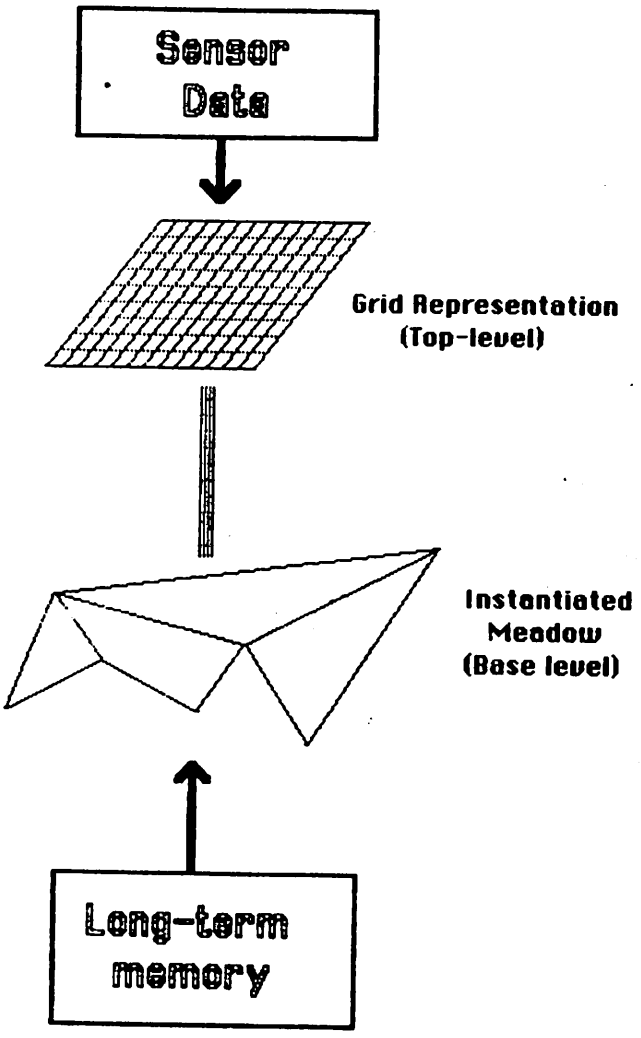


Figure 40: Short-term memory

STM (which is represented in world coordinates from LTM) so that the robot's bearings can be measured against known world features. The uncertainty in the top level of STM is expressly represented through the use of a numerical value whose absolute value increases for a particular cell based on confirmatory sensor data and decreases based on contradictory data. The STM manager, described below, handles the maintenance of this level of STM.

The spatial resolution of the top-level grid is typically 64 feet by 64 feet. Its structure need not be square but was so decreed, for the first implementation. As the path leg is generally longer in one dimension a 128 by 32 grid might be more appropriate for future generations of AuRA. This grid moves through navigational space each time the robot undertakes a new leg on its journey. The cells consist of the following data:

- Occupancy Value
  range: −1 (highly probable to be unoccupied and hence unnavigable)
  to 1 (highly probable to be occupied. 0 denotes no information - (as in [45])

- Symbolic tags
  Again with certainty level (range 0 to 1) This would include not only LTM landmarks but also path edge information and terrain-type identifier where appropriate. Several may be active for each cell.

A major problem for this approach, as discussed by Brooks [25] and handled by Smith and Cheeseman [118], is the incorporation of multiple measurements from different uncertain positions into a single representation (i.e. the STM regular grid). There is no simple solution. In this case, the assumption must be made that the position from which the robot is currently taking its measurements is relatively well known. Fortunately, the STM representation need only be used when reflexive motor schema-based navigation fails, which should be relatively rare.

A form of environmental learning is also feasible when information regarding obstacles or features of high certainty could be moved from STM into LTM for future reference. This is left for future AuRA implementations.

Certain other information is stored in STM. This includes the route-list developed by the navigator. Event flags are also present indicating items such as the deposition of the route by the navigator as well as route completion by the pilot. Current leg pointers are

also maintained here. These flags are used for inter-process synchronization. Bounds for STM extent are also present. A point estimate of the robot's position is also maintained based on shaft encoder data (independent of the spatial error map).

§5.2 *STM meadow instantiator process*

The meadow instantiator is a separate process running under the control of the cartographer. Its operation is fairly straightforward. On start-up it initializes the base-level of STM. It then waits until the navigator has placed a route into STM and has set the route-deposited flag. Meadows are then moved into STM based on the first leg of the route. The meadow instantiator then waits until the pilot signals it has completed its leg. The old meadows are then deinstantiated (made inactive) but not removed from STM (unless available memory requirements necessitate it). The next leg of the navigator's route is fetched and the relevant inactive meadows already in STM are reactivated, while any new meadows needed are accessed from LTM. This process repeats until the navigator's route is completed.

§5.3 *STM Manager*

The STM manager's role is to modify short-term memory based on arriving interpreted sensor data. The approach of Moravec and Elves [92] for incorporating multiple sensor readings from different spatial locations into a single grid representation is used. Although their work to date deals solely with ultrasonic data, information from other sensor modalities such as vision can be incorporated.

No single sensor reading is sufficient to guarantee that the existence of an obstacle or other environmental object is present. Instead, multiple readings from different locations are merged using a probabilistic approach to build up certainty in the position of an obstacle. The ultrasonic reading does not give a precise environmental location of an object. Due to the nature of the sonar scan, only a wedge of possible locations is available. The multiple readings are folded together to yield a measure of the uncertainty of occupancy or free space for each grid square [92]. The STM manager is built around code imported from CMU (Moravec and Elves' work). Only minor modifications have been made to enable it to be tied into the AuRA architecture. Figure 41 shows a typical map built up in STM from multiple sonar readings of the UMASS robot lab.

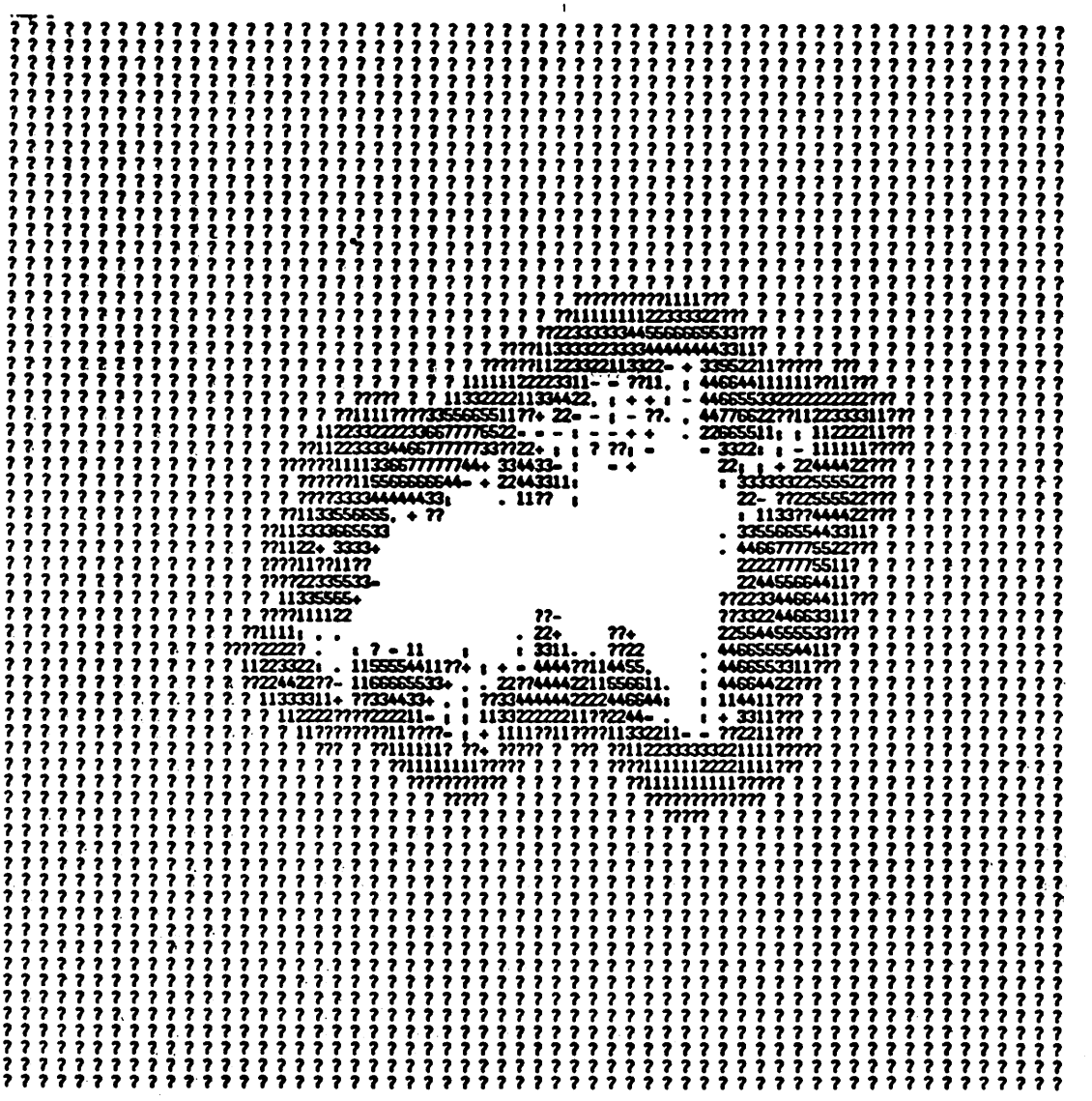


Figure 41: Top-level of short-term memory (Occupancy view of the robot lab)

Grid values indicate certainty of occupancy - numbers denote the likelihood of being occupied, while whitespace and punctuation marks denote the likelihood of being unoccupied. (This is a view of the ground-plane from above).

Based on code supplied by Hans Moravec [92]

The STM manager accepts arriving reports from the interpreter processes in the perception subsystem and incorporates this data into the STM grid. This perceived world model is built up concurrently and independently of the motor schema manager. It is referred to by the planning system only upon failure of the motor schema manager to achieve the specified pilot goal. This is detected by the exceeding of a hard real-time deadline for goal attainment or by the robot's velocity dropping to levels that are deemed too slow for successful completion. The pilot then draws on this data, merging it into the instantiated meadows in STM and then fracturing the meadows (Figure 42) to allow local navigational replanning. This meadow fracturing strategy used for local path reorientation by the pilot is described below in Section 6.2.

§6. Mission planner and pilot

AuRA's planning subsystem consists of the mission planner, navigator, pilot and motor schema manager. Motor schema based navigation is described in Chapter 5 and the experimental motor schema system is described in Chapter 8. What remains to be discussed in this section are the implementation details of the mission planner and pilot.

§6.1 *Mission planner*

The mission planner's implementation for the first pass design of AuRA is at best rudimentary. Reviewing the goals of the mission planner, they are: to perform spatial reasoning, the determination of navigation and pilot parameters and modes of operation, the selection of optimality criteria and the handling of navigator failure. As the mission planner serves as the interface to the human commander, a natural language front-end facility is also desirable. The mission planner's chief function is to provide a series of specific subgoals, based on a high-level request, for the navigator to act upon. These requests might include such things as:

- Survey or reconnoiter an area
  (search for something lost, make reports on unusual events, etc.)
- Obtain a particular list of items found in different locations
  (requiring the determination of where the items are and then an appropriate order-

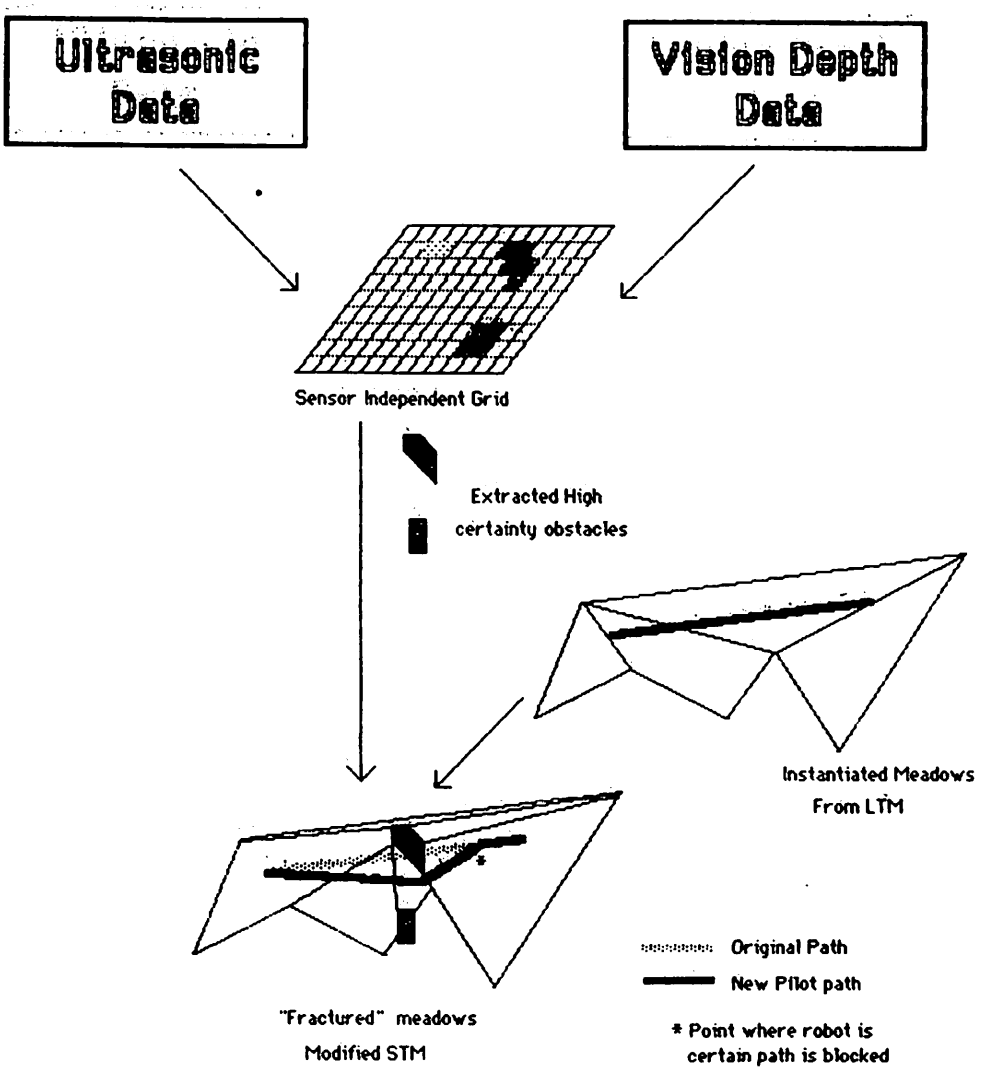


Figure 42: Meadow fracturing
The two polygonal obstacles with relatively high certainty are extracted from the top-level grid and are embedded in the low-level instantiated meadows. This enables the pilot (Sec. 6.2) to recompute the path within a local context, thus avoiding reinvoking the navigator.

ing for their retrieval)

Currently, the mission planner performs none of these functions other than acting as the interface to the commander (without natural language ability). All goal ordering is established by the human agent and is passed through the mission planner without interpretation. The mission planner also provides a facility to set the parameters that affect navigator and pilot operation interactively, but does not interpret the "mission" to automatically determine those settings.

This rudimentary form reserves the rightful function of the original mission planner design for a more complete implementation. The responsibility is passed upward to the human commander and is not relegated to lower levels of the planning hierarchy. It is hoped, and research interest exists at UMASS, that other researchers will complete and/or enhance the capabilities of the existing mission planner. New versions should be readily incorporable into AuRA.

§6.2 *Pilot*

The pilot serves a dual purpose in the AuRA architecture. First and foremost is its responsibility to analyze available data regarding the current navigator leg to be completed. From this information (gleaned from both LTM and STM) it selects appropriate motor and perceptual schemas to be instantiated within the motor schema manager. The pilot then suspends itself while the motor schema control system manages the actual path traversal. The second role of the pilot is to handle failure of the motor schema manager to reach its desired goal. This can be evidenced in two ways: by the cessation of motion by the robot without goal attainment, or by a clock time-out signaling failure to reach the goal within a pilot-established hard real-time deadline. The pilot then attempts to reroute the path *locally* based on information stored in STM by the cartographer. Only if this pilot-based reorientation procedure proves unsuccessful is the navigator reinvoked to recompute a new *global* path in light of the newly discovered environmental blockage.

The following subsections describe these two roles of the pilot.

Pilot schema selection process

In order to provide appropriate motor action for a specific path traversal, the pilot must access information that is present in both long- and short-term memory as well

as the current goal that the navigator has established for the pilot. Long-term memory contains the terrain characteristics, landmarks and other relevant characteristics which the pilot is required to draw upon for motor schema slot-filling. Short-term memory contains the instantiated meadows (Sec. 5.1) that are relevant for the current pilot leg. Blackboard values specifying motion characteristics (velocity, acceleration, etc.) are also drawn upon. These are generally set by the mission planner and possibly modified by the homeostatic control system. The navigator's subgoal is passed to both the pilot and cartographer through global memory and is used by the pilot to establish the hard real-time deadline for the navigator and to set failure limits for certain motor-schemas (e.g. **move-ahead**). Information is extracted by the pilot from all these sources and moved into a *fact-base* at the start of each pilot leg.

The approach used in the first pass implementation uses a simple set of rules and a primitive inference engine to select appropriate motor schemas based on the current navigational subgoal. A *rulebase* is maintained separately for the pilot's use. The rules contained therein are applied by the pilot to the current context (the *fact-base*). The result is a list of parameterized motor and perceptual schemas (drawn from a set of schema templates) customized for the particular navigational leg. Some schemas are invariably produced (i.e. **avoid-static-obstacle** schema), while others are produced as the current context dictates. Slot-filled **find-landmark** schema templates, in particular, are produced by the pilot by application of the rules in the *rulebase* to the specific landmark data in the *fact-base*. Activation conditions, useful perceptual schemas for identification, and other related criteria are passed with the schema itself to the motor schema manager.

These motor and perceptual schemas effectively exist in three forms: the naked unparameterized (but with specified defaults) schema templates, residing as structures in LISP code in the schema database; parameterized, but as yet uninstantiated, schemas passed from the pilot to the motor schema manager; and the schema instantiations themselves (SIs) existing within the motor schema manager. The schemas passed to the motor schema manager undergo a transformation into a form compatible with the schema shell. Only the information necessary to flesh out the schema shell SIs is passed from the pilot to the motor schema manager, keeping the communication bandwidth at reasonable levels.

Local path reorientation

The other function of the pilot is to handle failure of the motor schema manager to attain its specified goal. The potential field methodology is vulnerable to failure due to local potential minima or isolated peaks (see Chapter 5). Although this should be relatively rare due to the navigator's selection of a path that avoids all modeled obstacles, the presence of significant numbers of unmodeled obstacles can cause the potential field based system to falter. When this occurs, as evidenced by exceeding a real-time deadline to achieve the pilot's established goal (for cycle detection) or by the vehicle's velocity dropping to too low a level, the pilot is reinvoked.

The pilot, upon motor schema manager failure, attempts to reach the navigator's subgoal by moving obstacles from the STM grid representation into the instantiated meadows so that they now serve as modeled obstacles for the path planner. These newly discovered obstacles are first merged into the border created by the instantiated meadows. The same algorithms used by the cartographer to build LTM are then used to decompose these instantiated meadows into a new series of convex regions. The navigator's path finding algorithm is then run within this limited context. The new series of path legs gives the robot a new approach towards negotiating these unmodeled obstacles. If a path is unattainable, the navigator is reinvoked to find a new path on a global basis. This level of planning within the pilot is "local-global" as it draws both upon the perceived world model as well as the instantiated meadows taken from LTM. Figure 42 illustrates this process.

It should be remembered, however, that this role of the pilot should be infrequently used. Nonetheless, due to the fact that potential fields are susceptible to local sensing problems, techniques to handle this difficulty must be available.

§7. Summary

In order to produce a reasonable path through a partially modeled environment, a hybrid vertex-graph free space representation was chosen for a long-term memory model of the world. This meadow map decomposes free space into a group of connected convex regions. Data for landmark recognition, localization, uncertainty modeling, and the like,

can be associated with these regions or their obstacles through the use of a feature editor. Although LISP might be a preferable language due to its symbol manipulation capabilities, all coding was done in C to insure rapid processing to meet the demand for real-time response.

Multiple terrain situations are accommodated by extending the basic algorithms to include the construction of transition zones. These zones assure minimum distance traversal when the robot changes terrain types, minimizing the increase in positional uncertainty inherent in this maneuver. When facet models are applied to the meadows to reflect topography, and the ground plane assumption discarded, this method will become even more powerful for navigation in outdoor terrain.

It is conceivable that this map could be acquired dynamically by interaction with the environment as has been demonstrated by work with HILARE at LAAS [52] and Neptune at CMU [92]. Environmental acquisition via learning will not be addressed in the near future in our work, although other UMASS researchers may be involved in this research.

The output of the map-builder is utilized in part by the navigator component of the planner process whose duty it is to build a collision-free path through the partially modeled world represented in LTM. An A* search is conducted through the midpoints (A*-1) or triads (A*-3) of the bordering passable convex regions to arrive at a coarse path. The A*-3 method offers a tighter initial path at the expense of a greater search space. In a highly cluttered environment, this can be of value. Generally, the A*-1 search method works almost as well and is less costly to produce.

The resulting raw path is then subjected to path improvement strategies which tighten and straighten the path subject to parameters specified by the mission planner. These path improvement strategies are extremely important for producing quality paths and are an important contribution to the overall path planning process. They allow for the production of short paths, safe paths, or other types of paths, attempting to optimize across a set of paths more freely than other representations might allow. This flexibility offers distinct advantages over a Voronoi diagram or vertex graph approach to world representation. Certainly the regular grid's search space makes it computationally infeasible for all but the simplest domains. The regular grid's susceptibility to digitization bias (quadtrees even more so) also make it a less desirable representation. The flexibility to accommodate diverse new representations, without any changes to the underlying path

planning representation is another of the advantages of our approach. The meadow map, in our estimation, is the best general purpose navigational representation, and as such will, in various forms, continue to be a major force in mobile robot navigation.

Regarding implementation, the navigator, mapbuilder, and LTM representational structure for AuRA are all complete. Knowledge acquisition is ongoing, especially regarding 3D landmark models. There is always more knowledge to be added. The mission planner is currently rudimentary, serving as a command interface. The pilot and STM are not yet fully integrated. For the experimental system in Chapter 8, the schema handoff from the pilot to the schema system requires user intervention. The rulebase has a limited set of rules for schema generation and will be expanded in the near future. The STM instantiator process is complete, as is the first pass version of the STM manager. These exist as separate processes running under the control of the cartographer. The meadow fracturing process will require additional work on the obstacle extraction from STM component for its implementation, but as this is used only in the relatively rare circumstance of schema navigation failure, it is not a top priority.

CHAPTER V

MOTOR SCHEMA BASED MOBILE ROBOT NAVIGATION

Path planning and navigation, at the execution level, can most easily be described as a collection of behaviors. *Don't run into things! Go to the end of the sidewalk then turn right! Stay to the right side of the sidewalk except when passing! Watch out for the library - the turn is just beyond it! Follow that man!* This collection of commands constitutes some of the possible behaviors for an entity trying to move from one location to another. Traditional control structures – those that use an inflexible and rigid approach to navigation – do not provide the essential adaptability necessary for coping with unexpected events. These events might include unanticipated obstacles, moving objects, or the recognition of a landmark in a seemingly inappropriate location. These unexpected occurrences should influence, in an appropriate manner, the course which a vehicle (or person) takes in moving from start to goal.

A potential solution can be drawn from models that have been developed in the domains of brain theory and robotics. Schemas, a methodology used to describe the interaction between perception and action, can be adapted to yield a mobile robot system that is highly sensitive to the currently perceived world. Motor schemas operating in a concurrent and independent, yet communicating, manner can produce paths that reflect the uncertainties in the detection of objects. Additionally they can cope with conflicting data arising from diverse sensor modalities and strategies.

The purpose of this chapter is to provide insights into the design of a control system based on motor schemas for mobile robots. Section 1 describes the motivations for the use of schema theory in this domain - drawing from work in both brain theory and robotics. Section 2 discusses the tack taken for a motor-schema-based control system in the Autonomous Robot Architecture (AuRA), utilizing a mobile robot equipped with ultrasonic and video sensors; specifically the role of the pilot and the motor schema

manager. Section 3 presents the results of simulations using schemas that specify different behaviors and draw on simulated sensor input. Section 4 describes the implementation of the motor subsystem in AuRA. A summary and evaluation concludes this chapter.

§1. Motivation

The concept of schemas originated in psychology [19,99,109] and neurology [57,48]. Webster [138] defines a schema as "a mental codification of experience that includes a particular organized way of perceiving cognitively and responding to a complex situation or set of stimuli". The model used for this paper draws on more recent sources: the applications of schema theory to brain modeling and robotics. As brain theory can unequivocally be called a sound basis for the study of intelligent behavior, the first part of this section will present the contributions of brain science that influenced the design of the schema control system described below. Roboticists for some time have drawn on schema theory, not always in the form envisioned by brain theoreticians. The previous work in robotics that relates to the schema-based approach to navigation is described in the final part of this section.

§1.1 *Brain Theory and Psychology*

The action-perception cycle (Fig. 43) provides a principal motivation for the application of schema theory [95]. Sensor-driven expectations provide the plans (schemas) for appropriate motor action, which when undertaken provide new sensory data that is fed back into the system to provide new expectations. This cycle of cognition (the altering of the internal world model), direction (selection of appropriate motor behaviors), and action (the production of environmental changes and resultant availability of new sensory data) is central to the way in which schemas must interact with the world.

Most significantly, perception should be viewed as action-oriented. There is no need to process all available sensor data, only that data which is pertinent to the task at hand. The question for the roboticist would be: how do we select from the wealth of sensor data available that which is relevant? By specifying schemas, each individual component of the overall task can make its demands known to the sensory subsystem, and thus guide the focus of attention mechanisms and limited sensory processing that is available.

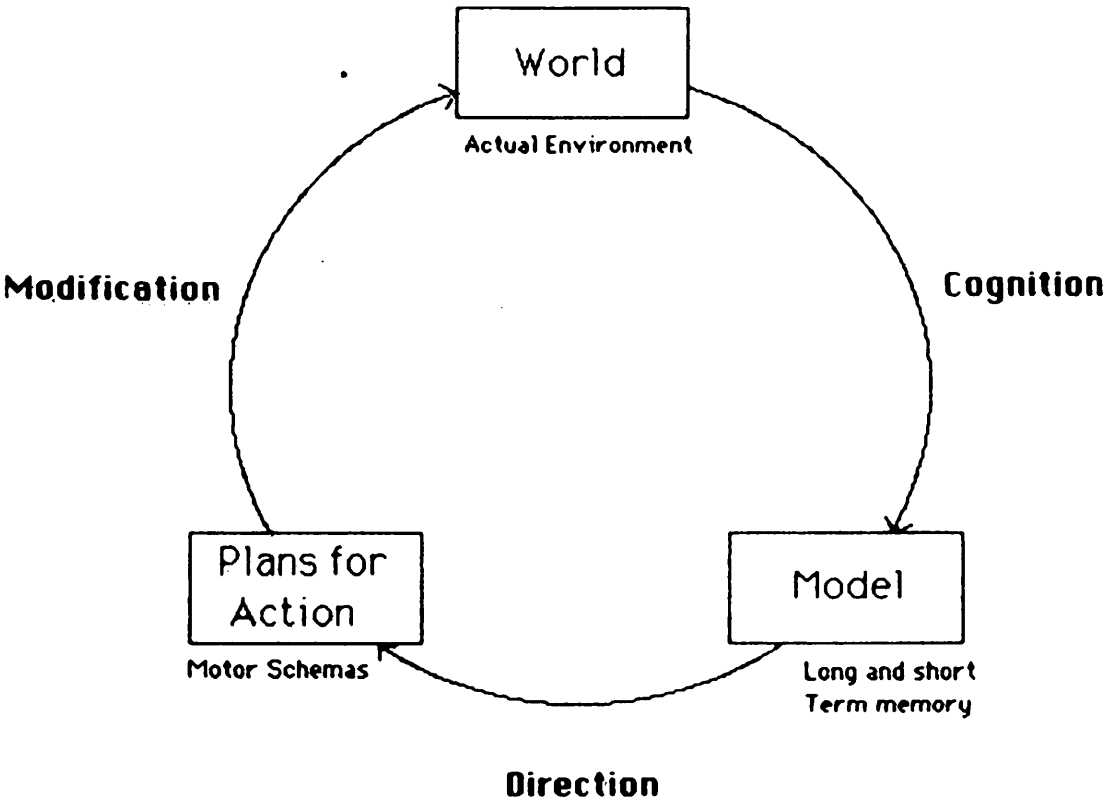


Figure 43: Action-perception cycle

Guided by Arbib's work [6,5] in the study of the frog and its machine analog *Rana Computatrix*, the frog prey selection mechanism serves as a basis for analysis. In particular, Arbib and House [8] have developed a model for worm acquisition by the frog in an obstacle-cluttered environment (a spaced fence - Fig. 44). Although Arbib and House describe two models to account for the behavior of the frog, the second is the most readily applicable to the mobile robot's domain (the first model is based on visual orientation). In their work, they describe primitive vector fields (Fig. 45): a prey-attractant field, a barrier-repellent field, and a field for the animal itself. These fields, when combined, yield a model of behavior (Fig. 46) that is consistent with experimental observations of the frog.

In the mobile robot system described below, analogs of these fields are used (prey-attractant ⇒ **move-to-goal**, barrier-repellent ⇒ **avoid-static-obstacle**). Additionally, new fields are added to describe additional motor tasks (**stay-on-path**, **avoid-moving-obstacle**, etc.)

This model, in conjunction with expectation-driven sensing, provides a basic correlate with the functioning of the brain (albeit the frog brain). Although the brain has been handling visually guided detours since time immemorial, the benefits of using a neuroscience model would wane if it proved impractical for a mobile robot. In the sections following, the practicality of this approach is demonstrated, especially regarding the decomposition of the task to a form which is readily adaptable to distributed processing. This is essential if the real-time demands of mobile robot environmental interaction are to be met.

§1.2 *Robotics*

Schema theory as applied to robotics has almost as many different definitions as there are developers. In the realm of robotic manipulators, Lyons' schemas [78] and Geschke's servo processes [50], (a schema analog), are used as approaches to task level control. Overton [101] has described the use of motor schemas in the assembly domain. The UMASS VISIONS group, guided by Hanson and Riseman, has applied perceptual schemas to the interpretation of natural scenes; Weymouth's thesis [140] and Draper's paper [41] are prime examples of this work. Although AuRA will, in the future, include perceptual schemas running in the context of the VISIONS system, perceptual schemas

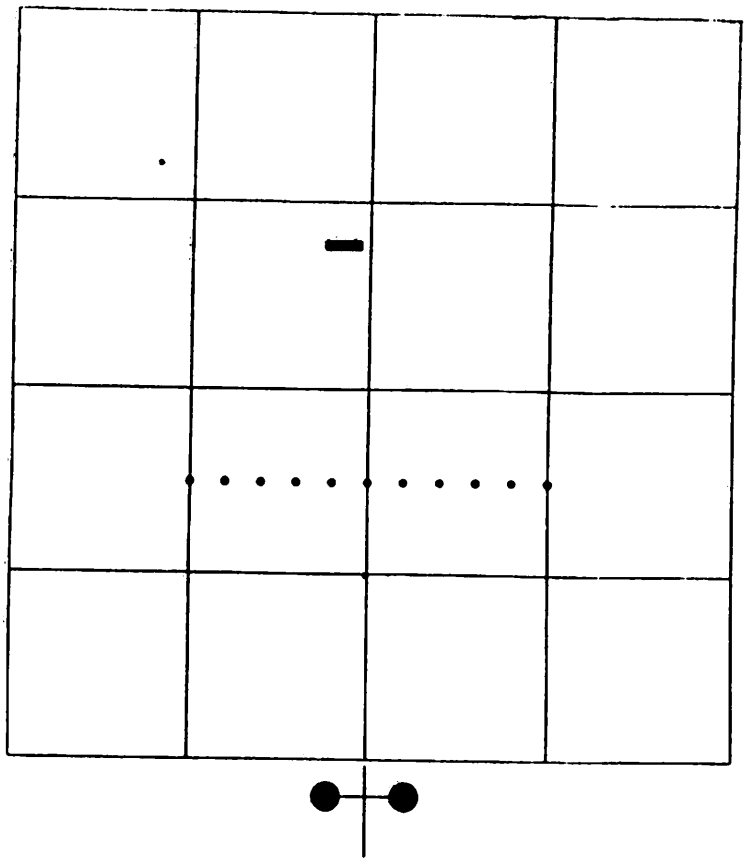


Figure 44: A depiction of a frog prey-selection scenario.
The two large blackened circles at the bottom of the figure denote the frog's eyes, the smaller circles are fence-posts, and the darkened rectangle a supply of worms.
(reprinted from [8] with permission)

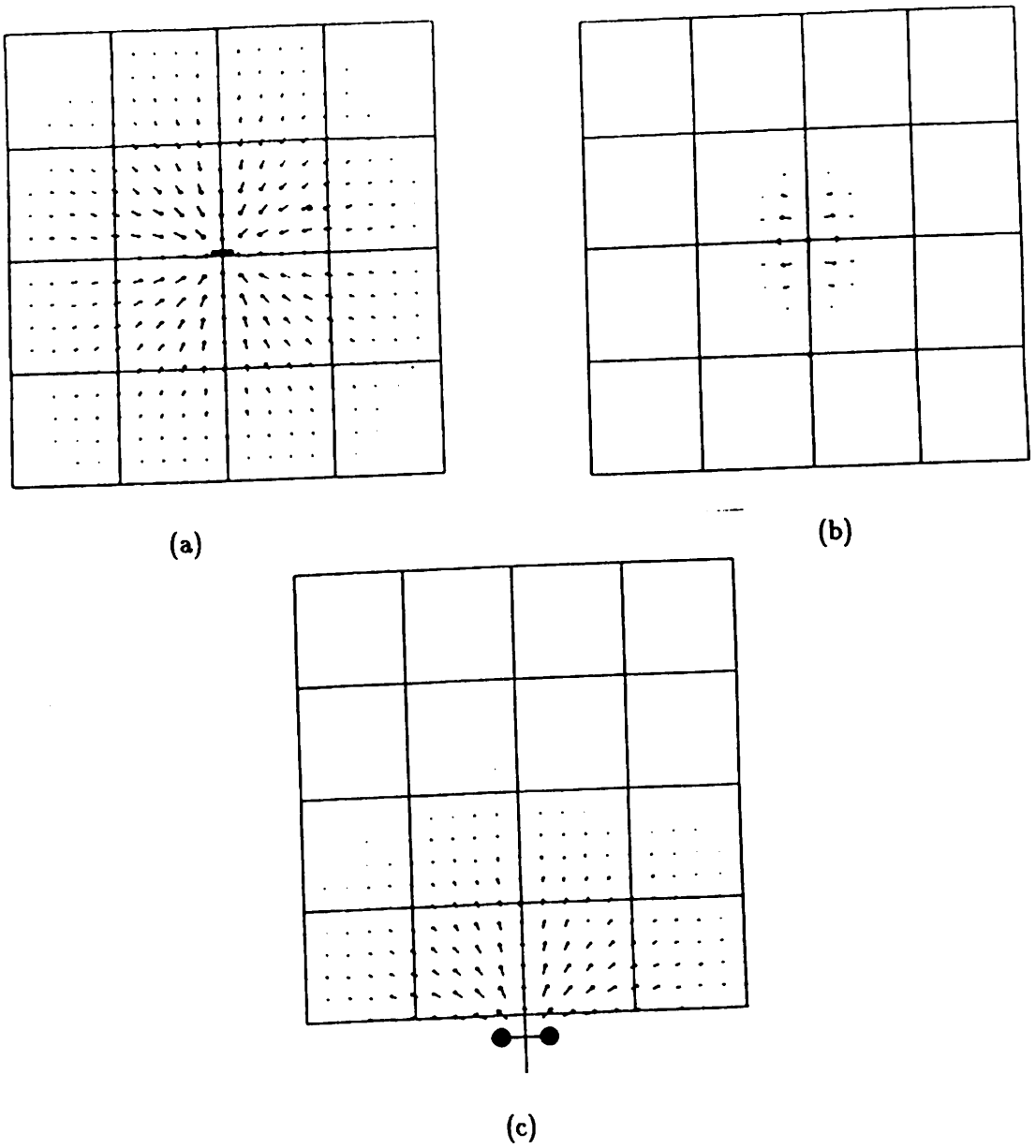


Figure 45: Primitive vector fields associated with Figure 44.

a) Prey-attractant field.

b) Barrier repellent field.

c) Frog representation field.

(reprinted from [8] with permission)

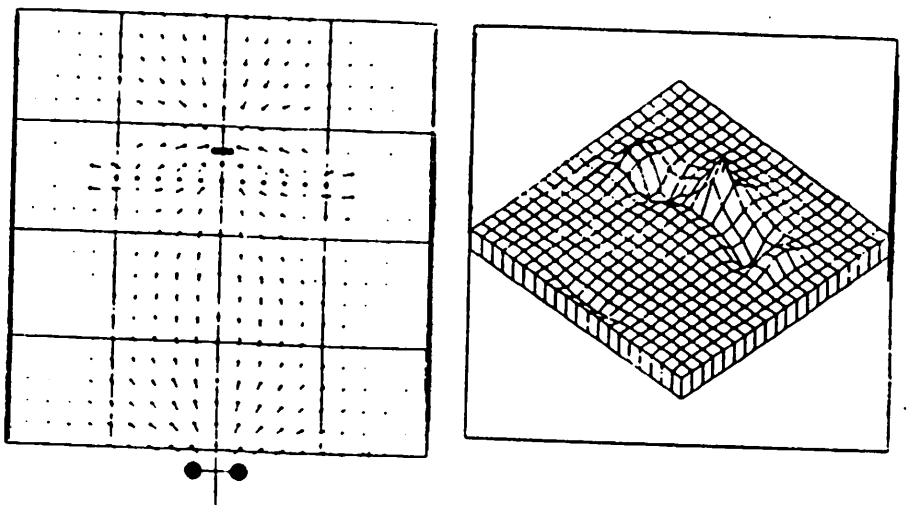


Figure 46: Resultant frog-prey selection vector field.
(reprinted from [8] with permission)

as they appear in the VISIONS system are not a principal concern of this chapter.

One of the simplest and most straightforward definitions for a schema is "a generic specification of a computing agent" [78]. This definition fits well with the concept of a behavior (an individual's response to its environment) – each schema represents a generic behavior. Schema-based control systems are significantly more than a collection of frames or templates for behavior, however. The way in which they are set into action and interact immediately distinguish them from simpler representational forms. The instantiations of these generic schemas provide the potential actions for the control of the robot. A schema instantiation (SI) is created when a copy of a generic schema is parameterized and activated as a computing agent.

Lyons further defines a motor schema as a control system or motor program which describes a task. Overton [101] describes a motor schema as "a control system which continually monitors feedback from the system it controls to determine the appropriate pattern of action for achieving the motor schema's goals (these will, in general, be subgoals within some higher-level coordinated control program)". This more constrained definition is also in accord with the system described below. Sensory perception provides the feedback to affect individual instantiations of motor schemas, each SI thus providing an appropriate behavior which collectively determine the overall system's behavior. Some other definitions for motor schema include an "interaction plan" [7] or "unit of motor behavior" [77].

Other work in the path planning domain, although not schema based, bears a resemblance to the schema control system. Brooks [24] uses a planning system with a "horizontal decomposition" which effectively emulates multiple behaviors. Although related, there is still a rigid layering present which distinguishes it from a schema-based approach. Payton [107] describes a multi-behavior approach for reflexive control of an autonomous vehicle. The association of virtual sensors with a selected set of reflexive behaviors bears a similarity to the schema-based approach. An arbitrary choice of behavior, however, based on a priority system, is made during execution, without provision for a mechanism to combine the results of concurrent behaviors. Kadonoff et al [60] also incorporate multiple behaviors for the control of a mobile robot and similarly arbitrate between these behaviors, proposing a production system for arbitrating competitive strategies and the use of an optimal filter for the treatment of complementary strategies.

The schema system described below is strongly influenced by Krogh's [68] generalized potential fields approach and to a lesser degree by Lyons' [79] tagged potential fields. It bears a superficial resemblance to the integrated path planning and dynamic steering control system described by Krogh and Thorpe [69]. Potential fields are used, in each case, to produce the steering commands for a mobile robot. A major distinction between their system and our schema model lies in the tracking of the individual obstacles (individual SIs for each obstacle, important for the treatment of uncertainty) and the incorporation of additional behaviors such as road following and treatment of moving obstacles. The state of each obstacle's SI is dynamically altered by newly acquired sensory information. The potential functions for each SI reflect the measured uncertainty associated with the perception of each object. The schema approach is not limited to obstacle avoidance, but is versatile enough for road following, object tracking and other behavioral patterns.

§2. Approach

Motor schemas, when instantiated, must drive the robot to interact with its environment. On the highest level, this will be to satisfy a goal developed within the planning system; on the lowest level, to produce specific translations and rotations of the robot vehicle. The schema system enables the software designer to deal with conceptual structures that are easy to comprehend and handle. The task of robot programming is fundamentally simplified through the use of a divide and conquer strategy.

§2.1 *Schema-based Navigation*

AuRA's pilot is charged with implementing leg-by-leg the piecewise linear path developed by the navigator. To do so, the pilot chooses from a repertoire of available sensing strategies and motor behaviors (schemas) and passes them to the motor schema manager for instantiation. Distributed control and low-level planning occur within the confines of the motor schema manager during its attempt to satisfy the navigational requirements. As the robot proceeds, the cartographer, using sensor data, builds up a model of the perceived world in short-term memory. If the actual path deviates too greatly from the path initially specified by the navigator due to the presence of unmodeled obstacles or positional errors, the navigator will be reinvoked and a new global path computed.

If the deviations are within acceptable limits, (as determined by higher levels in the planning hierarchy), the pilot and motor schema manager will, in a coordinated effort, attempt to bypass the obstacle, follow the path, or cope with other problems as they arise. Additionally, the problem of robot localization is constantly addressed through the monitoring of short-term memory and appropriate **find-landmark** schemas. Multiple concurrent behaviors (schemas) may be present during any leg, for example:

- **Stay-on-path** (a sidewalk or a hall)
- **Avoid-static-obstacles** (parked cars, trees, etc.)
- **Avoid-moving-obstacles** (people, moving vehicles, etc.)
- **Find-intersection** (to determine end of path)
- **Find-landmark**(building) (for localization)

The first three are examples of motor schemas, the last two perceptual schemas. To provide the correct behavior, a subset of perceptual schemas must be associated with each motor schema. For example, in order to stay on the sidewalk, a **find-terrain**(sidewalk) perceptual schema must be instantiated to provide the necessary data for the **stay-on-path** motor schema to operate. If the uncertainty in the actual location of the sidewalk can be determined, the SI's associated velocity field, applying pressure to remain on the sidewalk, will reflect this uncertainty measure. The same holds for obstacle avoidance: if a perceptual schema for obstacle detection returns the position of a suspected obstacle and the relative certainty of its existence, the actual avoidance maneuvering will depend not only on whether an obstacle is detected but also on how certain we are that it exists. Differing strategies within each SI will determine how to manage the perceptual uncertainty. If an event is potentially fatal, even large amounts of perceptual uncertainty would produce motor behavior, but erring in the direction of safety.

An example illustrating the relationship between motor schemas and perceptual certainty follows. The robot is moving across a field in a particular direction (**move-ahead** schema). The **find-obstacle** schema is constantly on the look-out for possible obstacles within a subwindow of the video image (windowed by the direction and velocity of the robot). When an event occurs, (e.g. a region segmentation algorithm detects an area that is distinct from the surrounding backdrop or an interest operator locates a high-interest

point in the direction of the robot's motion), the **find-obstacle** schema spawns off an associated perceptual schema (**static-obstacle** SI) for that portion of the image. It is now the **static-obstacle** SI's responsibility to continuously monitor that region. Any other events that occur elsewhere in the image spawn off separate **static-obstacle** SIs. Additionally an **avoid-static-obstacle** SI motor schema is created for each detected potential obstacle.

The motor schema SI hibernates waiting for notification that the perceptual schema is sufficiently confident in the obstacle's existence to warrant motor action. If the perceptual schema proves to be a phantom (e.g. shadow) and not an obstacle at all, both the perceptual and related motor SIs are deinstantiated before producing any motor action. On the other hand, if the perceptual SI's confidence (activation level) exceeds the motor SI's threshold for action, the motor schema starts producing a repulsive field surrounding the obstacle.[1] The sphere of influence (spatial extent of repulsive forces) and the intensity of repulsion of the obstacle are affected by the distance from the robot and the obstacle's perceptual certainty. Eventually, when the robot moves beyond the perceptual range of the obstacle, both the motor and perceptual SIs are deinstantiated. In summary, when obstacles are detected with sufficient certainty, the motor schema associated with a particular obstacle (its SI) starts to produce a force tending to move the robot away from the object. Fig. 47a shows a typical repulsive field for an **avoid-static-obstacle** SI. The control of the priorities of the behaviors, (e.g. when is it more important to follow the sidewalk than to avoid uncertain but possible obstacles) is partially dependent on the uncertainty associated with the obstacle's representation. Other isolated motor schema velocity fields are shown in Fig. 47b-d. Various combinations of motor schemas are illustrated in Fig. 48. Recognize the fact that although the entire field is expensive to compute, each active motor SI need only determine the velocity vector at the robot's current location relative to the environmental object, making the computation very rapid. Further, as the SIs are activated in parallel, even better performance is attainable.

Multiple instantiations of a single schema are frequently the case. Each generic "skeleton" is parameterized when instantiated. Consequently, it is entirely possible that two different instantiations of the same generic schema produce significantly different fields

[1]The obstacle is first grown in a configuration space manner [76] to enable the robot to be treated henceforth as a point for path planning purposes.

Figure 47: Isolated motor schema SI vector fields

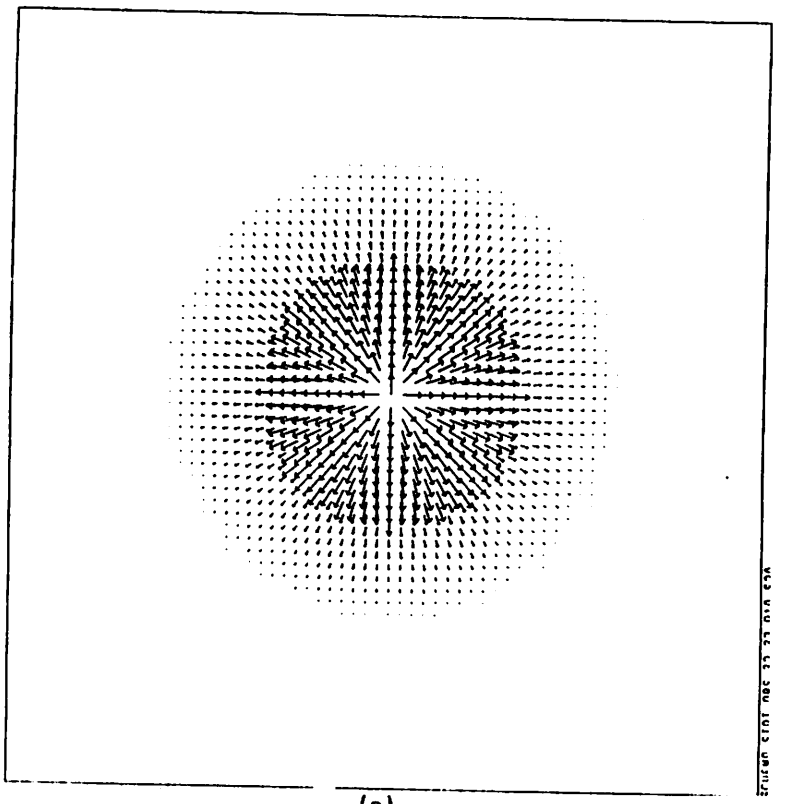
a) Avoid-static-obstacle.

b) Move-to-goal.

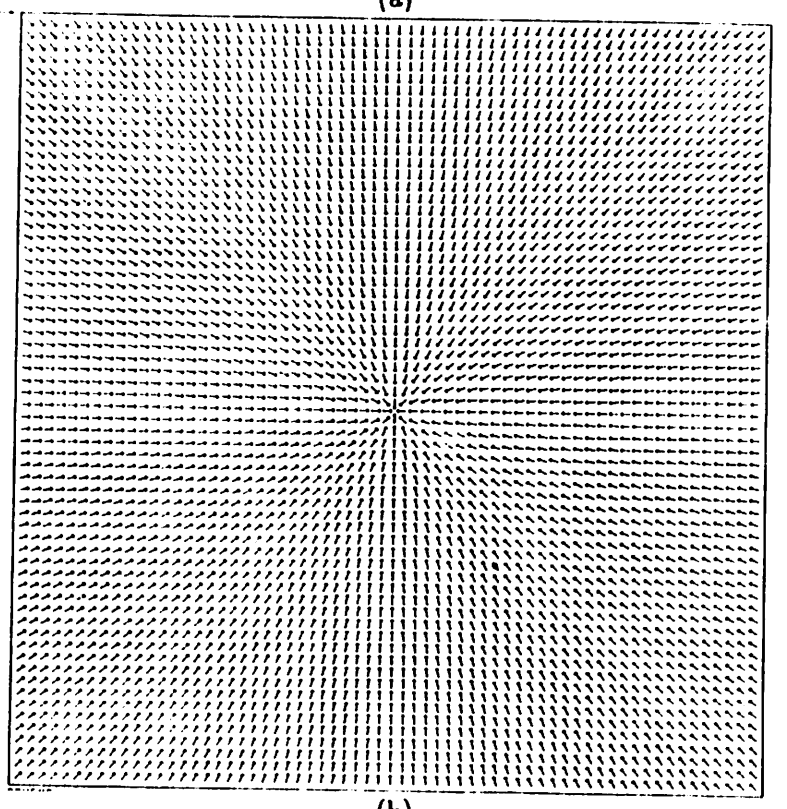
c) Move-ahead.

d) Stay-on-path.

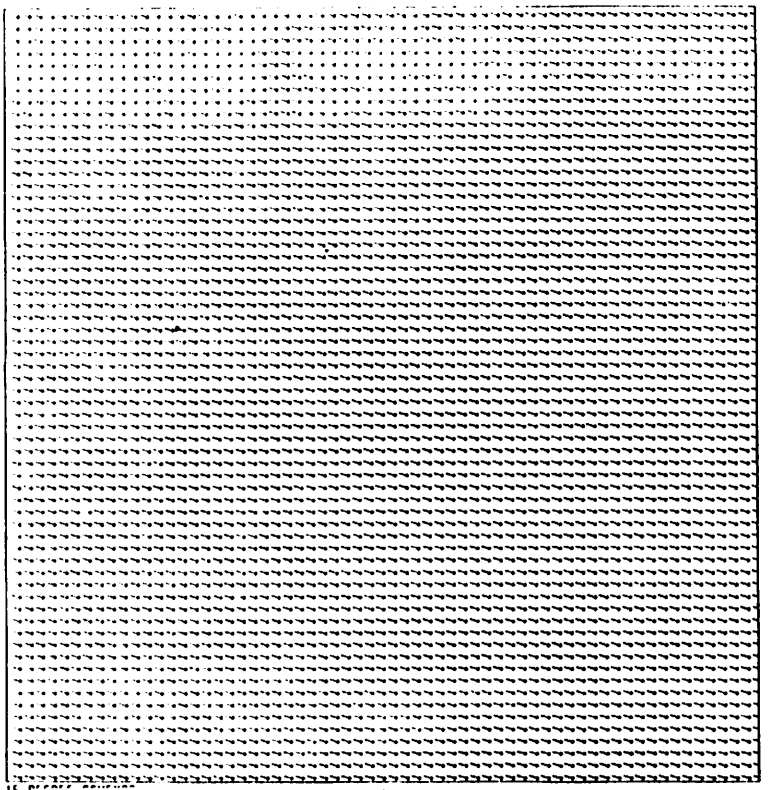
(Figure continued on following page).



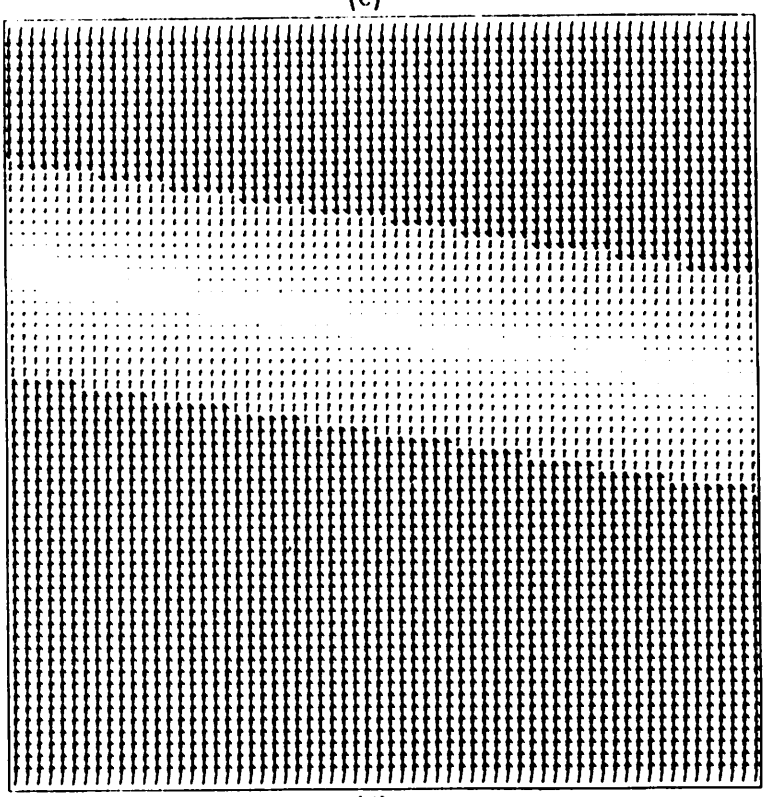
(a)



(b)



(c)



(d)

Figure 47 continued.

Figure 48: Several combined motor schemas

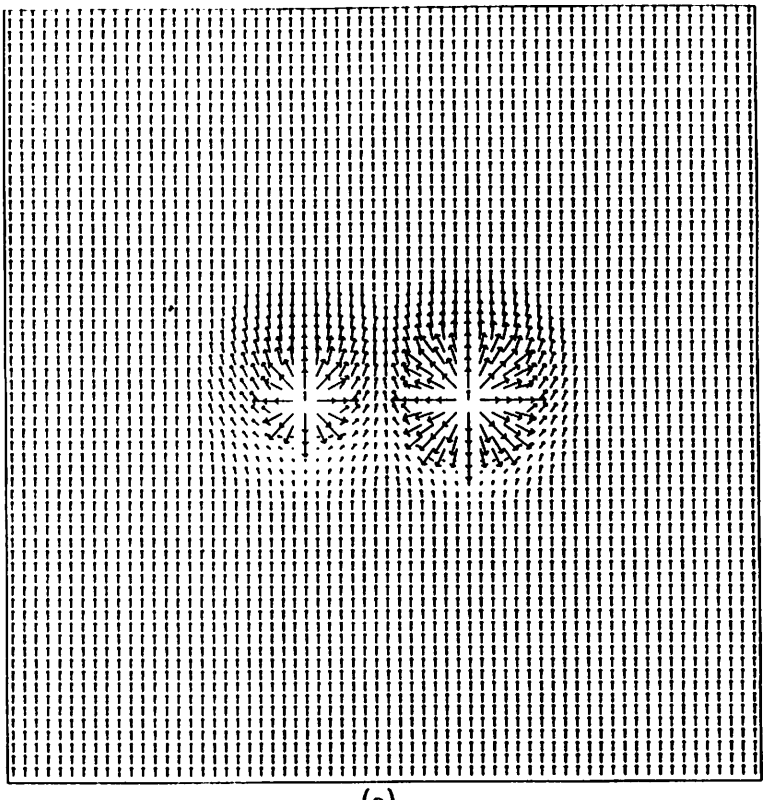
a) Move-ahead SI + two avoid-static-obstacle SIs.

b) Move-ahead SI + stay-on-path SI.

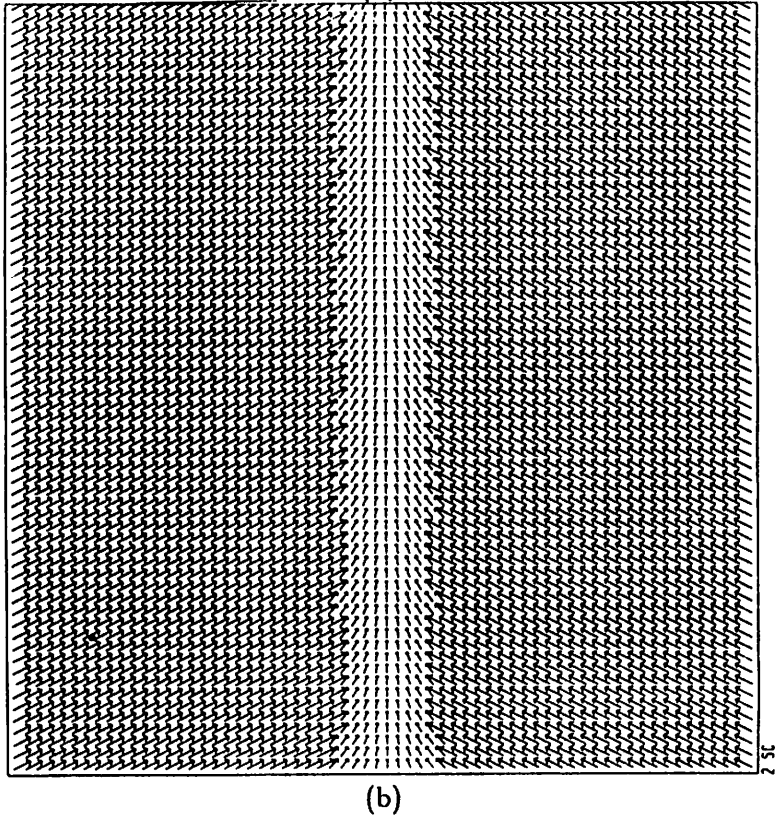
c) Move-ahead SI + stay-on-path SI + one avoid-static-obstacle SI.

d) Move-to-goal SI + stay-on-path SI + two avoid-static-obstacle SIs.

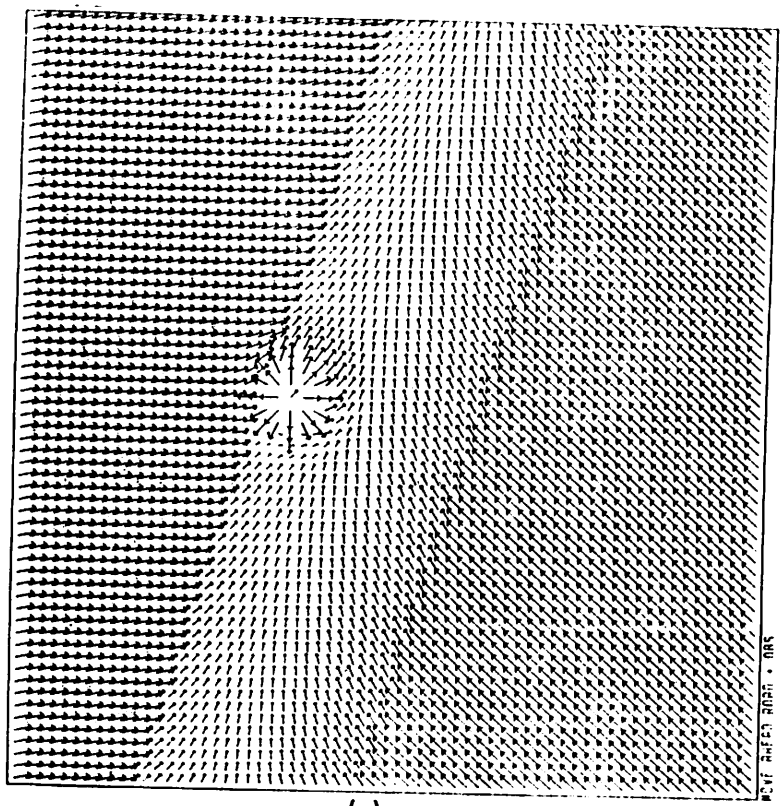
(Figure continued on following page).



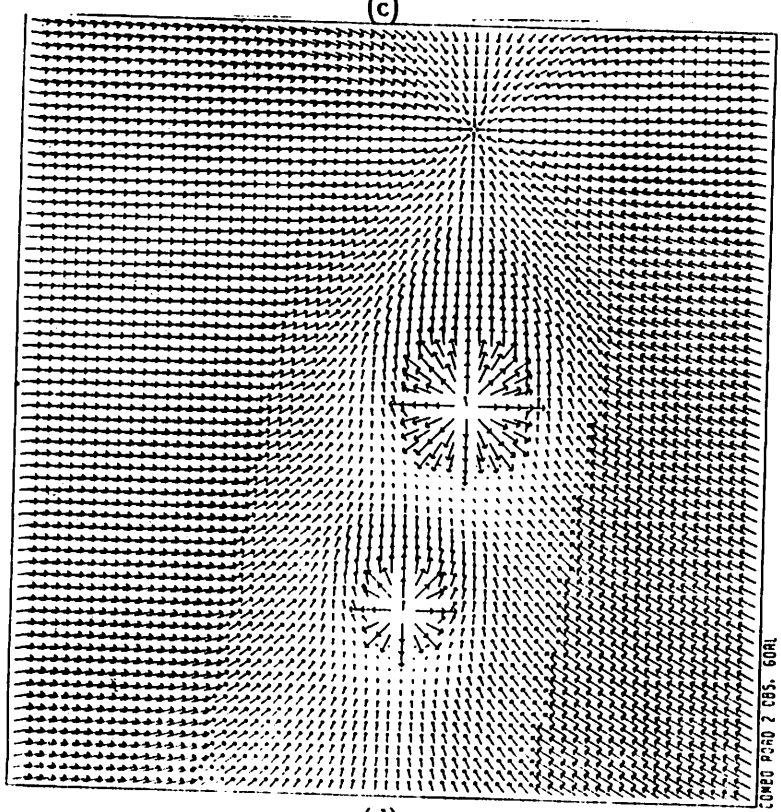
(a)



(b)



(c)



(d)

Figure 48 continued.

under similar sensory conditions (as in the case of path following for a sidewalk or hall discussed above). The parameters set at instantiation may depend on the sensory events that triggered the instantiation or from information retrieved by the pilot from LTM.

If each schema functions independently of each other, how can any semblance of realistic and consistent behavior be achieved? Two components are required to satisfactorily answer this question. First, a combination mechanism must be applied to all the SI-produced vectors. The result is then used to provide the necessary velocity changes to the robot. The simplest approach is vector addition. By having each motor SI create a normalized velocity vector, a single **move-robot** schema monitors the posted data for each SI, adds them together, makes certain it is within acceptable bounds and then transmits it to the low-level robot control system. In essence, the specific velocity and direction for the robot can be determined at any point in time by summing the output vectors of all the active individual SIs. As each motor SI is a distributed computing agent, preferably operating on separate processors on a parallel machine, and needs only to compute the velocity at the point the robot is currently located and a few points in its projected track (and not the entire velocity field), real-time operation is within reach.

The second component of the response to the question posed in the previous paragraph is communication. Potential fields can have problems with dead spots or plateaus where the robot can become stranded. By allowing communication mechanisms between the SIs, the forces of conflicting actions can be reconciled. Lyons [78] proposes message passing between ports on one SI and connected ports on another SI as a schema communication mechanism. Alternatively, a blackboard mechanism is used in the VISIONS Schema Shell (discussed below). In either case, communication mechanisms can solve problems that might otherwise prove intractable. An example to illustrate this point follows.

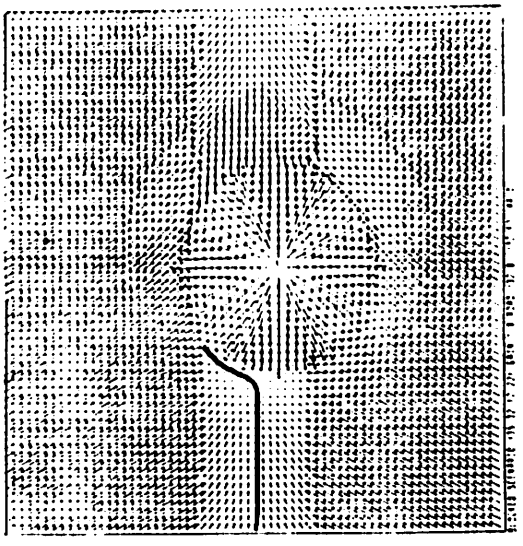
The robot is instructed to move in a particular direction, stay on the sidewalk and avoid static obstacles. Suppose that the sidewalk is completely blocked by an obstacle; eventually the velocity would drop to 0 and the robot would stop (Fig. 49a). The fact that the vehicle has stopped is detected by the **stay-on-path** SI through interschema communication with the **move-robot** SI (the **move-robot** SI combines the individual motor SIs and communicates the results to the low-level motor control system). The **stay-on-path** SI, when created for this particular instance, was instructed to yield if an obstacle blocks the path. The **stay-on-path** motor schema reduces its field (Fig. 49b)

and allows the robot to wander off the sidewalk thus circumnavigating the obstacle. As soon as the direction of the force produced by the offending obstacle indicates it has been successfully passed, the **stay-on-path** field returns to its original state forcing the robot back on the path (Fig. 49c).

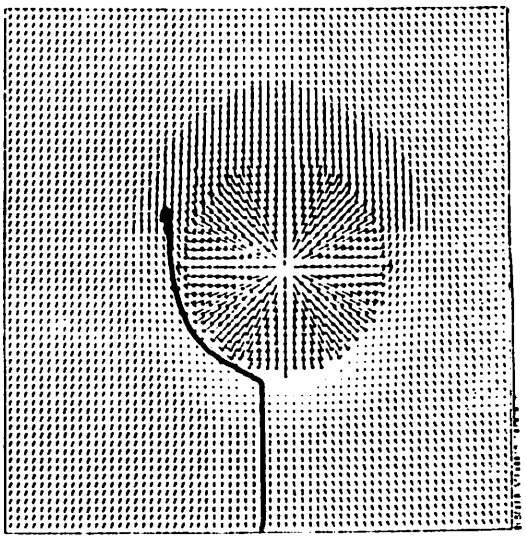
Suppose, however, the **stay-on-path** SI was instantiated for a hall. Then, under no circumstances, would the force field associated with the **stay-on-path** SI be reduced or else the robot would crash into the wall. The robot would instead stop, and signal for the navigator (higher level component of the planner) to be reinvoked and produce an alternate global path that avoids the newly discovered blocked passageway. These communication pathways are specified within the schema structures themselves.

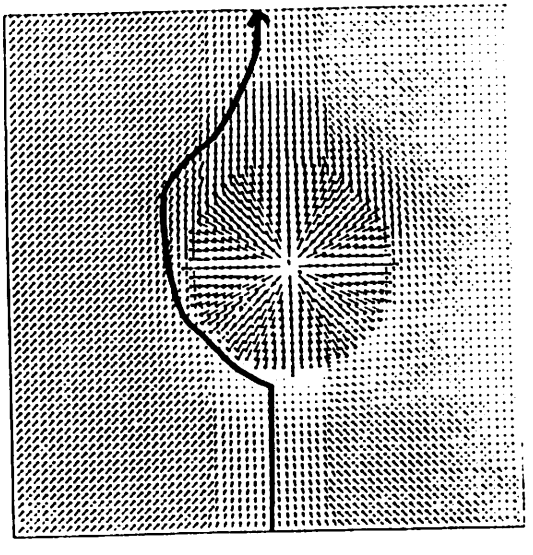
It is entirely possible that the trajectory of the robot can be computed for a small distance rather than just its instantaneous velocity at the immediate location. Each motor schema would have to interact with the **move-robot** SI, using the vector summation output to determine the position of the robot relative to its perceptions for the next time step. This is of particular significance if the sensor sampling rates are low. Trajectories can be determined that reflect the robot's perceptions at a given point in time, rather than just reacting to current sensing. This is of value in determining when to activate other schemas in anticipation of special problems or needs. Care must be taken in highly dynamic environments (e.g. moving objects) so that the plans developed do not ignore changes in the world that are evidenced only through perception.

Another approach explored is the addition of a background stochastic **noise** schema. This SI produces a low-magnitude random direction velocity vector that changes at random time intervals, but persists sufficiently long to produce a change in the robot's position if the robot's velocity was otherwise zero. Its role is to perturb the velocity of the robot slightly, removing the robot from undesirable equilibrium points, which arise when the active motor SIs counterbalance each other. The behavior produced by this schema corresponds to the "wander" layer in Brooks' horizontally layered architecture [24]. This schema would serve to remove the robot from any potential field plateaus or ridges upon which the robot becomes perched (e.g. from a direct approach to an obstacle - Fig. 50). Other traps common to potential field approaches (e.g. box canyons) can be handled by establishing hard time deadlines for goal attainment. If these deadlines are violated, the pilot would be reinvoked to establish an alternate route using STM data



(a)





(b) (c)

Figure 49: Blocked sidewalk scenario

a) Robot stops in dead spot due to pressure to both remain on sidewalk and avoid the obstacle.

b) Gain lowered on **stay-on-path** SI allows robot to bypass obstacle.

c) Once obstacle is passed **stay-on-path** SI returns to normal, forcing robot back onto the sidewalk.

gathered by the cartographer during the route traversal.

It is worth noting that a single sensory event may have two or more SIs associated with it. For example: if the robot is looking for a mailbox to get its bearings for localization purposes, a perceptual schema for localization (**find-landmark**) would process portions of the image that are likely to be mailboxes. If the mailbox happens to be in the path of the vehicle, a concurrent **avoid-static-obstacle** SI would view that object not as a mailbox but rather as an obstacle, and would be concerned only with avoiding a collision with it. This "divide and conquer" approach based on action-oriented perception simplifies programming and overall system design. A more complex scenario appropriate for a mobile robot appears in Fig. 51.

§3. Implementation Strategy

The implementation tool chosen for the motor schema system is the Schema Shell [40,41,42], a system developed by the VISIONS group at UMASS for use in the perceptual schema analysis of natural scenes. It currently runs on a Texas Instruments Explorer workstation and is tied to the Computer Science Department's VAXen over Chaosnet. The schema communication mechanism is blackboard-based. The Schema Shell system is expected to be moved to the department's newly acquired Sequent parallel processor. Although the Explorer only simulates distributed processing, everything points towards the availability of a truly distributed environment in the not too distant future.

The schemas themselves (in the Schema Shell) consist of a schema template and multiple strategies. Associated with each instantiated schema is an object hypothesis maintenance (OHM) strategy. This part of the SI monitors the blackboard for new events (e.g. sensory data) that would produce changes in the SI's posted output. Other strategy components for each SI handle conflict resolution, cooperative enhancement, initialization and other relevant factors. Not all strategies are necessary or desirable for all schemas. Figure 52 shows a typical generic motor schema cast in the Schema Shell format.

Pocock at the UMASS Laboratory for Perceptual Robotics is developing an alternative schema-based robot control system [107] based on Lyons' port automata formalization [78] of schema theory. As it seriously addresses real-time distributed scheduling, it may prove

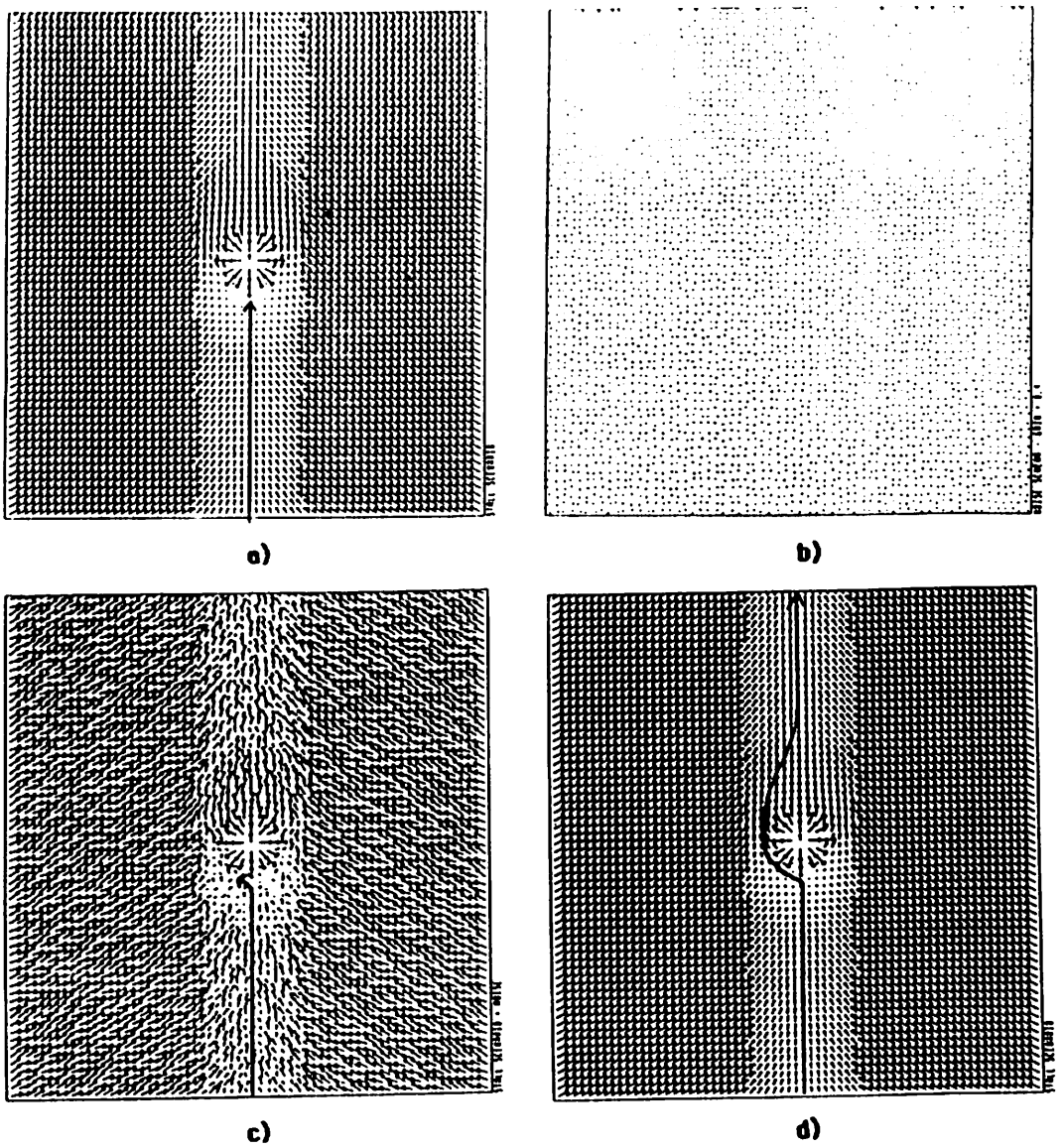


Figure 50: Stall scenario

a) If the robot approaches an obstacle exactly head-on, it is possible for it to become stalled.

b) **Noise** SI provides small magnitude random direction vector to push robot off of the tiny plateau.

c) **Noise** schema added to a).

d) The robot can now successfully bypass the obstacle. The **noise** SI is then deinstantiated.

Pilot issues instructions to follow sidewalk while avoiding obstacles.
Continue approximately 200 ft on sidewalk then turn right at lamppost onto intersection (first encountered). Watch for landmarks (mailbox on left, building edge on right) for localization.

Motor Schemas instantiated by pilot:

- Stay_on(identify_terrain(sidewalk,60%)) (assumes sidewalk is ahead to start).
- Move_ahead(210,start_heading) (nominal distance plus some slop).
- Avoid_static_obstacles(15,identify_obstacle(robot_heading,nil,70%))
  Start maneuvering around when within 15 feet. Nil denotes static obstacle. 70% is threshold for motor action.
- Avoid dynamic_obstacles(20,identify obstacle(robot heading,-robot heading,40%))
  Start evasive action when head on approach within 20 feet.
- Follow_dynamic_obstacle(8,start_heading,identify_obstacle(True,start_heading,95%))
  When an obstacle is moving in the correct direction, within 8 feet of the robot , follow it (regardless of robot's current heading).
- Avoid dynamic obstacles(5,identify obstacle(robot heading,True,40%))
  Start evasive action when within 5 feet for any dynamic obstacle (includes crossing dynamic obstacles).
- Turn when(find landmark(lamppost 1+5ft,90,90%)) - right 90 degrees.
- Turn when(find landmark(intersection 3a,90,90%)) - right 90 degrees.
- Localize(find landmark(mailbox 7,90%)).
- Localize(find landmark(building 2a.edge3, 85%)) - prune spatial uncertainty map on landmark recognition.
- Stop_when(not (sidewalk_1 = identify_terrain(ahead,90%))) - missed turn.

  (Percentages denote thresholds for motor action).

Perceptual Schemas instantiated by pilot:

- Identify_obstacle(robot_heading,obstacle_heading,certainty) Only detects obstacles in the way of the robot (distinct from landmarks). Robot heading and obstacle heading are directional filters. Certainty is threshold for identification. Returns obstacle position and type. 1 identify_obstacle spawned for each strategy type above.
  obstacle - generic - many spawned for each identify obstacle.
  Returns certainty. Tracks motion over time.
  types: static_obstacle and dynamic_obstacle (predicates)
  obstacle heading (nil if static), speed (0 if static).
- Find landmark(LTM model,certainty) - 1 spawned per landmark.
  Assumes robot's current position for observation is available in global coordinates (spatial uncertainty map). Certainty is threshold for recognition. Returns landmark location. Landmark is not necessarily in direct track of robot, could be anywhere.
  landmark(LTM_model) - many/landmark spawned off
  Returns certainty.
- Identify_terrain(position,certainty) - Returns terrain type.

At end of maneuver, deinstantiate all obstacle schemas.

Figure 51: Example mobile robot schema scenario

```
;-------------------------------------------------------------------------
;                        MOVE-AHEAD Motor Schema
;       ->
;       ->        x-axis is 0; frame of reference is robot's initial heading afterturn;
;       ->
;       ->                          .

 (make-motor-schema
       :name "MOVE-AHEAD"
       :default-argument-list (list '("heading" 0))
       :body
       (progn
              (de-schema  move-ahead (original-heading current-heading move-ahead-force-table))

              (de-strategy move-ahead ohm ()
                (progn
                  (call-strategy 'move-ahead 'init)
                  (write-to-window "move-ahead ohm")
         (loop
                  (write-to-window (setf #!current-heading
                (read-or-wait '(current-orientation-messagep robot-position))))

                    (write-to-window (write-to-blackboard
                      (list "move-ahead" (look-up-move-ahead-force #!current-heading) (time-stamp))
                            'vector-section)) ; write resultant force to vector section
                    )
         ))

;             strategies for move-ahead

              (de-strategy move-ahead init()
                (build-move-ahead-force-table)                          ; initialize lookup table
                (setf #!original-heading (read-or-wait '(orientation-messagep robot-position)))
                (write-to-window "move-ahead init done")
              )


       ; conflict in case of reverse direction- send message-to-mover-to-terminate
       ; contains list of contradictory motions or evidence

       ; support in case of confirmed direction
       ; support contains list of related schemas use to confirm hypothesis
       ; e.g. to stop-when's
            )  ); End make-motor-schema
```

Figure 52: Example move-ahead schema as implemented in the Schema Shell

a useful tool for mobile robot research when completed. At that time, its relationship to the VISIONS Schema Shell will be considered.

§4. Simulation

Simulations were run on a VAX 750 using the following motor schemas: **stay-on-path, move-ahead, move-to-goal, avoid-static-obstacle**. Each simulation run (Figs. 53–54) shows the sequence of resultant overall force fields based on perceived entities. These entities include path borders, goals, and obstacles. The grid size is 64 units by 64 units and the sensory sampling update time (once per second) is based on a nominal velocity of 1 unit/second. The maximum vector length for display purposes has been set to 2.0 normal velocity units. The actual vector magnitude within the obstacles is set to infinity (a discrete approximation). All obstacles are currently modeled as circles (as in Moravec's tangent space [93]). The field equations for several of the motor schemas appear below.

The field equations for both the **avoid-static-obstacle** and **stay-on-path** schemas are linear. An example showing the velocity produced by an obstacle (O) is given below:

Avoid-obstacle

$O_{magnitude} =$

$$\begin{cases} 0 \ for \ \ d > S \\ \frac{S-d}{S-R} * G for \ \ R < d \leq S \\ \infty \ \ for \ \ d \leq R \end{cases}$$

where:

S = Sphere of Influence (radial extent of force from the center of the obstacle)

R = Radius of obstacle

G = Gain

d = Distance of robot to center of obstacle

$O_{direction}$ = along a line from robot to center of obstacle moving away from obstacle

More complex equations could be used (e.g. cubic as in [69]) but were deemed unnecessary in these early stages of the research.

Stay-on-path

$V_{magnitude}$ =

$$\begin{cases} \text{P for } d > (W/2) \\ \frac{d}{(W/2)} * G \text{ for } \quad d \leq \frac{W}{2} \end{cases}$$

where:

W = Width of path

P = Off path gain

G = On path gain

d = Distance of robot to center of path

$V_{direction}$ = along a line from robot to center of path heading toward centerline

Move-ahead

$V_{magnitude}$ = fixed gain value

$V_{direction}$ = in specified compass direction

Move-to-goal

$V_{magnitude}$ = fixed gain value

$V_{direction}$ = in direction towards perceived goal

In some of these simulations the uncertainty in perception was allowed to decrease the sphere of influence of an obstacle. When a threshold was exceeded (50% certain), the sphere of influence of the obstacle started increasing linearly as the certainty increased up to its maximum allowable value. Another alternative is to increase the gain on the

obstacle proportionately with the increase in certainty (up to its maximum).

Figure 53a illustrates the robot's course on a sidewalk moving towards a goal. The course is studded with 8 obstacles, only 7 of which are perceptible to the robot during its journey (Fig. 53b). Note how the vector fields change as the robot encounters more obstacles along the way (Figs. 53c-e). When it has successfully navigated obstacles and they have moved out of range, their representation is dropped from short-term memory and the associated motor schema is deinstantiated (Fig. 53e). The robot stays on the path for the complete course successfully achieving its goal while avoiding each obstacle. An expanded version could update long-term memory as a result of experience, thus incorporating learning.

Figure 54 shows the robot's path to a specified goal through a field of 9 obstacles. This simulation prevents perceived objects that have too great an uncertainty from producing a repulsive field. In this case, the uncertainty increases with the distance from the obstacle. A decrease in uncertainty results in an increase in the sphere of influence of the obstacle. Consequently, the uncertainties and the resultant obstacle fields change as the robot moves through the course. Figures 54b-f use a **move-to-goal** SI while Figs. 54g-h use a **move-ahead** SI. Actually the robot would operate under the control of a **move-ahead** SI until the goal is perceived (assuming dead-reckoning or inertial guidance is not used). At the moment of goal perception, the **move-ahead** SI would be deinstantiated and a **move-to-goal** SI created in its stead.

§5. Motor subsystem

AuRA's motor subsystem accepts the output from the motor schema manager's **move-robot** SI and produces the required velocity for the vehicle. Little has been said thus far about the vehicle interface and other components of the motor subsystem other than stating that this component of AuRA is largely vehicle dependent. In the case of the UMASS Denning Research Vehicle (DRV), the motor controllers and motors themselves have been provided by the manufacturer (Denning Mobile Robotics Inc. of Woburn, Mass.). The interested reader is referred to the Denning documentation set [38] for the details of the control circuitry.

Communication with the vehicle is another story. The robot runs a terminal emulation

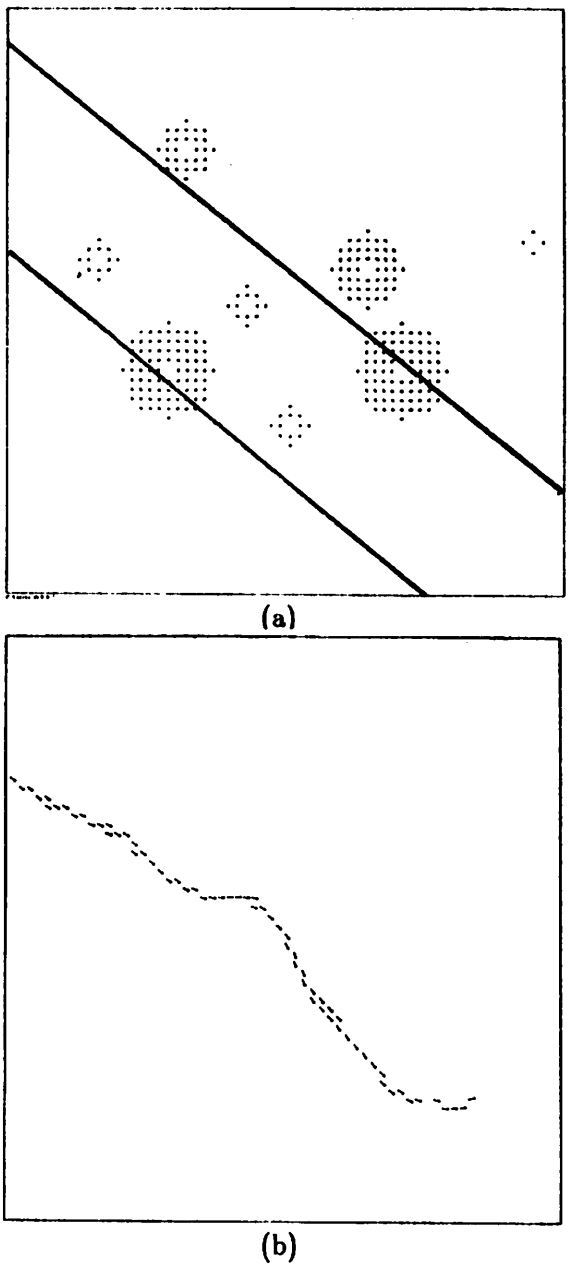


Figure 53: Schema simulation run

This simulation shows 7 **avoid-static-obstacle** SIs, a **move-to-goal** SI, and a **stay-on-path** SI.

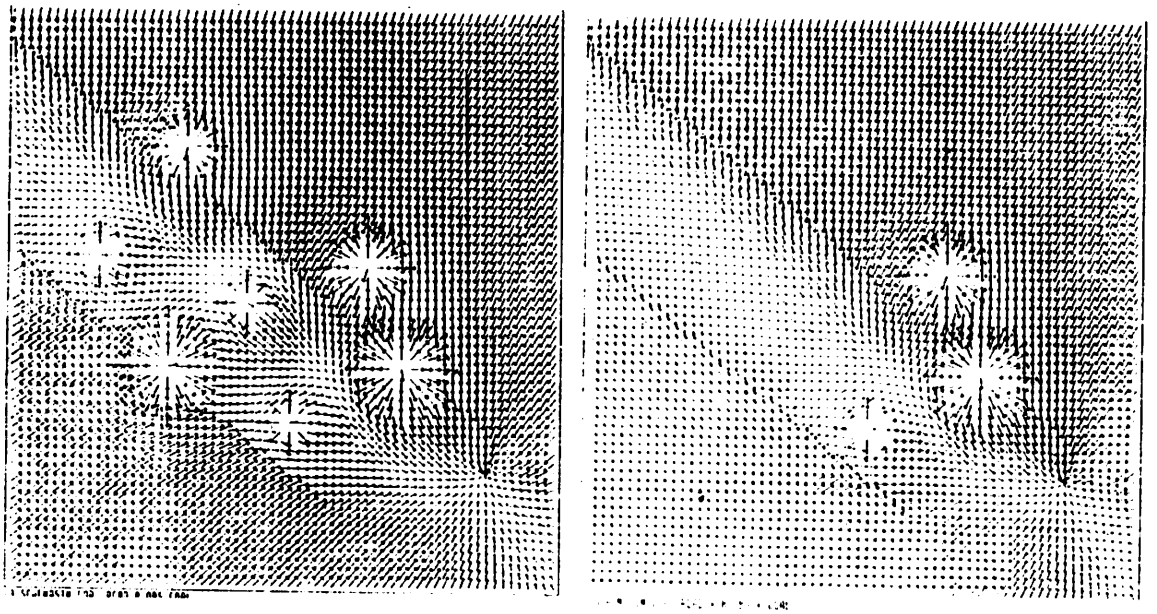
a) Shows the layout of the obstacle ridden course.

b) Simulated robot path through course.

c-e) With the robot starting at the upper left, the robot's progress through the course can be observed. Note that the obstacles are added as they are perceived by the sensory system. No *a priori* knowledge of their location is assumed.

(Figure continued on following page).

(c)



(d) (e)

Figure 53 continued.

Figure 54: Another simulation run

This simulation includes 9 avoid-static-obstacle SIs and 1 move-to-goal SI.

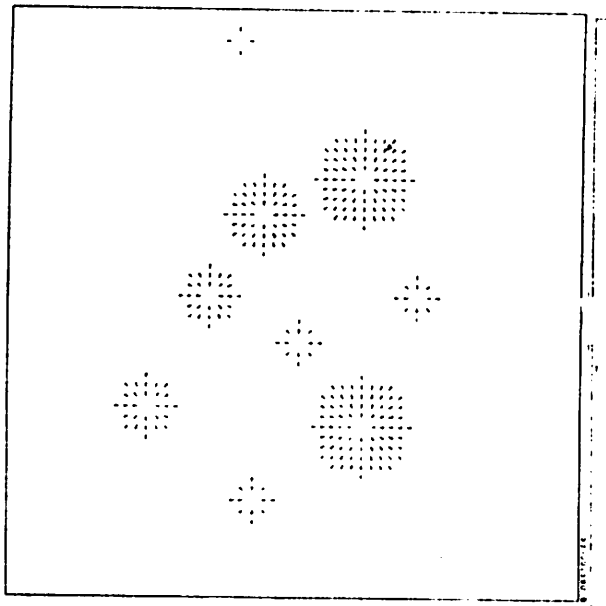
a) Location of 9 obstacles.

b) Path of robot as it crosses from left to right around obstacles to the goal.

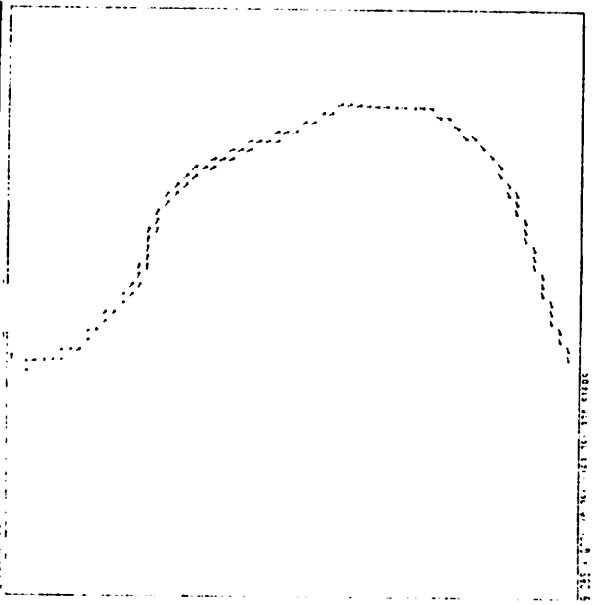
c & d) Velocity fields based on robot's perceptions as it moves from left to right as shown in b).

This simulation includes an uncertainty measure for obstacles which increases with the distance of the obstacle from the robot. If the obstacle is relatively uncertain, its position is shown but it produces no field (e.g. the two rightmost obstacles in Fig. c). As the robot approaches, it becomes more certain of the obstacles and starts to produce a repulsive field surrounding the obstacle.

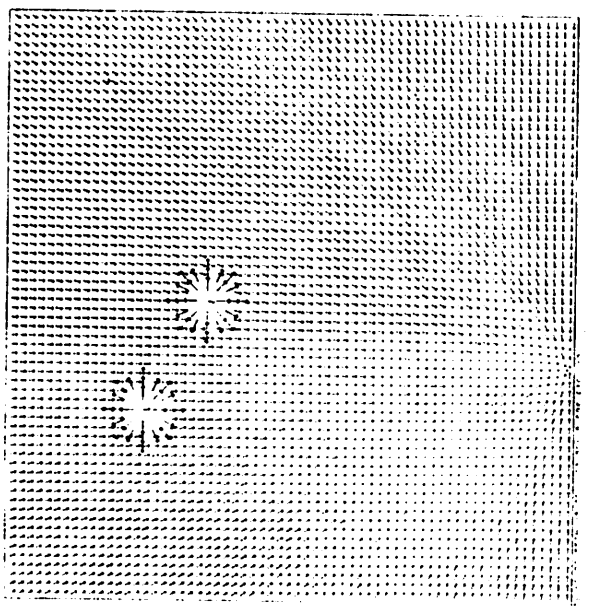
(Figure continued on following page).



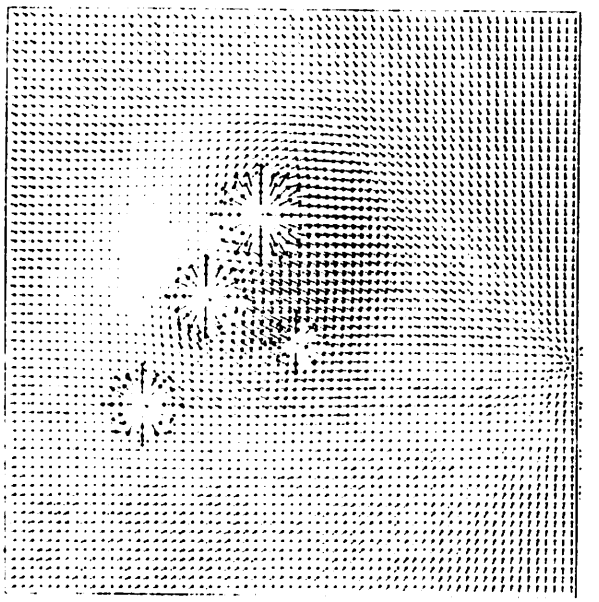
(a)



(b)



(c)



(d)

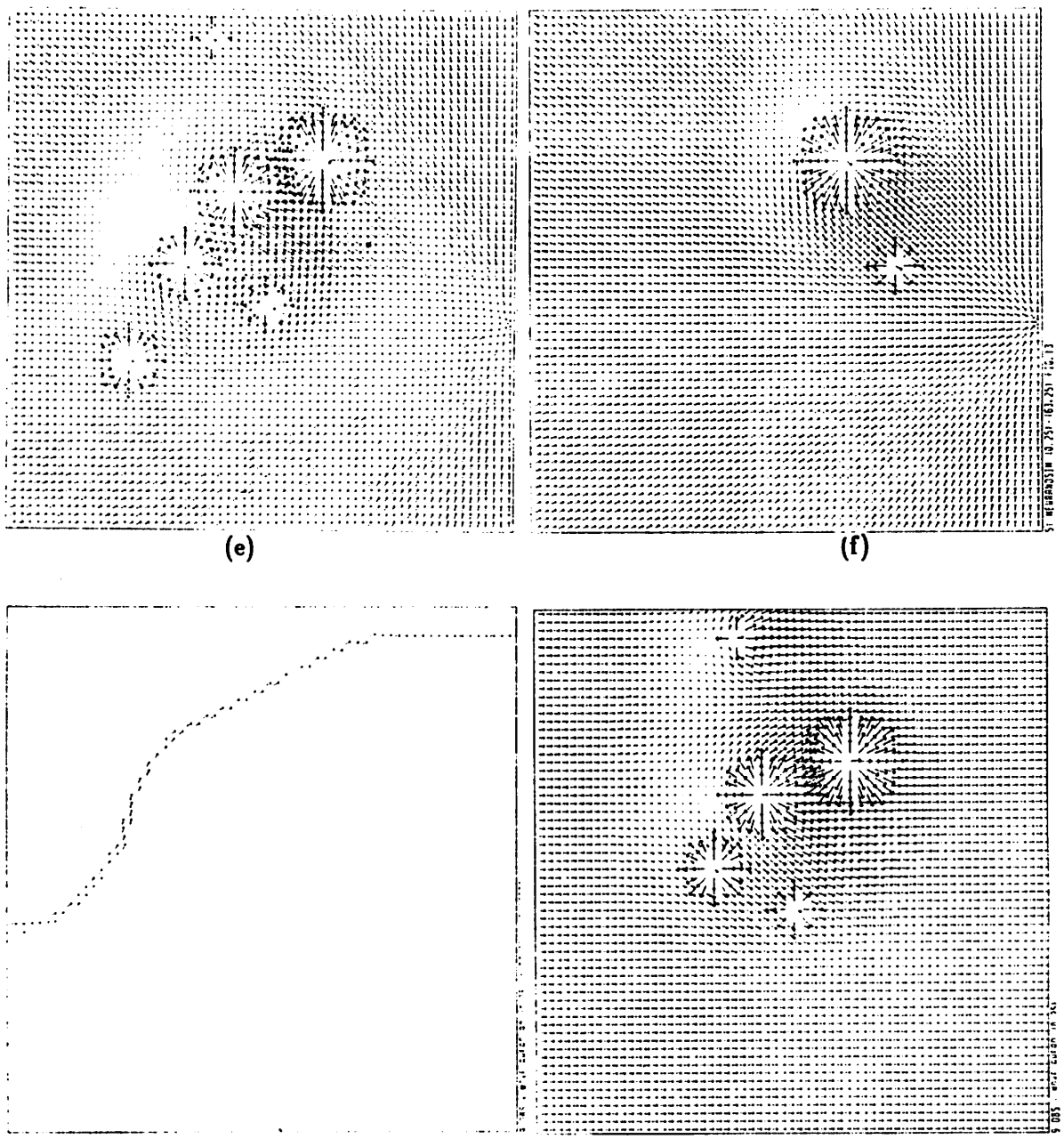


Figure 54 continued.

e & f) Continuation of sequence shown in Fig. 54 c-d.

g) Robot path using the same starting point as in Fig. 54b but a **move-ahead** SI replaces the **move-to-goal** SI.

h) A typical vector field for the path shown in g). Contrast this against Fig. e to see the distinction between moving towards a specific goal (as in e) or just moving in a general direction (as shown here).

program on its MC68000 processor. This provides a small library of functions that are accessible when a terminal is directly attached to the vehicle. To have the robot communicate with a host computer required the development of a primitive library coded in C and documented in [12]. These routines invoked device drivers, coded by Laboratory for Perceptual Robotics co-worker R. Ellis, which were essential in effective and reliable communication with the host VAX. Most of the primitive functions (many based on the DRV counterparts) are listed in Appendix B. These routines are the ones that generally would need to be recoded for a different set of robotic hardware.

In most instances the asynchronous communication protocol is adequate for the task at hand. The major deficit lies in the transmission of the ultrasonic data over a serial line. Typical time for a single package of ultrasonic data (24 readings) to be sent to the host is on the order of 2-3 seconds. Using a time-shared VAX caused an even greater variability in real-time response. One solution was to boost the process priority to very high levels, effectively shutting or slowing down the other user processes. Although this makes response times more predictable, when dealing with multiple AuRA processes running on the same VAX other components of the overall system suffer. When the system is moved to the Sequent in the future, many of the host processing problems should evaporate. Nonetheless, recoding of HARV's on-board terminal emulator to package the ultrasonic data in a more compact format would still be advisable.

§6. Summary

Motor schemas serve as a means for reactive/reflexive navigation of a mobile robot. This schema-based methodology affords many advantages. These include the use of distributed processing, which facilitates real-time performance, and the modular construction of schemas for ease in the development, testing and debugging of new behavioral and navigational patterns. Complex behavioral patterns can be emulated by the concurrent execution of individual primitive SIs.

The use of velocity fields to reflect the uncertainty associated with a perceptual process is another important advance. By allowing the force produced by a perceived environmental object to vary in relationship to the certainty of the object's identity (whether it be an obstacle, goal path, or whatever), dynamic replanning is trivialized. Since the

sensed environment produces the forces influencing the trajectory of the robot, when the perception of the environment changes, so do the forces acting on the robot, and consequently so does the robot's path. This is all accomplished at a level beneath the *a priori* knowledge representations.

It is interesting to note that what might appear to be a naive approach, the summing of the individual vector outputs of the SIs, works quite well, both in simulations and the experimental results described in Chapter 8. Certainly as the velocity increases, so does the need to account for the velocity of the robot itself in the generation of its trajectory. More complex formulations have been forwarded by both Khatib [64] and Krogh [68] for obstacle avoidance using potential fields. These and other approaches for both potential field formulation and combination mechanisms surely merit additional investigation.

There are times when this methodology of low-level reactive planning will fail, as it suffers from the pitfalls common to potential fields. Failure is detected when the robot's velocity drops to unacceptably low levels (in the case of potential field minima) or by exceeding a hard real-time deadline (in the case of cyclic behavior). At those times, the pilot is reinvoked to conduct a "local-global" form of planning (see Chapter 4). The pilot draws on information present in short-term memory including instantiated meadows that are relevant to this particular leg and a sensor-based world model built by the cartographer. This form of replanning should only be needed rarely as navigational planning helps to ensure avoidance of modeled obstacles. Generally only unmodeled obstacles can lead to the breakdown of schema-based navigation. Higher level knowledge then must be invoked to maneuver the robot out of its dilemma. Most of the time however, schema-based navigation is more than adequate for the task.

A working motor-schema-based navigation system has been implemented as part of the AuRA architecture and is used to conduct actual robot experiments validating the concepts shown only as simulations in this chapter. Many different behaviors have been produced using our mobile robot HARV. These include avoidance, exploration, hall following, navigation amidst obstacles, door entry, impatient waiting, "drunken sailor" single wall following, and follow-the-leader behaviors. Experiments demonstrating these simple to more complex activities are described in Chapter 8. The current experimental testbed is not implemented on a multiprocessor, but it is anticipated that when the schema shell is transferred to the Sequent multiprocessor, the motor schema manager

will soon follow. Work is currently underway in extending the two-dimensional schema system to three dimensions [15], ultimately providing navigational capabilities in both the aerospace and undersea domains.

CHAPTER VI

PERCEPTUAL STRATEGIES FOR MOBILE ROBOT NAVIGATION

In order for a mobile robot to be able to navigate intelligently in an only partially modeled world, environmental sensing is necessary. The previous chapter described the role of motor behaviors in driving a vehicle to satisfy its navigational goals. The importance of embedded perceptual strategies (or "identification procedures" [6]) for action-oriented perception as the means for producing these behaviors should now be clear.

The AuRA design (Fig. 2) ideally includes the VISIONS system being developed at the University of Massachusetts under the guidance of Hanson and Riseman [53,55] as the sensory gateway. Due to the extremely high computational requirements of sensory processing and the continually evolving but partially incomplete status of VISIONS, AuRA's initial implementations of necessity require reliance on individual vision algorithms drawn from within that framework rather than the entire VISIONS system itself. These modular algorithms have been modified to come closer to efficient real-time performance than would otherwise have been available. Additionally, they draw upon top-down knowledge and expectations as provided by LTM and/or previous images. Action-oriented perception is the key concept employed in their modification.

Certain common threads run throughout these algorithms. The clear separation of a "start-up" phase from the "update" phase can be seen in most of the visual strategies. Tuning of an algorithm's expectations on a frame-to-frame basis are made based on current environmental conditions, such as lighting, relative robot position, and the like. By providing for adaptability in the tracking of image features, whether they be obstacles, paths, or landmarks, the goal is robust feature recovery. Other work [137,134] in visual navigation uses similar techniques.

Although vision is a principal concern of AuRA, other sensor modalities are exploited where available. Ultrasonic data, available from the ring of 24 sensors surrounding HARV,

provides information regarding the distance to surfaces. Although the discriminatory capabilities of ultrasonic data is limited, it serves a useful purpose in obstacle avoidance and confirmation of visual interpretations.

Shaft encoders, measuring the distance traveled and the amount of rotation of the vehicle, provide limited sensor information. Chapter 7 describes the use of encoder data to manage spatial uncertainty growth. Distances traveled can be approximated using these sensors as long as the realities of their limitations are incorporated into the system and its representation of uncertainty. Shaft encoder data does not truly involve environmental sensing. It only records the number of rotations of the robot's motors, not the changes in the robot's position relative to the world. These limitations are discussed later in this chapter as well as in Chapter 7. An inertial navigation system would be highly preferred if available or its cost could be justified. Unfortunately, neither is true for our experimental environment.

The balance of this chapter is divided into the following sections. Section 1 discusses the potential relationship between VISIONS and AuRA. Section 2 describes the practical short-term role of specific modular vision algorithms used within AuRA. These include a fast line finder, a fast region segmenter, a depth-from-motion algorithm, and interest operators. Section 3 describes the ultrasonic algorithms in use in AuRA, including obstacle avoidance, door finding, panic sensing, and localization. The limitations and use of shaft encoder data appear in Section 4. Section 5 briefly describes other desirable sensors that may be valuable for potential integration into AuRA. Section 6 discusses some of the implementation details for AuRA's perception subsystem. The chapter then concludes with a summary and evaluation of the role of the different perception techniques currently employed in AuRA.

§1. VISIONS and AuRA

Scene interpretation has long been a primary research effort within the VISIONS group at the University of Massachusetts. Considerable literature exists describing the progress to date [102,140,54,53,55,41]. The remainder of this section will first briefly describe the operation of the schema system, followed by the role that the schema system can play in mobile robot navigation. It should be understood from the onset that schema-

based scene interpretation is currently a very time-consuming process. Work is underway, however, to provide parallel hardware (the UMASS Image Understanding Architecture [55,139]) to speed up this process by several orders of magnitude. Additionally, available *a priori* knowledge present in LTM can be used to guide schema instantiation and reduce the processing requirements dramatically.

§1.1 *The Schema System*

The VISIONS schema system accepts an image as input and produces a labeled interpretation of the observed environmental objects (Fig. 55) and, to the degree possible, a 3D representation of the environment. There are three levels of processing available utilizing both bottom-up and top-down processing (Fig. 11). Taking a bottom-up view first, the low-level processes operate on pixel level data producing an intermediate symbolic representation of line, region, and surface tokens. At the highest level, schema processes exist which interpret and collect the intermediate representations into labeled objects.

If no top-down guidance were available, it would be virtually impossible for the system to converge on an acceptable interpretation. Perceptual schemas (in the context of VISIONS) post hypotheses about what specific image events mean. Each highly rated hypothesis guides intermediate and low-level processes in an effort to find self-supporting evidence. This top-down guidance brings the intermediate and low-level processing requirements down to tolerable levels. If the hypothesis cannot find sufficient support or is contradicted by other data, it is deinstantiated. On the other hand, if sufficient support for a hypothesis is available, that particular portion of the image will be labeled as being associated with a particular environmental object and inference mechanisms can direct further semantic processing.

It is quite difficult to describe the operation of the schema system in a few paragraphs. It is hoped that the interested reader will refer to the more comprehensive descriptions cited above [esp. 55,41] for a better understanding of its operation.

§1.2 *Utilization of VISIONS Schemas in Mobile Robotics*

The principal test domains to date for VISIONS schema-based scene interpretation have been house scenes and road scenes (Fig. 55). These efforts have been predominantly concerned with full scene labelings with no expectations of specific instances of object

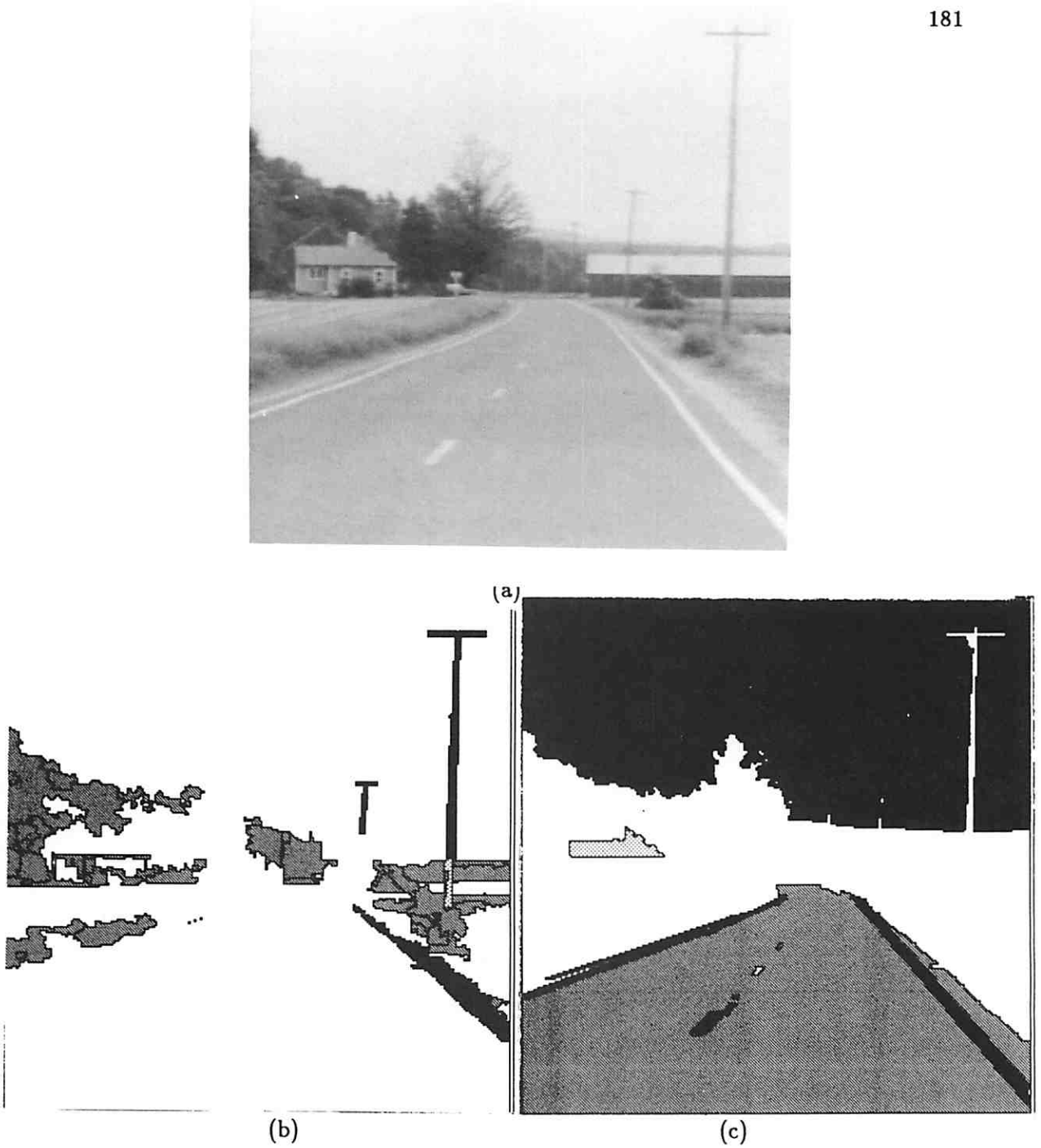


Figure 55: Schema-based scene interpretation

a) Original image.

b) Foliage, tree trunk, telephone pole, and gravel.

c) Sky, roof, road, and roadline.

classes in the particular image or environment in question other than it having such examples present; thus a general house or road scene is expected, but there is no world map of the domain.

If *a priori* knowledge of a specific environment is available, it can guide the posting of schema hypotheses, increasing the reliability of object recognition and reducing the amount of computing time required to achieve a satisfactory labeling. If the robot's position is approximately known within a global map, this information regarding the potential position of environmental objects (e.g. landmarks or roads) can be used to restrict the formation of object hypotheses to particular portions of the image. The occurrence of two known objects in predicted positions relative to each other can significantly increase the plausibility of a proposed interpretation.

Where can schema-based scene interpretation be used in mobile robotics? In the most grandiose sense, one can say for everything. If a completely and correctly labeled image is available, it can be used for navigation, obstacle avoidance, localization, goal recognition, etc. Indeed several of the other algorithms described below (line finding, region extraction, etc.) actually constitute some of the lower level processes used within the VISIONS system. Being realistic however, one must recognize that real-time responses are necessary for mobile robot navigation, indicating that schema-based scene interpretation is presently too slow to be effective. A more appropriate current use of the VISIONS schema system would be to provide for the top-down extraction of semantic objects of interest required for several of the other visual processes. If the initial image is analyzed by the scene interpretation mechanisms, it could yield the road edges that can be used to bootstrap the **stay-on-path** motor schema and **find-path** perceptual schema. Additionally it could provide the initial region statistics to seed the region extraction algorithm for path-following and landmark or goal recognition. Start-up information for the depth-from-motion algorithm could also be provided, in addition to potential corners that are of use for localization purposes by the interest operator. Finally, if the robot becomes sufficiently disoriented relative to its global map, the schema interpretation system could be invoked to enable the robot to regain its bearings relative to the modeled world.

§2. Modular Vision Algorithms

Although nothing in AuRA restricts sensor processing to be predominantly visual, much of the architecture is constructed to utilize this form of sensing. Action-oriented perception is the fundamental premise on which motor schema sensing and navigation is based. It is not necessary for the robot to fully understand the entire scene before navigation can be initiated (although this would certainly make things easier). Instead, by directing specific sensing strategies and the available computational resources to the motor needs of a particular task, only those portions of the scene which can contribute to the attainment of the pilot's goals are analyzed. Particular sensor algorithms are chosen to fit the demands of the specific path leg at hand.

No single perception algorithm is a panacea for navigation. The designer's goal instead is the development of a wealth of visual and other sensing algorithms which can provide the breadth that multi-domain navigation requires. A design goal of AuRA is to provide navigational capabilities in both indoor and outdoor environments, allowing for considerable environmental diversity in each of these cases.

Computationally efficient vision algorithms are used to provide navigational information for the robot and are initially implemented on a single processor. In later implementations specific processors will be dedicated for each algorithm to improve performance and eventually the load will be distributed over parallel hardware.

From an experimental point of view, this architecture affords the flexibility to try new perceptual strategies without forcing significant changes in the supporting system components. By embedding motor actions as behaviors and perceptual strategies as focus-of-attention mechanisms, both represented in a schema form, the addition, modification and deletion of these program units is manageable. The emphasis is on modularity. New world representations can be embedded in LTM through the use of the feature editor, providing for representational extensions that may be needed by new algorithms.

Typical of many of the algorithms is their ability to be decomposed into two phases: start-up (bootstrap) and update (feedforward). The start-up phase performs more slowly and has less, if any, *a priori* knowledge to work from. The start-up process produces initial region seed statistics, depth information, line orientation, etc., for establishing expectations which are used to advantage in subsequent frame analysis. The update stage uses

the information provided from the start-up phase to restrict the possible interpretation of image events and limit the search area for those events, thus reducing processing time significantly. The initial output of the start-up phase is updated after each processing run and is fed forward to provide a basis to guide analysis of the next image.

The remainder of this section will discuss some of the sensor algorithms that exist for use within AuRA. The strategies described below are not exhaustive, but rather represent the current initial elements being introduced by VISIONS researchers for use within AuRA's framework. It should be noted that a vision algorithm by itself is useless for navigational purposes. Considerable amounts of additional software must and has been created in order to produce intelligent motion of a robot vehicle using that algorithm. Each of the algorithms described below has been used for navigation experiments with HARV (Chapter 8).

§2.1 *Line Extraction*

Line extraction has the potential for multiple uses within AuRA. These include path edge extraction for use by **stay-on-path** schemas, landmark identification for **find-landmark** schemas, and as a texture measure for terrain identification. Of these, the first two are currently being developed for use in AuRA. The remainder of this section will first describe the fast line finding algorithm, and then its application to both path following and localization purposes.

Fast Line Finder (FLF)

A fast line finder based on Burns' algorithm [31] has been developed by Kahn, Kitchen and Riseman [64]. It is a two-pass algorithm which first groups the image data based upon coarse quantization buckets of gradient orientation into edge-support regions. This grouping process collects pixels of similar gradient orientation into separate regions via a connected components algorithm. The gradient magnitude does not affect the line extraction process. A line is then fitted to the resultant edge support region. FLF differs from the original Burns' approach by permitting the specification of the gradient orientation buckets and the extraction of the representative line for each edge support region. Many of the elementary computations can be further speeded up through the use of a conventional pipeline processor which supports a look-up table and convolution

processing. An outline of the algorithm appears in Appendix C.

Fragmentation of a potentially long image line often occurs if no *a priori* knowledge is available regarding the approximate orientation of the line in the image. The likelihood of extracting a particular long line increases by tuning the bucket's orientation to be centered on the anticipated orientation of a road edge or other line model in the image through the use of available knowledge extracted from LTM or previous images.

The key concept is *action-oriented perception*, performing only that computation which is necessary for the specific task at hand. Features available within the FLF algorithm to support this concept include the ability to scope the image (i.e. perform line extraction on a subwindow of the image). If the robot has approximate knowledge of the world position (and hence image position) of the line feature being sought (derived from LTM, the spatial uncertainty map, and/or previous images), substantial processing reductions are attained by ignoring those portions of the image where the feature is unlikely to occur. In addition to orientation, the FLF can be adjusted to filter lines based on gradient magnitude, dispersion, size of the region, and length. By adjusting these filters in advance, based on the features desired (e.g. short lines for texture, or long lines for roads), unnecessary processing is minimized. A secondary filtering procedure is also available for removing lines after the fast line finder has been run, making it possible to collect different sets of lines with different characteristics from only a single run of the more time-consuming FLF. This is possible because when the lines are produced, statistics regarding each line are collected and stored with the endpoint data for later reference.

Figure 56a is an image of a sidewalk scene. Figure 56b shows the results of the FLF using the full default set of coarsely quantized orientation buckets for the entire image. Figure 56c shows the results with the orientation buckets tuned and the subimage scoped to the anticipated road edge based upon the internal model of the vehicle position and orientation, while Figure 56d shows the results with the buckets tuned to horizontal and vertical edges, filtering to retain longer lines and with the image scoped above the horizon.

Path Following

A significant contribution of this dissertation involves the application of the FLF to path following. Using line finding to extract path boundaries requires the grouping of

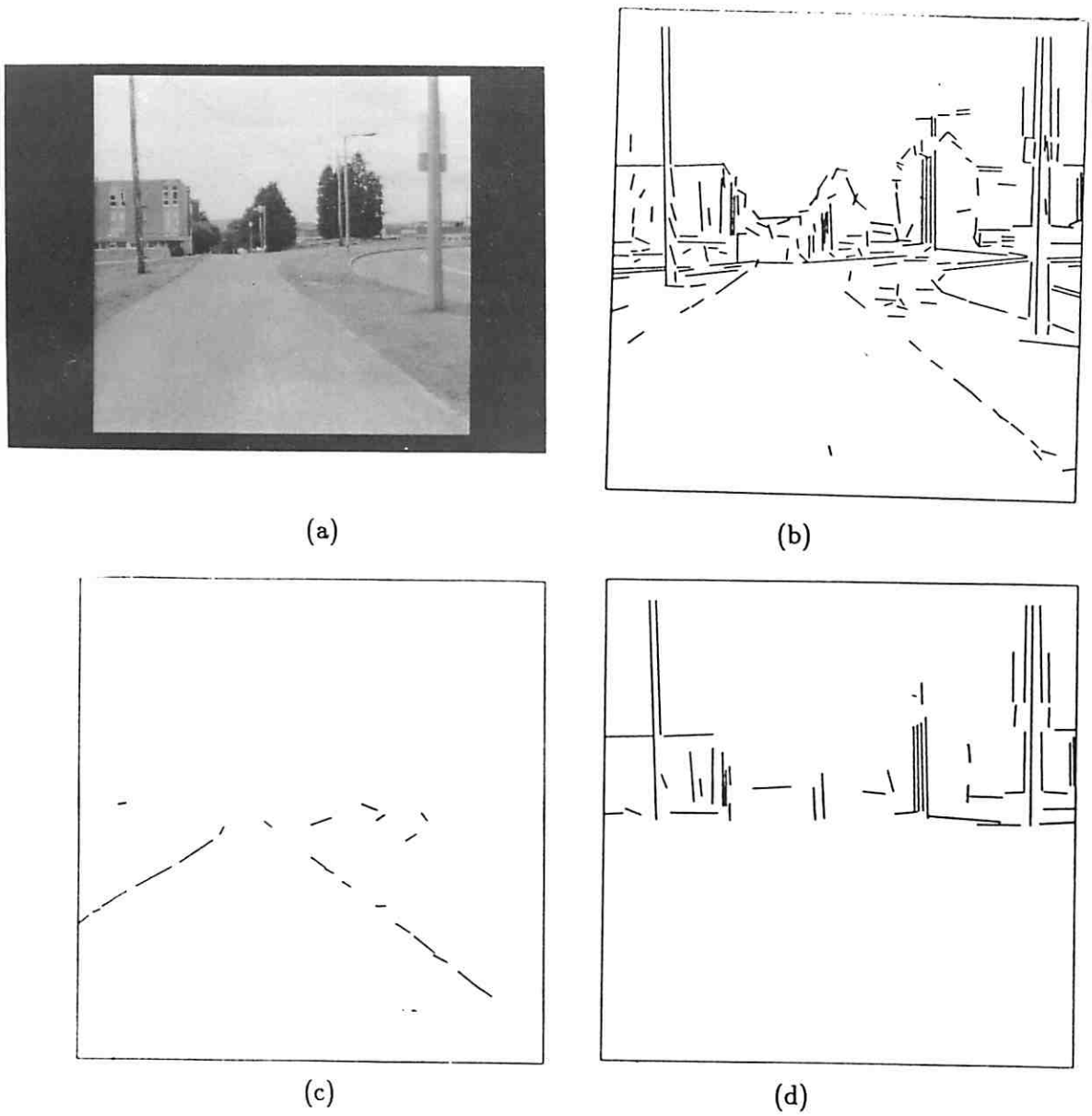


Figure 56: Fast line finding

a) Original sidewalk image.

b) Default bucket orientation.

c) Buckets tuned to road edges.

d) Buckets tuned to long vertical and horizontal lines above horizon.

resultant FLF line fragments into a single line representing each path edge. No effort is made to condition or modify existing paths to make this process easier (e.g. by adding stripes, cleaning, etc.). The grouping strategy used must be able to deal with fragmentation and edge discontinuities, such as path intersections, leaves, etc.

If the uncertainty of the vehicle is within reasonable limits, predictions of the position and orientation of the road lines in the image plane can be made. As described above, there are two distinct components of road-following (see also [137]): the bootstrap or start-up phase, where the road edge is determined in the image for the first time; and the feedforward or update phase - where a previous image is used to guide the processing for the next image. Line finding is not necessarily the best strategy for initially finding the road's position. Nonetheless, it can be reasonably effective if the road appears on a global map of the terrain and there is approximate information about the vehicle's position and orientation. These are both present within AuRA: in LTM and the spatial uncertainty map, respectively.

The feedforward phase assumes that the approximate position of the road was known in the last image. This information, when coupled with the commanded translation and rotation the robot has undertaken since the last image acquisition, can be used to predict approximately where and at what orientation the road edges will occur in the newly acquired image. As anyone who has worked with mobile robots knows, the motion that a robot actually takes may differ quite significantly from that which it was commanded to perform. Consequently, there must be a considerable margin for error in these predictions if the algorithm is expected to be robust. Additionally, there must be some measure of the confidence in the line produced representing the road edge.

Path edge grouping proceeds as follows: The buckets are tuned based on the anticipated position of the road edge in feedforward mode; in bootstrap mode either knowledge from LTM or the default buckets would be used. The fast line finder is then run, producing line fragments in the approximate orientation of the path edge (Fig. 57a). These fragments are then filtered based on their distance from the anticipated image line and the expected orientation of either the right or left edge. Again the amount of tolerance allowed is controllable. This yields two sets of line fragments (one for each path edge - Fig. 57b-c). All the fragments above the vanishing point of the road, (obtained from feedforward information), are discarded. The center of mass of the midpoints of

the remaining line fragments is computed, each midpoint weighted by the length of the fragments themselves. The average orientation is computed in a similar manner. The resulting point on the line and computed line orientation determine the line equation for each road edge. The left and right edges are then used to compute the road centerline (Fig. 57d). The centerline is the basis for determining the rotational deviation of the vehicle relative to the road's vanishing point as well as the translational deviation from the road centerline. These newly computed path edges are then used as the models for the next feedforward step.

The total length of the line fragments used in producing the path edges serves as a measure of uncertainty. If this value drops below a specified threshold, special processing is undertaken. This includes increasing the error tolerances and margins in the FLF to see if a more confident line can be extracted from the same image, or if that fails, to digitize another image in the event that a passing obstacle blocked one or both path edges. If both of these strategies fail, the robot will reposition itself slightly and try another image. If this yet fails, alternate bootstrapping methods must be brought to bear.

Although the FLF algorithm was written by other members of the VISIONS group, considerable work was required to produce a useful tool for mobile robot navigation. The feedforward mechanisms, line fragment grouping, centerline extraction, image sequence acquisition, and vehicle servoing routines all had to be produced before the algorithm was suitable for navigational purposes. This is typical of all of the vision algorithms described in this chapter.

The robot is able to successfully navigate both an outdoor sidewalk and an indoor hall using the FLF. Approximately 10 CPU seconds (VAX-750) are required for each step to provide the robot information for traveling 5.0 feet ahead. This is approximately two orders of magnitude faster than the original Burns' algorithm [31]. The 512 by 512 image digitized on a Gould IP8500 is averaged to 256 by 256 before line extraction. This time can be reduced by using pipelined hardware available on the digitizer. The vehicle servos on the computed centerline position, correcting both orientation and translational drift as it proceeds. See Chapter 8 for details of these and other line-finding navigation experiments.

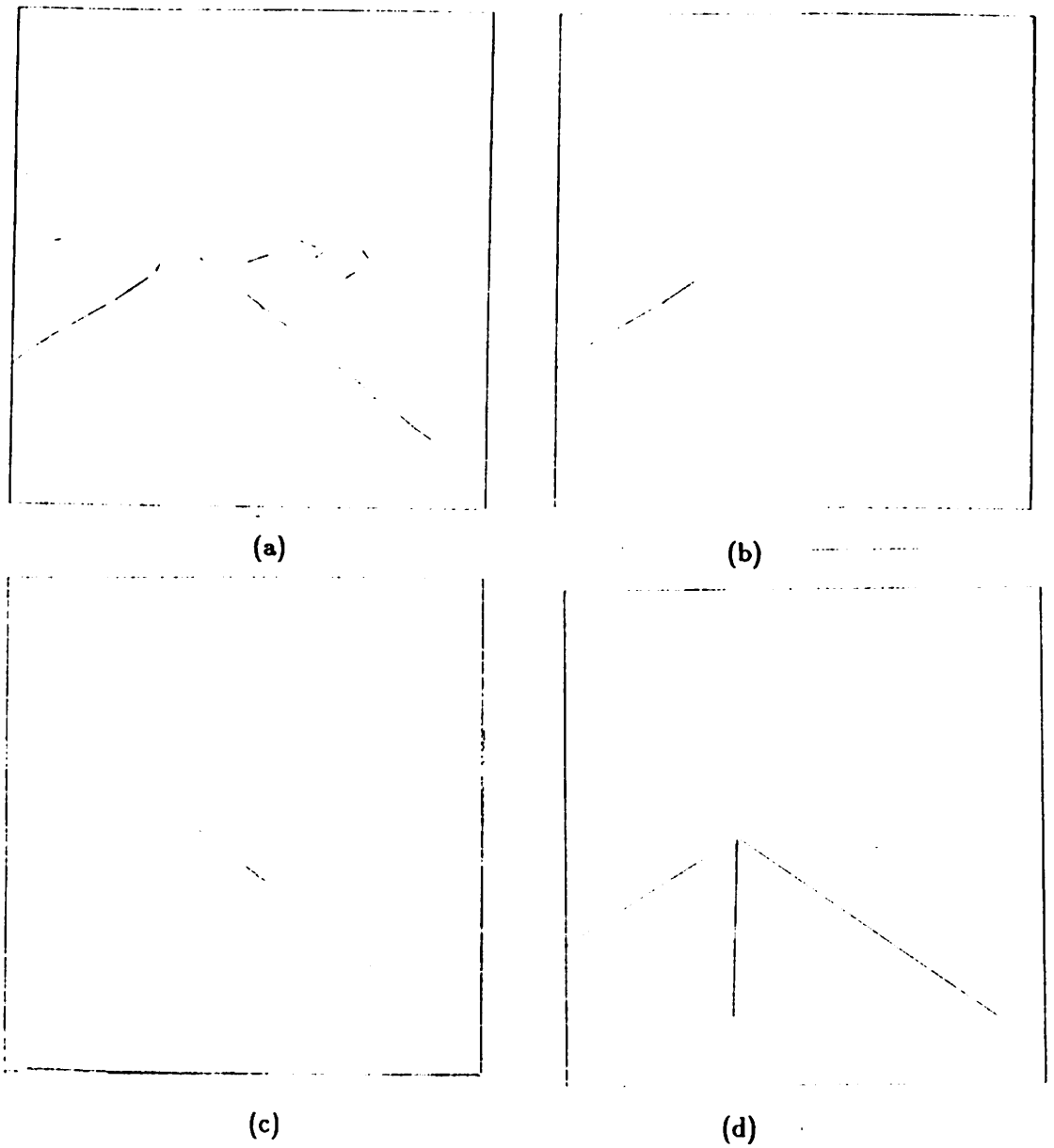


Figure 57: Path following with FLF
a) Output of FLF when run on image 57a.
(lines below horizon with orientation-tuned buckets).
b) Fragments left after filtering and windowing for left path edge.
c) Fragments left after filtering and windowing for right path edge.
d) Resultant path edges and computed road centerline.

Landmark Identification through Line Finding

Vehicle localization is addressed by the line finding algorithm using data stored in LTM. Localization is simply orienting the vehicle relative to its global map; in other words getting its bearings. It is not proposed that lines are the only mechanism for localization, but they should serve in conjunction with other relevant algorithms. In the role of a confirmation mechanism, or for tracking from frame-to-frame a previously identified landmark feature, FLF localization is well suited. Extracting the edges of a path as described above also provides information for localizing the vehicle, assuming the path is represented in the world map.

Long, strong vertical lines and corners derived from such lines are probably the most appropriate general category of lines suitable for this application. Edges of buildings, telephone poles, lampposts or doorframes can be tracked using the line finder. Figure 56d shows the result of running the FLF on image Figure 56a with the buckets and filters tuned for long horizontal and vertical lines of high gradient magnitude. This orientation can be used to identify the roofs of buildings against the sky or road intersections directly in front of the vehicle. By windowing the image for a certain landmark based on the position of the vehicle as indicated by the spatial uncertainty map and *a priori* knowledge of the global coordinates and dimensions of the object feature in question (from an environmental map in LTM containing object attributes and locations), it becomes possible to isolate features such as the corner of a building by combining the evidence from both horizontal and vertical lines. This then is used to constrain the positional uncertainty of the vehicle by backprojecting the 2D data to 3D world coordinates when combined with the knowledge of the height of the feature (see Chapter 7).

§2.2 *Fast Region Segmenter (FRS)*

FRS is a region extraction algorithm operating in a manner akin to the fast line finder, but based upon similarity of color and intensity features. It functions by first defining a look-up table that is used for classifying an input image. This algorithm has been motivated by histogram-based segmentation algorithms [67], but achieves great simplification via constraints from stored object knowledge in LTM or the result of processing previous frames, e.g. the look-up table ranges for specific objects may be defined on the basis of previous frames or from object data in LTM. The input image used can be an

intensity image, a gradient image, a color image component, etc. This input image is scoped (windowed) as in the case with the FLF. The look-up table maps ranges of pixel values to specific region labels. Available knowledge is used to define expected ranges of spectral attributes of interesting objects (Figs. 11,58,59). The resulting classified image is then subjected to a region extraction algorithm which groups the classified pixels into regions. Statistical data is then collected regarding each region. See Appendix C for an outline of the FRS algorithm.

The speed of this algorithm arises from the use of the look-up table to provide a quick mapping to the image. The connected components routine is then run on a restricted portion of the image selected through the use of top-down map constraints (see Chapter 7).

This segmentation is used for path extraction as in [124,128]. Preliminary experimentation using intensity images can be seen in Figure 58. Figure 58a shows the original image and Figure 58b the region extracted representing the sidewalk. The statistics collected for the sidewalk region are then used for providing the expectations (feedforward) for the next image in the sequence [as in 124]. Color was not used for this segmentation and the algorithm would be much more powerful with RGB input. Chapter 8 presents experimental results using FRS path following with HARV.

Landmark extraction is handled similarly. The centroid of the landmark can be used for localization purposes, in contrast to the edge detection methods used by the FLF or the corner detection approach used with the Moravec operator described below. A bright yellow road sign (very dark in the blue sensory band) is segmented for localization purposes in Figure 59.

§2.3 *Depth from Motion*

Passive navigation by the determination of the position of environmental points via vision is an important sensor strategy for AuRA. The motion research group within VISIONS has long explored the extraction of depth from motion [142,73,2]. A more recent algorithm, developed by Bharwani, Riseman and Hanson [22], uses a sequence of frames under known translational motion of the sensor to incrementally refine positional estimates of objects over time. It can be used in mobile robotics for obstacle avoidance, position localization, and as evidence in object identification.

(a)

(b)

(c)

Figure 58: Sidewalk extraction via region segmentation

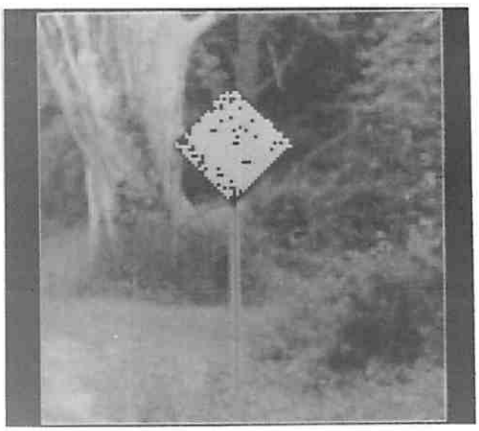
a) Sidewalk image.

b) Resultant extracted region representing sidewalk.

c) Resultant extracted region representing central portion of sky.

(a)

(b)



(c)

Figure 59: Landmark identification via region segmentation

a) Original sign image (combined RGB into intensity).

b) Blue plane of (a) chosen via LTM for analysis due to the spectral data of anticipated landmark.

c) Extracted region representing sign.

Algorithm

A brief sketch of the multiple frame depth-from-motion algorithm developed by Bharwani, Riseman, and Hanson follows. The reader is referred to [21,22] for the details of this approach. The algorithm allows refinement of depth over time up to some detectable limit, while maintaining a constant computational rate. This is very important for real-time processing.

The problem of recovering depth from motion in a sequence of images again involves the decomposition of the problem into two components: start-up and updating. This algorithm makes the assumption that the camera is undergoing pure translational motion and the position of the focus-of-expansion (FOE) is known within some reasonable estimated degree of accuracy.[1] This implies that the image displacement paths for a static environmental feature are constrained to move in a straight-line emanating radially from the FOE. An *interest* operator is used to extract *distinctive* points in the image, i.e. those of high curvature and contrast, that are likely to avoid false correlation matches in future frames.[2] It is assumed that the obstacles or landmarks will exhibit some such points on their boundaries. This is probable if the backdrop is bland (e.g. the road itself) or by deliberately retrieving only "interesting" landmarks from LTM. However, it should be expected that interest points will be extracted from both relevant and non-relevant image events.

The correspondence problem is the principal difficulty; how can one be sure that the feature in one frame corresponds to the same feature in the next frame after the robot has undergone translation? The start-up phase involves finding initial correct feature correspondences between the first two frames, while the update phase involves the use of the start-up analysis and the consequent approximate depth values to restrict the search area for corresponding matches at a higher match resolution in subsequent frames, thus reducing (or in general bounding) computation and providing refinements of the original depth estimates. Work by Snyder [119], addressing the limits of uncertainty in this type of motion, is fundamental to efficient use of previous correct correspondences in constraining

[1]These assumptions are not necessarily safe when using real world images. Frame registration and FOE recovery are problems that need to be solved. A discussion appears at the end of this section.

[2]See [2] for a method in dealing with the correspondence problem using a confidence measure. These ideas are implicit in the Bharwani algorithm.

the match in future frames.

Assuming the robot is traveling at a known velocity, the pixel displacements found between the first two images of a sequence (start-up) are used to further reduce the search for feature matching in successive frames, once the images have been registered so that non-translational motion of the camera has been subtracted out. Known sensor motion leads to a constraint on the match path, and approximate depth (from start-up) constrains the portion of the path to be matched. Progressive refinements can be made in the estimation of feature displacement and hence distance to a relevant feature.

Different strategies such as histogramming the collection of points on the basis of depth, determining orientation of surfaces based upon the depth of several points on associated regions, or identifying landmarks by correlating distance from the viewer with the objects in the environmental map in LTM, are all possible methods for extracting objects from the environment. The current approach for obstacle extraction is described in the subsection dealing with depth-from-motion system issues below. This data can then be used to provide information to the motor schema manager for effecting evasive action in the case of obstacles or for use in localization in the case of landmark location.

Applications

A primary goal of the depth-from-motion algorithm is to provide information about the distance of an object lying in the path of the robot. In obstacle avoidance applications, computational requirements are made tractable by restricting the processing to interest points (i.e. trackable image points of high contrast and curvature) and only to those that are lying within the current path of the robot.

Figure 60 and Table 1 illustrates some results using the depth-from-motion algorithm for obstacle avoidance. Chapter 8 discusses the experimental results obtained with this method using HARV. The biggest problems encountered in the use of this algorithm in mobile robotics include first, accurate recovery of the FOE, which can be minimized through accurate calibration of the camera relative to the robot, and second, ensuring registration of the images. Stabilizing the camera with a gyroscopic platform affords a hardware solution to the registration problem. A software solution [106] can be partially achieved by registering the images via correlation matching using points near and above the horizon, i.e. distant features (hence relatively unmoving with respect to the modest

amount of camera translation, thus any image translation or rotation observed can be assumed to be a consequence of improper image registration) that can be registered from frame to frame. Large rotations pose a particular problem and require many distant interest points and significant computation. The depth-from-motion algorithm is quite sensitive to misregistration due to rotation in the image plane, so every effort is made to minimize or eliminate any roll movements of the camera relative to the scene. Additionally, the FOE can only be extracted up to one degree of accuracy causing errors up to ± 5 pixels in a 256 by 256 image. This causes error in the returned value for depth, although the point tracking itself is generally unaffected. Extraction of depth in this manner is a difficult although promising problem.

The motion algorithm can be used for landmark identification as well. This is actually a simpler task than obstacle avoidance in many respects due to the availability of LTM knowledge to guide processing in a top-down manner. Knowledge of the approximate distance of a landmark to the vehicle in a restricted portion of the overall image substantially reduces the computation required. When approximate ranges for the distance to an obstacle are known, the algorithm will perform more robustly than when underconstrained. Portions of the image can be searched that are outside the obstacle avoidance regions. As these are usually further from the FOE than points in the robot's direction of motion, greater pixel displacements will occur and hence better results in the depth analysis.

Depth-from-motion system issues

The depth-from-motion algorithm requires considerable support from other vision algorithms in order for it to be used for obstacle avoidance. The overall flow of control of the different components is shown in Figure 61. These components consist of a standalone Moravec interest operator, extract-focus-of-expansion (FOE), depth-from-motion, and obstacle extraction algorithms. It is also desirable to have a registration algorithm if the images are not acquired from a stabilized platform. Each of these components will be described in turn.

The Moravec operator is used to find the initial "interesting" points in the first incoming image. These points are used for establishing correspondences with other points in future images via intensity correlation of subwindows after the vehicle has undergone

(a)

(b)

(c)

Figure 60: Depth from motion

a-c) Three image sequence (distance traveled is approximately 4 feet between frames). Figures b) and c) show the corresponding tracked points. The results for the interest points are presented in Table 1.

(Image sequence taken from CMU NAVLAB).

Table 1: Depth from motion results

object	feature	nominal depth and uncertainty (± pix)			true depth (feet)		
		1-3	3-5	5-7	$frame_1$	$frame_3$	$frame_5$
cone1	1	76.38 ± 25.54	77.29 ± 16.96	72.48 ± 7.05	76.00	72.00	68.00
	5	90.16 ± 30.18	66.14 ± 10.65	62.07 ± 4.46	76.00	72.00	68.00
	6	92.63 ± 30.07	70.43 ± 11.66	61.33 ± 4.25	76.00	72.00	68.00
cone2	2	66.45 ± 16.57	90.93 ± 20.66	69.30 ± 5.61	76.00	72.00	68.00
	7	84.04 ± 22.51	82.15 ± 13.60	67.06 ± 4.46	76.00	72.00	68.00
	8	87.30 ± 25.19	77.68 ± 13.03	67.94 ± 4.74	76.00	72.00	68.00
cone3	3	53.12 ± 9.39	55.54 ± 6.47	47.44 ± 2.23	56.00	52.00	48.00
	11	59.84 ± 10.04	53.62 ± 5.17	47.73 ± 1.91	56.00	52.00	48.00
	12	54.80 ± 8.22	50.68 ± 4.48	46.05 ± 2.08	56.00	52.00	48.00
cone4	4	80.81 ± 26.83	60.07 ± 9.74	48.07 ± 2.93	56.00	52.00	48.00
	13	58.23 ± 10.88	58.56 ± 7.06	46.16 ± 2.06	56.00	52.00	48.00
	14	57.06 ± 10.38	53.97 ± 5.93	45.60 ± 2.73	56.00	52.00	48.00
can	9	48.21 ± 6.39	44.97 ± 3.52	38.54 ± 1.19	46.00	42.00	38.00
	10	45.75 ± 6.32	46.74 ± 4.18	39.92 ± 1.41	46.00	42.00	38.00
	16	44.78 ± 4.38	44.38 ± 2.83	37.65 ± 0.94	46.00	42.00	38.00
	17	46.84 ± 5.24	45.39 ± 3.45	39.01 ± 1.09	46.00	42.00	38.00
cone5	15	38.30 ± 4.25	35.86 ± 2.34	27.36 ± 0.79	36.00	32.00	28.00
	18	40.01 ± 4.38	33.22 ± 1.65	28.50 ± 0.56	36.00	32.00	28.00
	19	35.94 ± 3.26	32.79 ± 1.50	28.43 ± 0.53	36.00	32.00	28.00
cone6	20	20.34 ± 0.63	16.17 ± 0.22	14.17 ± 0.15	21.00	17.00	13.00
	21	20.53 ± 0.54	16.91 ± 0.21	* ± *	21.00	17.00	13.00
	22	19.95 ± 0.48	17.88 ± 0.22	* ± *	21.00	17.00	13.00

These tables contain the results for the images from Figure 60. (from [22])

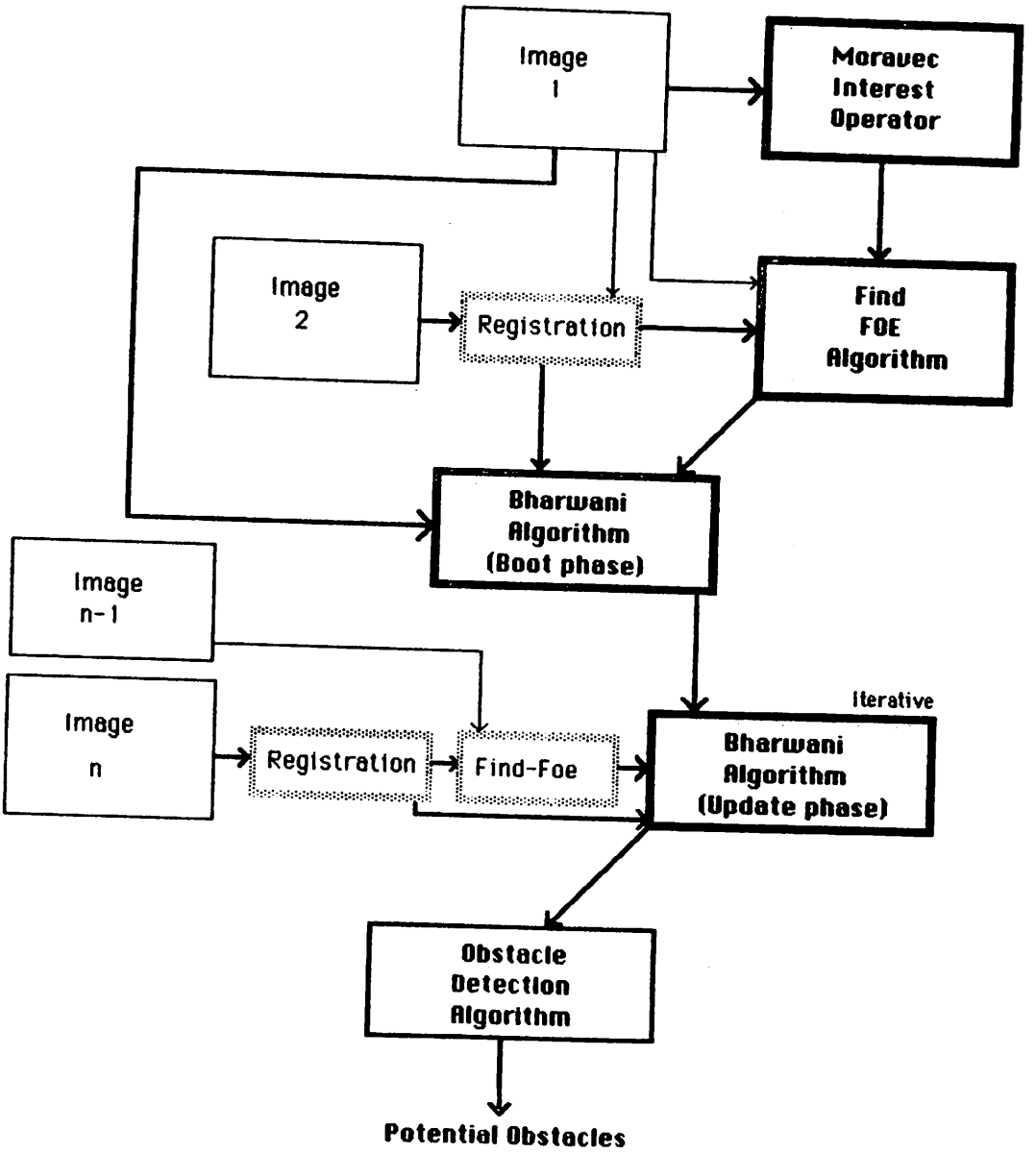


Figure 61: Depth from motion system

The fuzzy boxes (registration and second find-foe) are desirable components for the overall system, but are not automated in this implementation. Image registration is by hand and the FOE is assumed constant for the duration of the sequence.

translation. At this stage, the portion of the image in question is restricted to only those areas of interest for obstacle detection. This window is maintained throughout the rest of the overall system processing. The number of points extracted is user controllable as well; the fewer the points, the less processing time in making correspondences, but of course the less robust the results might be.

The second image now arrives, the robot having moved an approximately known distance. At this point, a registration algorithm, if available, is used to remove any translations or rotations of the camera that are not along the direction of translation. These can arise from bumps or rough spots in the road, eccentricities in the tires, etc. Registered images should be used throughout the rest of the processing.

The two available images are now presented to the find-FOE algorithm. This program [105] determines the focus of expansion for two successive images. If pure translational motion can be assumed for the sub-sequence of future frames, then the find-FOE algorithm need only be run once, as the same FOE will be present in all subsequent images. As this is not necessarily the case, the find-FOE algorithm may have to be run between all pairs of incoming frames, to allow for the movement in the position of the FOE.

The depth recovery algorithm is then run. Iterative refinement of depth occurs as each new image is acquired because the previous value of depth more tightly scopes the displacement in future frames which are matched at a higher correlation resolution. After a prescribed number of images and/or translational motion, the depth-from-motion algorithm transfers its tracked points and associated depths to the detect-obstacle module.

The detect-obstacle component, coded by the author, makes the assumption that the tracked points returned contain environmental points both on the obstacles and on the ground plane. An assumption is made that the area in front of the vehicle can be reasonably approximated by a ground-plane and that the vehicle itself is on that plane. A least squares line-fit is made to a plot of the row of the image versus the inverse of the depth (Fig. 62). This takes advantage of the perspective transform and its relationship to $1/Z$. If all the tracked points were located on the ground plane and the depths returned were accurate, all of the points would fall on this line. A full least squares plane-fit can be made to the points in three-dimensional space, but in the experimental runs used thus far it appears unnecessary. The points above the line are the potential obstacles, those points furthest above the line being the most likely obstacles. Essentially if a point in

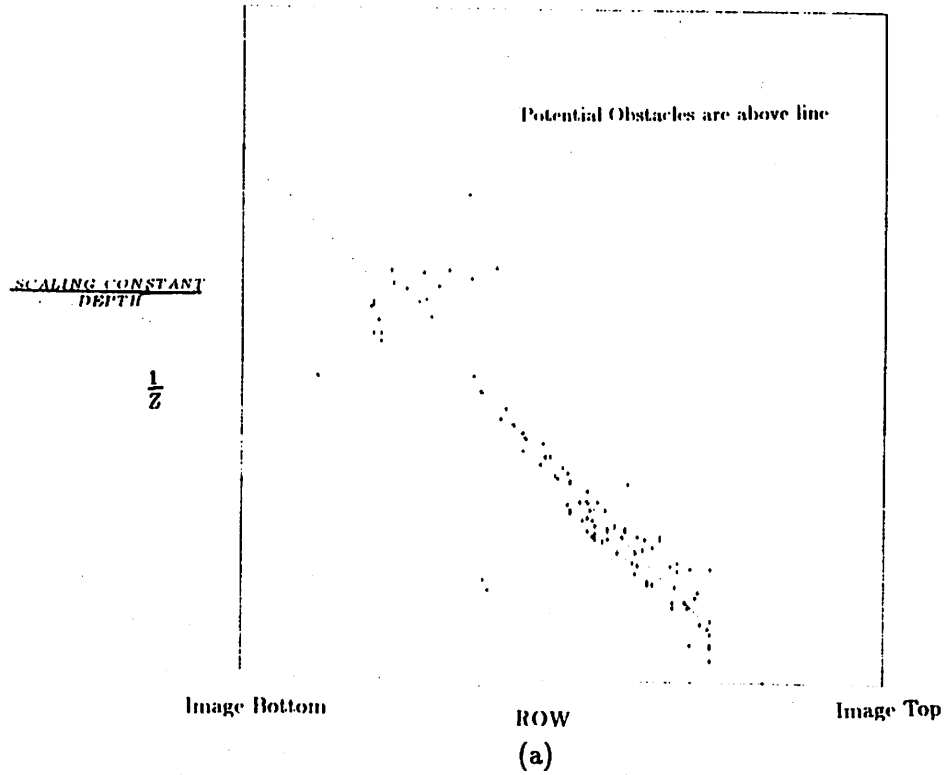
an image row is closer than a point in that same image row which is on the computed ground plane (based on the least squares fit), it is marked as a potential obstacle. In other words, points on potential obstacles which are off the ground return closer depths than do other points on the same image row which are on the ground plane. The taller the obstacle, the greater the difference in depth between the obstacle's point and a point on the ground plane on the same row. Thresholding, based on the distance of the point from the least-squares fit line, is then performed on the candidate points (typically the 30-50% of the points with the greatest difference in the fit to the line that are above that line), returning the obstacles (Fig. 63). Occasionally false positives arise, but usually all the close to mid-range obstacles are detected. An alternative approach would be to determine several points on an extracted region and use those points to compute surface orientation; when such orientations are found to be vertical they represent potential obstacles.

The returned obstacle data can then be associated with **avoid-static-obstacle** schemas and used within the confines of the motor schema manager for navigational obstacle avoidance. Unfortunately, the algorithm is too slow on our current hardware to be used for real-time navigation. See Chapter 8 for details of the off-line experiments using this system.

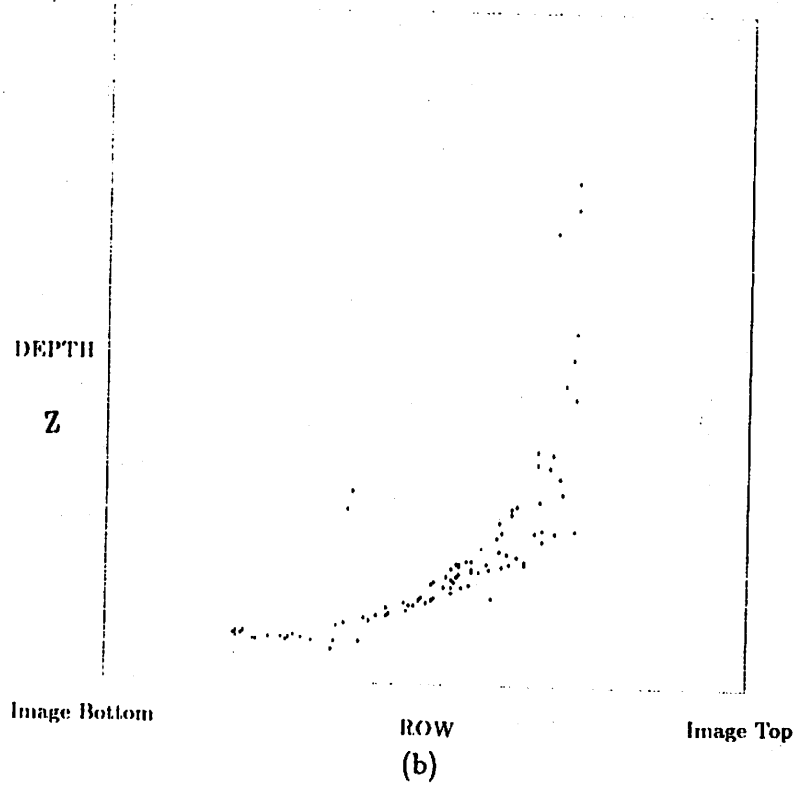
§2.4 *Interest Operators*

Interest operators are used in computer vision to pick out pixels associated with regions of high curvature and contrast. The Moravec operator [93] and the Kitchen-Rosenfeld gray-level corner detection interest operator [65] are two well-known examples. The depth-from-motion algorithm, described in Section 2.3, uses an interest operator (currently Moravec's), to determine the points on which to run the correspondence algorithms for registration from frame to frame, and the points to initiate correlation tracking in future frames.

Interest operators are quite primitive as a stand-alone method for obtaining information for navigation. Their primary advantage is speed. By combining knowledge available from long-term memory with image data, it becomes possible to use interest operators to *confirm* the position of landmark corners. A clear-cut example would be the position of a building corner against the sky (Fig. 64). When combined with knowledge from the robot's spatial uncertainty map and object size and location from LTM, this method can



(a)



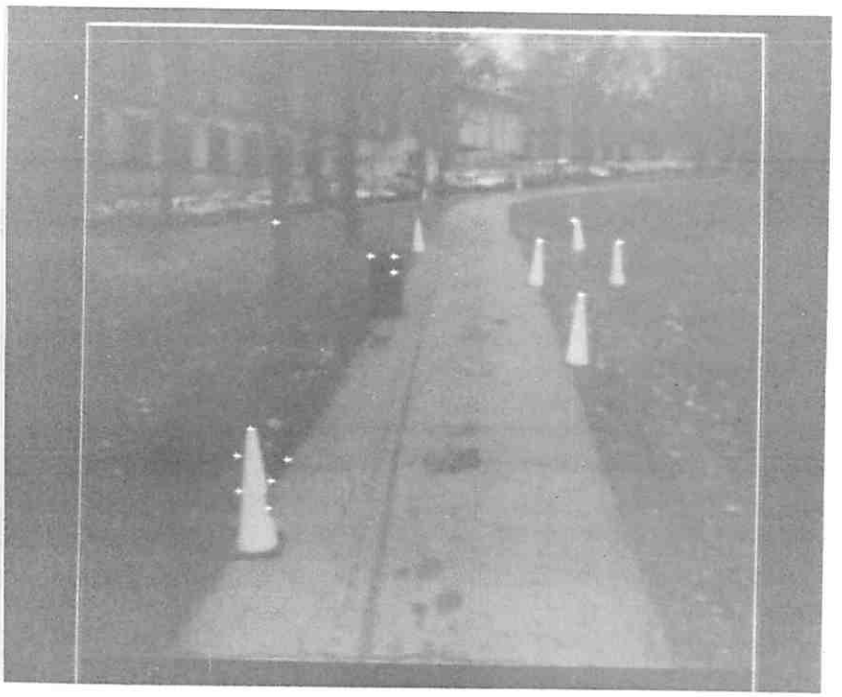
(b)

Figure 62: Obstacle extraction

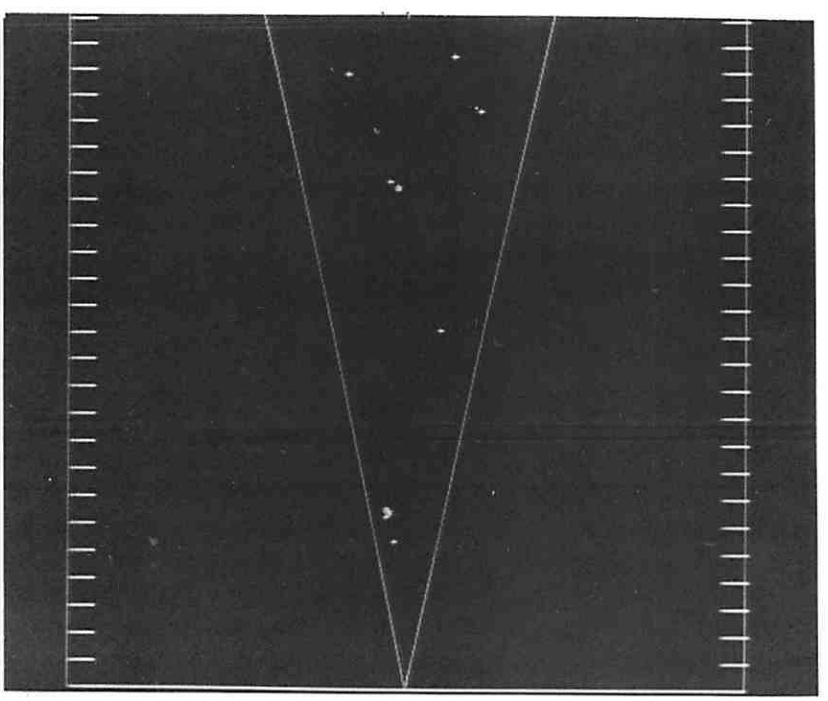
a) Image Row versus depth plot.

b) Image row versus inverse of depth plot.

Least squares line fit to points shown.



(a)



(b)

Figure 63: Extracted obstacles
a) Points on extracted obstacles (thresholded at 30%).
b) Depth to extracted points. The view is of the ground plane from above, with the robot at the bottom center and the V representing the field of view.

be used in restricted circumstances to confirm the position of a real corner as predicted by the line-finding method (Sec. 2.1). This information can then be used for spatial uncertainty management (Chapter 7). A succinct description of the Moravec operator appears in [18] for those readers unfamiliar with its operation.

As the interest operator provides a measure of distinctiveness (how different the pixel region is from its surroundings), the Moravec operator can also be used as a trigger event for spawning **avoid-obstacle** schema instantiations. When distinctive events occur against the relatively unchanging road backdrop, this indicates a potential obstacle. This low-cost focus-of-attention mechanism permits the concentration of higher-cost computational effort in such likely situations.

§3. Ultrasonic algorithms

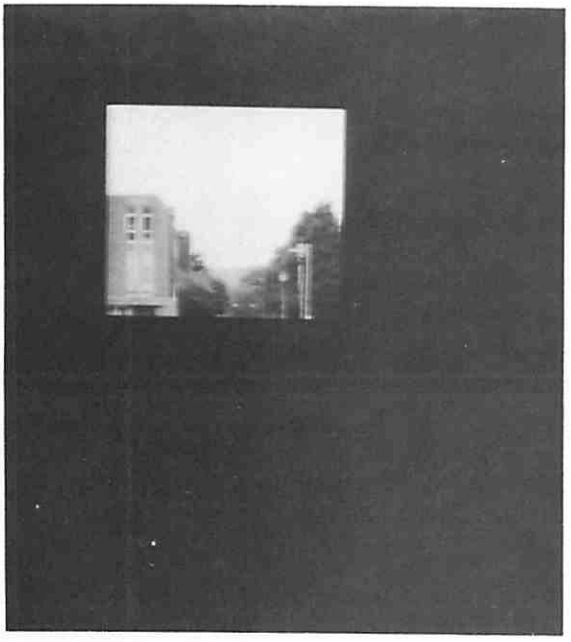
HARV is equipped with a ring of 24 ultrasonic sensors. It should be realized that ultrasonic data is poorly suited for many purposes. Using ultrasound has been likened to "standing in a room completely filled with mirrored objects and having only a penlight glued to your forehead as a source of light: specularity abounds and many surfaces are not visible" [30]. Serious problems involving reflectance and dispersion are present with this sensor modality. Nonetheless, researchers are spending considerable effort trying to utilize ultrasound as a viable means of environmental sensing for mobile robots [e.g. 45,84,43,30]. Drumheller's paper [43] in particular presents an excellent discussion of the limitations for this type of sensor. Having a firm grasp on the problems associated with this data form, ultrasound is used for a limited, although significant, role in AuRA.

HARV's 24 ultrasonic sensors (Polaroid laboratory grade) are controlled by a Z-80 microprocessor and are fired in three banks of eight sensors each to avoid interference. The sonar time-of-flight is converted onboard the vehicle to distances to a surface in tenths of a foot. The limit for detection is 25.5 feet away from our vehicle. No compensation is made for air temperature.

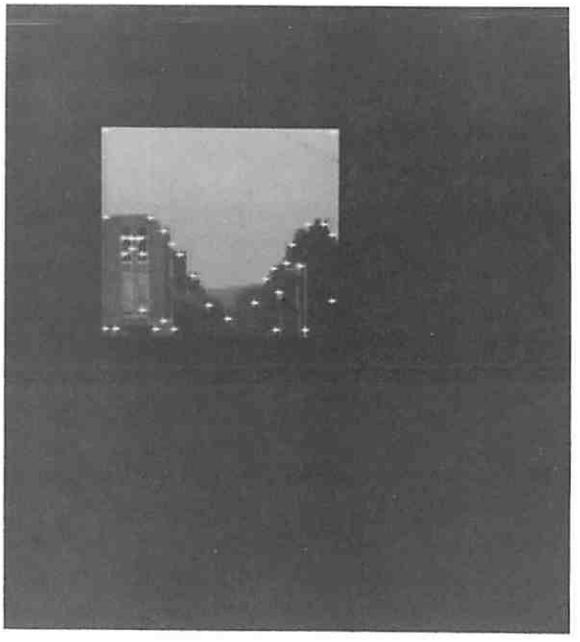
§3.1 *Obstacle avoidance*

The most common mobile robot application for ultrasound is in short-range obstacle avoidance or proximity sensing. For this system, the simplest and most appropriate use

(a)



(b)



(c)

Figure 64: Interest/Distinctiveness (Moravec) operator for localization
a) Original image.
b) Windowed image.
c) Results on finding building corner.

of ultrasonic data is to monitor the ongoing path of the robot and see if vision or any other sensor has missed a path-blocking obstacle. In essence, it serves as a safeguard or fallback system for collision avoidance. This application is fairly straightforward.

In the schema-based system described in Chapter 8, ultrasonic readings are associated with individual motor schemas. If the sensor indicates the presence of an object within the predetermined sphere of influence set by the **avoid-obstacle** schema, a repulsive velocity vector is produced. The magnitude of the vector is determined by the function decreed by the motor schema while the vector's direction is away from the sensed surface. All these vectors, plus any others due to other active motor schemas, are combined by the **move-robot** schema and transmitted to the robot's motor subsystem.

§3.2 *Door finding*

A separate strategy exists which uses ultrasonic data to detect a wall opening and then conduct the robot through it. It works essentially by first locating the wall and moving the robot in a position parallel to it. The robot then maintains an even distance from the wall while following it. The robot monitors for a large sonar reading increase in the direction of the wall coincident with the appearance of a door.

Once the opening is detected, the robot continues moving, waiting for the wall to be detected once again (indicated by a marked drop in the sonar reading). After the wall has been redetected, the robot splits the difference between the two detected wall ends, dead-reckons its position backwards, turns 90 degrees towards the wall and moves through the door.

The schema-based experimental system described in Chapter 8 can execute door entry in a different manner. Repulsive fields generated by the wall itself plus a velocity vector pressuring the robot through the wall replace the hard-coded approach described above. Essentially each sensor that returns a point within the prescribed sphere of influence for an **avoid-static-obstacle** schema generates a velocity vector in the direction away from that point and in magnitude proportional to the closeness of the obstacle. This is counterbalanced by the pressure of the **move-ahead** SI directing the robot to move into the wall. When the door opening appears, the obstacle avoidance pressure lessens and the robot moves into the doorway. As this technique is more consistent with the method of combining primitive motor behaviors to arrive at more complex ones, it is the method

of choice. See Chapter 8 for more information.

§3.3 *Panic sensing*

Ultrasonic sensing is valuable in protecting the robot from immediate danger. Imminent collisions due to obstacles missed by the vision system can cause the system to stop immediately (*freeze*). Moving objects bearing down on the vehicle can cause it to take evasive action (*flee*). The details of both these ultrasonic based processes appears in Section 6.2.

§3.4 *Localization*

Work on localization using sonar (locating the robot's position within the environment) has been progressing. Moravec and Elfes [92,45] perform a matching algorithm on sonar maps to try and position the robot. A variant and extension of this approach is used to supplement visual information for building STM in the AuRA (see Chapter 4). Miller [84] uses path planning that takes into account the positions that are most readily distinguishable for ultrasonic data.

Based on the premise that in a dynamic environment, with unmodeled obstacles possibly appearing and disappearing at any time and at any point in free space, the assumption that the detection of a surface tells us something useful about the robot's global position cannot be made. Ultrasonic data cannot make a qualitative distinction between what types of objects are sensed. It will be assumed that the absence of an expected or predicted surface tells us more than the presence of one. It must be remembered however, that certain critical signal angles must be adhered to if the data is to be considered reliable. If the angle of the signal relative to the wall is too small, the signal may skip off the wall's surface only to return from a more distant point (similar to a bank shot in billiards). Drumheller [13] describes how to limit the sonar data accordingly. By so doing, the uncertainty in the robot's position can be altered (by appropriate modification of the spatial uncertainty map) through the use of ultrasound.

§4. Shaft encoders

Conventional robot systems rely heavily upon the commands given their motors to produce expected changes in the robot's system. The problem with mobile robots is that if a command is given for the robot to move 10 feet, the wheels of the robot will be measured to have rotated the equivalent of a translational movement of 10 feet (by the shaft encoders), but the robot will not necessarily have moved that much. Wheel slippage due to poor traction or uneven tire inflation can produce significant deviations from the commanded movements. The use of terrain modeling can help establish reasonable ranges for errors, but even this is not foolproof. Changes due to a floor being waxed one day and dirty the next, or dewy grass in the morning versus dry lawn in the afternoon can pose serious problems for any *a priori* assumptions made about traversability. Consequently, caution must be used in the interpretation of shaft encoder data. Specific experiments have been performed to determine just how significant the conditions mentioned above are in affecting the translational error (see Chapter 7).

§5. Other desirable sensors

AuRA is an open architecture in the sense that it can readily absorb other sensor modalities. If funds permitted, other sensors could be added to provide greater accuracy or new information for the motor schema system to work with.

An inertial navigational guidance system could supplant the dead-reckoning system of the robot. Inertial guidance measures the actual rotations and translations of the vehicle as opposed to shaft encoders counting the number of wheel turns. The significance of this in limiting uncertainty and improving landmark prediction cannot be overemphasized. Unfortunately such a system is extremely expensive.

Inclinometers are desirable for several reasons. Information on topography could be obtained and correlated against an extended meadow-map representation of the world. Motor schemas for literal "hill-climbing" could be built (i.e. **move-up, move-down, stay-on-even-keel**). Registration of image sequences could be aided by knowing the actual differences in roll and pitch between frames. Distortions in expected landmark positions could be foretold by measurement of the camera's tilt. Inclinometers are not

overly expensive and are a suitable short-term goal for addition to AuRA.

A laser scanner for providing depth to environmental surfaces is highly desirable. This active sensor could supplant the computationally expensive and currently fragile depth-from-motion algorithm. Once again, however, cost is a factor. Several research groups are working with laser scanners nevertheless [141,94] and perhaps in the near future these sensors will be more affordable.

§6. Perception subsystem

Several disjoint topics all related to the perception subsystem constitute this section. Some are related to the general AuRA architecture (Fig. 2), others are more specifically concerned with the first pass implementation (Fig. 3). These topics include sensor processes, panic processes, and camera calibration.

§6.1 *Sensor processes*

Sensor processes serve as the gateway to both the clipboards described in Chapter 3 for the first pass implementation (Fig. 3) or the VISIONS system in AuRA's more general form. The role of sensor processing is to convert the raw data into a form that is readily integrated into the available representations (e.g. STM or schemas). The number of turns of a motor shaft or the time of flight of an ultrasonic return is of little value without some preprocessing.

Sensor processing in the case of vision involves the digitization of the incoming video image from the camera. This is currently performed on a Gould IP8500. Image resolution is also reduced from 512 by 480 pixels to a 256 by 256 pixel image. In some applications it may also be desirable to temporally average multiple frames to minimize the effects of noise, or to perform some form of smoothing prior to the image being posted on the clipboard.

Ultrasonic sensor processing utilizes the Z-80 microprocessor onboard the DRV to convert the time of flight of the ultrasonic return to a distance measured in tenths of feet. The hardware and firmware for this task were provided by Denning Mobile Robotics with the vehicle. Although, it does not take into account air temperature and other factors which can affect the result, the values returned seem adequate for the purposes they are

used for. The results are reported for each of the 24 sensors in tenths of a foot. The firing sequencing is also controlled to prevent overlap of return signals and resultant false readings.

Finally the DRV, again using another separate dedicated Z-80 microprocessor, converts the shaft encoder data into a form that is easier to interpret than simply the number of motor revolutions. Cartesian X and Y values representing the "distance" traveled (based on the number of drive motor shaft revolutions) is reported to the clipboard. Information regarding the direction the robot is facing is also available based on the steering motor encoder data. From the uncertainty treatment described in Chapter 7, it should be obvious that the "distance" the shafts have rotated does not correspond accurately with the actual location of the robot. Nonetheless it does provide valuable data which are used to constrain the spatial uncertainty map.

§6.2 *Panic processes*

Panic processes serve to alert the robot in a reflexive manner to potentially dangerous situations. The warnings and resultant commands issued by these processes bypass higher level processing and are communicated directly to the vehicle interface. The time saved in emergency situations can be critical to a successful response.

Several types of panic processes are conceivable. Most have animal behavior parallels. These include "freezing" in place in response to some unexpected event or "fight or flight" behaviors based on the approach of another entity. For the first pass AuRA implementation, the panic-stop ("freezing") and panic-avoid ("flight") behaviors have been constructed using ultrasonic data.

The panic-stop process continuously monitors the data posted on the clipboard by the ultrasonic sensor process. If any reading in the direction of the robot's motion is below some predetermined threshold (e.g. 2 ft), a continuous stream of stop commands is sent to the vehicle interface. This immediately prevents the robot from moving by counteracting any old motor commands currently being executed and preventing the implementation of any new motion commands. The panic-stop process continues to monitor the incoming sonar data and only when the offending obstacle is removed as evidenced by safe sonar readings does it allow the robot to continue moving. Information is also posted on the clipboard for other processes (such as the navigator and motor schema manager) to react

to the blockage in a more timely manner.

The panic-avoid process constantly polls incoming ultrasonic sensor data to see if an object is approaching the robot in a head-on direction. It accomplishes this by comparing successive time-stamped sonar readings from the clipboard, noting the difference in distance when compared to the robot's velocity, and determines if the robot is being approached. When a predetermined threshold for time-to-contact is reached, the robot turns 90 degrees and moves rapidly to the side until it no longer senses the object approaching. This is an evasive action maneuver. The panic-stop process remains in play preventing it from crashing into a wall or other object. The panic motor commands override any other commands already executing or waiting to be executed within the robot. Several consecutive confirming sonar readings are required to trigger the panic-avoid process, minimizing the likelihood of spurious ultrasonic data causing this event. The result is somewhat comical in appearance, but it is potentially very useful behavior for a mobile robot operating among moving equipment. In those cases, it is not enough for the robot to stop. It must get out of the way or it may possibly be damaged by (or damage) the moving body. Once the panic-avoid episode is completed, special procedures are invoked to allow the robot to regain its bearings and then satisfy the previously specified navigational goals.

The major drawback to the current implementation of the panic processes is the slow transfer rate for the ultrasonic data over the serial line to the VAX. A future solution would be to embed these routines onboard the DRV's MC68000 processor eliminating the communications delay to the VAX. The analogy to reflex arc behavior becomes even more apparent in such a circumstance.

§6.3 *Camera calibration*

Much work has been performed in the area of camera calibration, making the solution to the problem fairly straightforward. Monocular video systems are particularly well understood. Kak [62] and Thompson [123] both present excellent descriptions of the mathematics and techniques required for camera calibration. Stereo camera calibration has been discussed by many authors including [93,70]. Even a trinocular video system calibration has been described [72].

The mobile robot project was fortunate to receive camera calibration software from

the University of Rochester. Rigoutsos describes the process and constraints for this work in [111,112,113]. The code has been slightly modified for use in the calibration of the camera mounted on the vehicle.

A known three-dimensional Cartesian map is made of several readily recognizable points in our hallway, such as door corners, light fixtures, and hall borders. (Fig. 65 and Table 2). The robot is then rolled to a position in the hall. An image is taken. The image coordinates for each real world point are found (currently 20 points are used). Rigoutsos' algorithm operates on this data, producing the 4 by 3 calibration matrix. This matrix is used to convert points whose position is known *a priori* in (x,y,z) space relative to the robot to their expected image (u,v) coordinates. The prime user of this matrix is the Expecter process in the uncertainty management subsystem.

§7. Summary

Perceptual strategies are embedded in motor schemas to provide the necessary information for the robot to interact intelligently with its world. These strategies can take many forms but are based on the premise of action-oriented perception. This concept allows special-purpose perceptual techniques to be concentrated on individual components of the navigational process. These include methods for the detection of obstacles, pathways and landmarks.

The vision algorithms used in AuRA encompass a wide range of computer vision techniques. These include primitive interest operators, more sophisticated line-finding and region segmentation algorithms, a multiple frame depth-from-motion algorithm, and potentially, a scene interpretation system. Each approach has its purpose, advantages and disadvantages for use in mobile robot navigation. In all cases, however, versions of vision algorithms have been developed which extract image features rapidly (at the expense of reliability in some cases). In addition, the control of all algorithms attempts to use a top-down strategy of restricting processing to windows based upon LTM or previous frames. In this manner, real-time processing may be achieved for certain interesting navigation tasks that might not have been feasible until more powerful parallel hardware arrives.

Some of the specific contributions made in this chapter involve the adaptation of vision algorithms (FLF, FRS, depth-from-motion, interest operator) to mobile robot

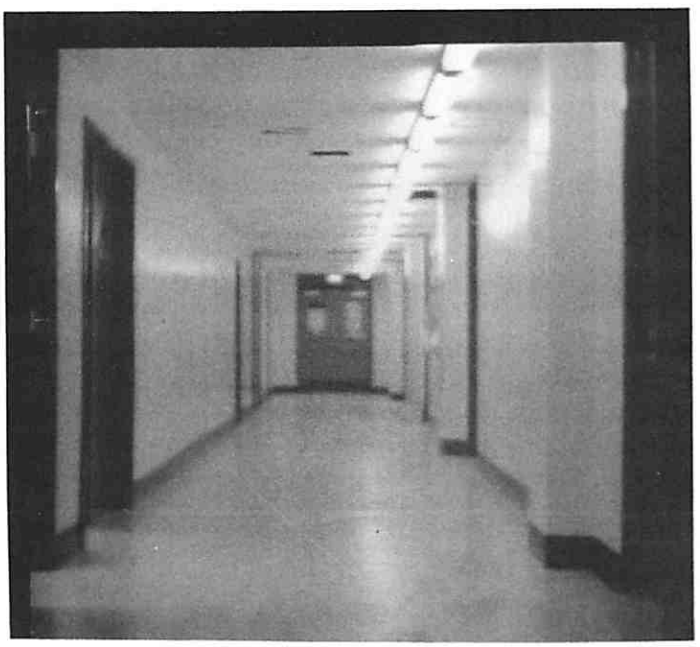


Figure 65: Camera Calibration Scene

Table 2: Camera calibration data

Point	X	Y	Z	U	V	Description
1	2.125	86.0	374.125	87	391	Corner of door
2	100.00	14.0	393.0	395	152	Top of outlet
3	102.063	82.0	529.00	361	337	Tag on A207
4	0.000	37.5	782.00	171	236	Door knob
5	52.5	46.75	1215.00	251	245	Bar on right door
6	80.5	95.25	366.875	348	422	Corner of tile
7	92.375	6.0625	535.625	340	162	Corner of strip
8	73.5625	93.125	352.875	328	423	Box on ceiling
9	73.25	92.875	473.375	307	386	Box on ceiling
10	2.00	5.9375	747.0	167	182	Frame of door
11	2.00	5.875	298.375	40	100	Door frame
12	32.75	95.75	380.25	192	415	Tile w/metal frame
13	51.00	10.375	1217.25	248	210	Corner of door
14	92.125	6.0625	1063.875	296	201	Corner of strip
15	102.5	6.5	271.5	423	88	Corner of strip
16	2.25	86.0	298.5	49	425	Corner of door
17	2.00	6.25	374.5	85	131	Black strip
18	2.5	83.5	747.25	168	308	Corner of door
19	44.0	95.5	1205.5	243	294	Corner of exit sign
20	5.75	68.75	618.5	149	294	Corner of sign

Calibration matrix

$$\begin{vmatrix} 5.196073 & -0.912248 & 0.946008 & -130.762170 \\ -0.829986 & 4.604287 & 0.879309 & -25.105374 \\ -0.003328 & -0.005164 & 0.003582 & 1.000000 \end{vmatrix}$$

navigation. This necessitated the development of representations (LTM) suitable for providing expectations for these algorithms. Application of the FLF and FRS algorithms to actual image sequences enabled path following behavior to be undertaken (Chapter 8). The actual perceptual algorithms are only a part of the overall system (in some cases a small part) and additional code involving the production of expectations, image sequencing, vehicle servoing, communications, etc., had to be produced before viable experiments could be undertaken. It's a big step from an image on a video monitor to intelligent motor action.

The ability of each of the vision algorithms to function in a domain as unconstrained as outdoor navigation varies considerably. The fast line finding algorithm's path-following software is the most robust. The ability to work with highly fragmented lines even under conditions of partial occlusion of the road edges makes it very versatile. Its capacity to be tuned based on expectations from either previous images or knowledge from LTM, couples it tightly with AuRA's design philosophy of action-oriented perception.

The fast region segmentation algorithm holds great potential when it is extended into the color spectrum. HARV only has the capability to digitize monochrome images currently, but that will change shortly. Then full advantage of the spectral characteristics of environmental objects will be exploited. Nevertheless, even in its current form it can be applied to useful tasks such as path following (Chapter 8).

The depth-from-motion system is a major project that is receiving considerable research effort at the University of Massachusetts. Preliminary results regarding frame-to-frame point tracking are very promising, but the difficulties remaining in FOE extraction and image registration need to be solved in order to produce a robust passive navigation system. Preliminary results using HARV as the test vehicle for this algorithm are presented in Chapter 8.

Other sensor forms are used within AuRA, taking advantage of their distinguishing characteristics. Ultrasonic data is well suited for obstacle avoidance and can serve as a confirmation mechanism for landmark recognition. The processing speeds for ultrasonic data make it suitable for conducting real-time continuous motion navigation with HARV (Chapter 8). As limited as this sensor modality is, it can still be used quite advantageously. Shaft encoder data provide constraints on the limits of motion that the robot has undertaken (Chapter 7). Additionally, they provide estimates of where a navigational

goal extracted from LTM is located relative to the robot's current position. As newer and better sensors become available, AuRA's design will easily accommodate them, extending this system's capabiilties even further.

A major drawback of the vision algorithms is the time required to process them. Although low resolution images may be suitable for industrial robotic applications, natural scenes require more detail. A 256 by 256 image contains a very large quantity of data. The time required to process such an image on our existing hardware forced us to use "lurch" mode for the visual navigation experiments. Continuous motion was possible, however, using ultrasonic data. Chapter 8 describes the experiments that were performed using both vision and ultrasound with our mobile robot HARV.

CHAPTER VII

SPATIAL UNCERTAINTY MANAGEMENT

Mobile robots have difficulties that are not found in more conventional robot systems. Robotic manipulators, through the use of inverse kinematics, and the fact that their position relative to the workspace is typically known to a high degree of accuracy (on the order of fractions of millimeters) using high resolution encoders, can in many cases ignore the uncertainty in the robot's position itself. That is not to say that uncertainty is a solved problem for this domain; quite the contrary. Most uncertainty in assembly-oriented tasks arises from the relationship of the manipulated parts to the modeled world rather than that of the robot to the world.

Automatic guided vehicles, such as wire or stripe following robots, have more uncertainty to contend with, but the problem is essentially one-dimensional. As the robot must maintain adherence to a line, the only uncertainty is in where the robot currently is located along that line. By embedding optical landmarks along the path or through the use of infrared beacons, the robot's world position can be readily confirmed.

Mobile robots are plagued with uncertainty problems. Uneven traction due to the terrain or tire inflation, and drift due to problems within the drive train can rapidly lead to disorientation of the robot relative to its modeled world. The mobile robot must contend with a minimum of three degrees of freedom (with significant uncertainty in each) - 2 degrees of translation, which can be represented as x and y coordinates in a Cartesian world model, and 1 degree of rotation (assuming a planar world), the actual heading of the robot relative to known compass headings.

Most mobile robot systems concerned with uncertainty management have handled this problem in the context of environmental acquisition, where the robot's world is not modeled ahead of time, but instead is built from sensor observations, typically sonar. This chapter describes an approach to uncertainty management that uses an explicit

representation of the robot's positional and angular uncertainty relative to an *a priori* model of the world. Vision is the principal means for reducing this uncertainty.

The goal of the spatial uncertainty management system is to guide expectations for sensory modules in AuRA, reducing the amount of processing required to determine the robot's whereabouts. The world representation used is the meadow map described in Chapter 4. Embedded within this map are data essential for uncertainty management. This includes relevant terrain data for a statistical approach to uncertainty growth, and visual landmarks which are used to guide perceptual processing for ultimate recognition and resultant reduction of uncertainty.

The remainder of this chapter is divided into the following sections. Section 1 presents a review of related work in the field of mobile robot uncertainty management. Section 2 describes AuRA's cartographic spatial uncertainty management subsystem and its relationship to other components of the AuRA architecture. The specific representations used and the process that manages the growth and reduction of uncertainty, based on the vehicle's motion in the world coupled with visual feedback, is detailed in Section 3. The process of landmark selection is described in Section 4 and that of landmark recognition in Section 5. Simulation results are presented in Section 6, while actual experimentation using the mobile vehicle is presented in Chapter 8. A summary and evaluation of the uncertainty system concludes the chapter.

§1. Related work in uncertainty management

A hallmark paper addressing uncertainty in robotics, more concerned with assembly than with navigation, was written by Brooks [28]. His discussions of visual map making [25,29], where he argues against the use of a global map for a system which acquires (learns) its environmental model solely from vision, are more pertinent to mobile robotics.

A fundamental problem, Brooks states, is that worse case error must be used with an absolute coordinate system. There are significant limitations if the entire world is to be modeled in a single world-oriented frame of reference. AuRA proposes two frames of reference: first, an egocentric model which is the basis for sensor data acquisition; and second, a world model which is used to represent *a priori* knowledge (a partially modeled world). A model based on knowledge of the terrain reduces the dependency on worst

case analysis.

Polyhedral models are to be avoided, according to Brooks, due to their poor time-performance and tendency to break down in real world situations (as the real-world is not polyhedral). INRIA's HYPER system [17], described below, argues against this. In AuRA, successful counter-arguments can also be made to this claim by recognizing particular classes of landmarks, subdividing the processing over multiple active schemas, and searching for landmark features in restricted portions of the image. Finally the desirability of vision over less informative sensors such as sonar is shared by both Brooks and myself.

Brooks' system uses, as does AuRA, shaft encoder data, visibility analysis, and visual landmark recognition. His representations include both "freeways" (generalized cylinders), a free-space representation method, and meadows (circles in [29], convex regions in [25]). His treatment of uncertainty involves the generation of 3D solid "uncertainty manifolds" arising from an uncertain transform. Brooks' approach combines ("cascades") these uncertainty manifolds, which arise from sequences of uncertain transformations, to provide information about the robot's current location relative to sensed landmarks. Projections of the uncertainty manifolds are represented in 2-dimensional space as circles that grow as the robot moves when there is no feedback available from landmark recognition. AuRA instead uses a convex polygon representation (circles represent the worst case analysis) due to the asymmetric nature of motion error. A fundamental difference in AuRA's approach lies in the backprojection of the uncertainty into world coordinates (i.e. the construction of the spatial uncertainty map) which is then overlaid on top of the absolute world model (the meadow map) to provide expectations of previously unseen landmarks whose whereabouts are known only in world coordinates.

Chatila and Laumond [33] independently developed an approach called "fading" that is similar to Brooks' method. This technique employs circular approximations for uncertainty and is used in a sensor-acquired (learned) meadow map, closely related in structure to AuRA's; indeed, this work for HILARE provided an incentive for the extended meadow map representation of *a priori* knowledge used for world modeling in AuRA. HILARE is concerned with acquiring its own world model and hence needs to associate a new frame of reference with each newly discovered landmark. AuRA assumes the existence of landmarks in its meadow map, added by the cartographer from available *a priori* knowl-

edge. In HILARE, landmark recognition is used to update the robot's model of the world as much as its own position relative to the perceived world. It is entirely possible that the landmark's position, and not just the robot's, must be updated. Focus of attention mechanisms, as found in AuRA, are not treated either in Brooks' paper or this one.

Smith and Cheeseman [118] have developed a basis for the handling of spatial uncertainty in the mobile robot domain. Drawing on Kalman filter theory, they include methods for merging (combining evidence from independent parallel measurements to improve the certainty over any single measurement) and compounding (chaining sequential uncertainty transformations). Smith and Cheeseman's paper, as in the two cited above [29, 33], handles the transformations over multiple frames of reference, each associated with the vehicle's position at the time of its observation. The goal is to describe landmark observations in terms of previously sensed, but uncertain, landmarks rather than to represent newly acquired data in terms of a world model. The choice of which landmarks to use is guided by the uncertainty in previous landmark recognitions. Although an elegant mathematical technique is developed for this purpose, it relies heavily on fully independent sensings and thus does not derive, in our estimation, the full benefit available from landmark tracking.

Active sensing using infrared beacons placed at strategic locations, (e.g. available for the commercial Denning Sentry and other similar robots) can be used to reduce positional uncertainty. Infrared beacons are also used in HILARE [52].

Fukui [49], in one of the first systems to use passive vision for positional uncertainty management, used a special landmark to position a robot in interior scenes. This early work placed an artificial diamond-shaped landmark of high contrast in locations that favored its detection so as to reduce the spatial uncertainty of the vehicle.

One of the more sophisticated systems developed thus far for maintaining the spatial uncertainty of a mobile vehicle arises from work performed at the University of Maryland [4]. The system is composed of three separate modules. The MATCHER identifies landmarks in an image by using a Hough transform based on an edge template of the landmark in question. The FINDER controls the pan, tilt, and zoom mechanism for the camera based on available spatial uncertainty data. The SELECTOR chooses good landmarks from a database that enables the vehicle to reduce its positional uncertainty.

A circular uncertainty region, called a disk, is used to model the positional uncertainty

of the vehicle in global coordinates. These data, in conjunction with the actual structure of the landmark, the angular uncertainty in the vehicle, and the uncertainties in the pan-tilt and zoom mechanisms, are used to constrain the direction and focal length of the camera. Landmarks are actively sought by the system and are not derived as a by-product of other available images. The entire landmark must be present in the image for recognition with the Hough transform.

Although the Hough transform method is fundamentally different from the visual strategies currently used in AuRA (see Chapter 6), the geometrical development of the FINDER and AuRA's Expecter is similar, since both are used to predict where a landmark will occur in an image. In the FINDER, this information is used to mechanically drive the camera to actively seek out the landmark, whereas in the Expecter it is used to provide appropriate subwindows of the image. Maryland's SELECTOR chooses from all of the available landmarks in the database and is not restricted to consider only those lying in one particular field of view. AuRA's Expecter selects only those landmarks that are expected to be encountered in the direction of the robot's current motion. The Maryland system is a triangulation oriented system while AuRA's Expecter can use to advantage information derived from a single landmark. This is largely due to the asymmetry available within AuRA's spatial uncertainty map. Although it is desirable to control the focal length of the camera while searching for specific landmarks, no provision is made for this in the current implementation of AuRA. The fundamental reason for this is that if a single image sequence is to be used for path following, landmark identification, obstacle avoidance, and other tasks, it is not feasible to optimize the image for any single perceptual schema. By judicious selection of landmarks, multiple tasks (including multiple landmark recognition) can proceed concurrently without altering the orientation or focal length of the camera.

A final distinction between the Maryland system and AuRA arises from the source of uncertainty. AuRA's empirical approach for terrain modeling serves as the basis for the growth of the spatial uncertainty map. Maryland's system appears to be largely based on previous identifications of landmarks alone to constrain the positional uncertainty. Convex regions, similar to the spatial uncertainty map, are used in the development of the spatial uncertainty, but they arise from the triangulation effects of multiple landmarks. This polygon is then approximated by a circle for later use. A convex polygonal repre-

sentation is also retained for AuRA's representation of spatial uncertainty. Additionally a point center within the convex region is maintained by AuRA to indicate the likeliest location of the robot.

Work performed at INRIA in the development of the HYPER system [17], used to guide a robot arm to pick up occluded or poorly illuminated parts, develops important ideas for extension to the mobile robot domain. The use of polygonal representations for part (in our case landmark) recognition as well as scene modeling is stated to offer several advantages. These include:

- local information (in contrast to conventional robot vision measures that use global numerical features)
- Low storage requirements (compact)
- A general method is available for diverse parts (landmarks)
- Position and orientation sensitive (hence their recovery is feasible)
- Simple and fast vision operations

HYPER's success in part recognition provides a justification for polygonal landmark representation. The most important contribution of HYPER, however, is related to the rating of potential matches through a hypothesis evaluation mechanism using a quality metric.

Work at Yale [37,82] regarding the representation of spatial uncertainty in SPAM (spatial module) is of interest. McDermott and Davis represent spatial uncertainty with fuzziness, and particular locations of environmental objects relative to each other with fuzzboxes. The reduction of uncertainty is termed fuzz constriction. Although their work has been directed toward route planning, several of the concepts, including fuzzboxes (generally rectangular but allowed to have other shapes as well) to represent spatial uncertainty, appear generalizable. The other proclaimed strength is the support of multiple frames of reference through *frobs*, a hybrid representation for an object in a perceived frame of reference. SPAM does not specifically address the uncertainty associated with visual landmark recognition, but the representations and methodology it uses seem extendible.

Specific systems dealing with the uncertainty associated with particular visual strategies are also worth mentioning. At CMU [80,81], a stereo based system has been employed for visual guidance of robot motion. The triangulation uncertainty method constrains the position of the vehicle, using a Kalman filter approach (similar to Smith and Cheeseman above). Two models are maintained; a local, moving, robot-centered frame and a global coordinate system. Correlation between the perceived landmarks in the local frame and the known position landmarks in the global frame constitute the task.

Uncertainty analysis for depth-from-motion image sequences has been studied by Snyder [119] at the University of Massachusetts, and is being applied in AuRA (Chapters 6 and 8). Limitations on the image displacements anticipated from frame to frame constrain the search space for match points and thus expedite the depth extraction process.

§2. Uncertainty subsystem - An overview

The bulk of the spatial uncertainty management subsystem (UMS) lies within the cartographer's responsibility in the overall AuRA architecture (Fig. 2). A block diagram of the UMS appears in Figure 66. The UMS is tied to other components of AuRA through the clipboards, vehicle interface, navigator, pilot and motor schema manager.

The UMS consists of both data structures (rectangles in Fig. 66) and processes (rounded rectangles). The relevant data structures represented include the spatial uncertainty map itself, an identified landmark buffer, data from long-term memory (LTM) including terrain characteristics and landmarks, the schema database, specific clipboard data containing positional reports, and the command buffer within the vehicle interface. The processes include the uncertainty map manager, the components of the pilot concerned with **find-landmark** schema instantiation, and the Expecter that is used to predict landmark position in incoming images.

The overall flow of control within UMS can be described as follows. The pilot first receives information that the robot is to traverse a specific leg of the overall global path developed by the navigator (see Chapter 4). Available within the cartographer is an express representation (the spatial uncertainty map) of the robot's current position (bootstrapped at start-up or known from previous legs) including both the uncertainty in heading and spatial location relative to the global meadow map. Available in STM,

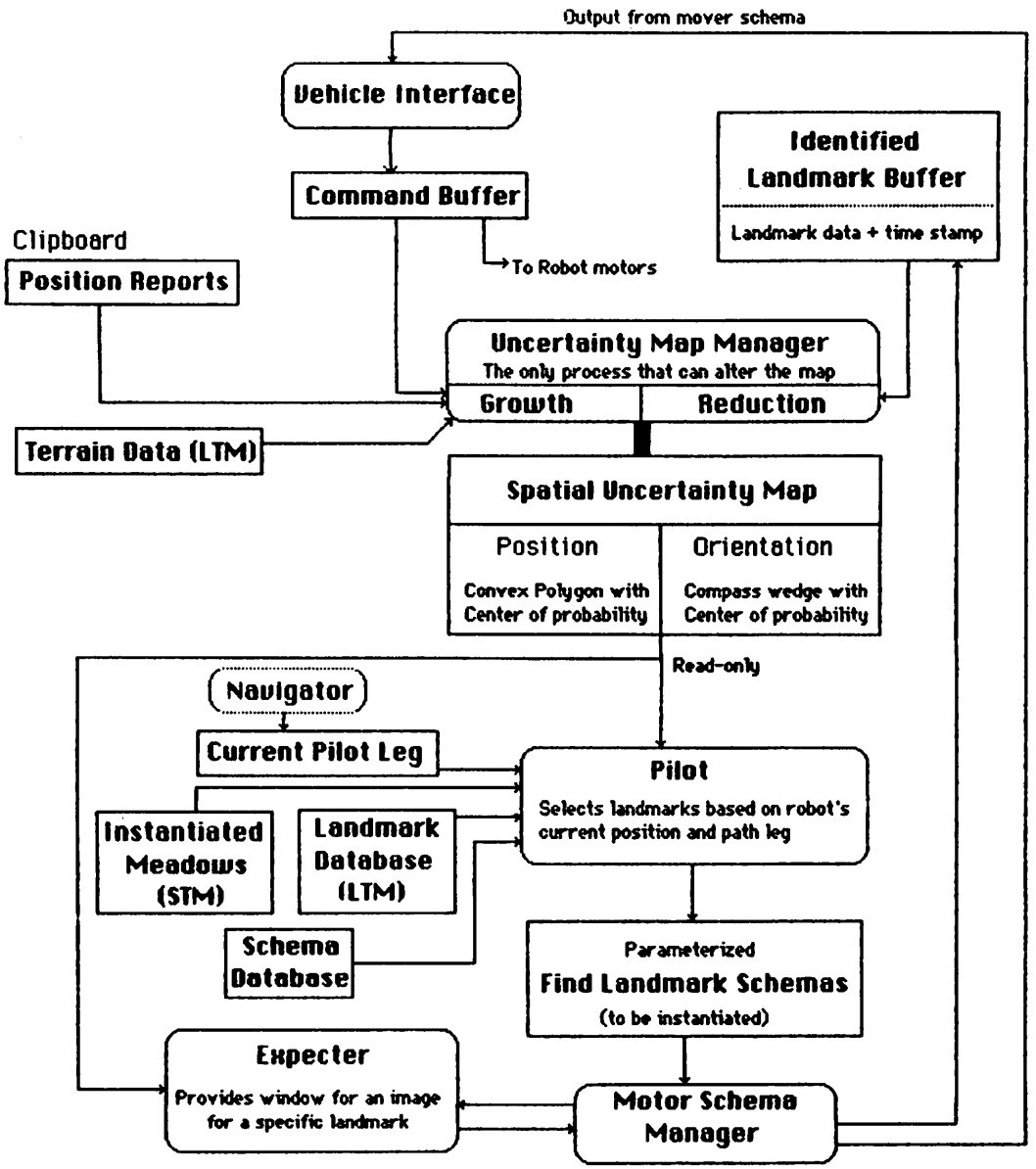


Figure 66: Uncertainty Management Subsystem

provided by another cartographic process, are instantiated meadows, those portions of the LTM map which are of concern to the vehicle during the piloting for this particular leg. This component of STM contains pointers to landmarks that are of potential value during this portion of the robot's journey. The pilot, acting on the available information, parameterizes **find-landmark** schemas obtained from the schema database and passes them to the motor schema manager for instantiation. There they remain, waiting for specific visual and positional reports to trigger their activation.

The robot generally begins its motion as a consequence of a **move-ahead** schema. As it moves, positional reports from the shaft encoders are fed to the uncertainty map manager. The uncertainty growth routines within the uncertainty map manager act on both this information and the characteristics of the terrain to increase the extent of spatial and angular uncertainty as the robot travels. This is usually done at the end of a leg or when a landmark is recognized. If no landmarks are recognized and the robot continues to move, eventually the spatial uncertainty of the robot would fill the entire map. It is essential that landmark recognition be accomplished to produce effective uncertainty management.

Based upon the robot's current uncertainty, the **find-landmark** schemas, when activated, make requests to the Expecter process to predict where in the image a landmark feature should occur. This restricts the perceptual processing associated with each landmark to reasonable limits. After the appropriate perceptual schema is run on that window of the incoming image, the result is passed to an evaluation function which determines whether or not the landmark has been recognized (i.e. exceeded its recognition threshold). Once a **find-landmark** schema has recognized the position of the landmark in the image, it posts its results in the identified landmark buffer. The uncertainty map manager uses this time-stamped information, after updating the growth of the uncertainty map based upon the likewise time-stamped position reports, to reduce the extent of the positional and/or orientation uncertainty of the robot (described in Section 3).

A feedback loop is achieved by the establishment of expectations based upon the current spatial uncertainty map and the subsequent recognition of landmarks within those established image boundaries, modifying the spatial uncertainty map. If no landmarks are recognized even though several have been predicted, the robot declares itself lost, stops, and then starts searching larger windows (and even rotating if necessary) in an

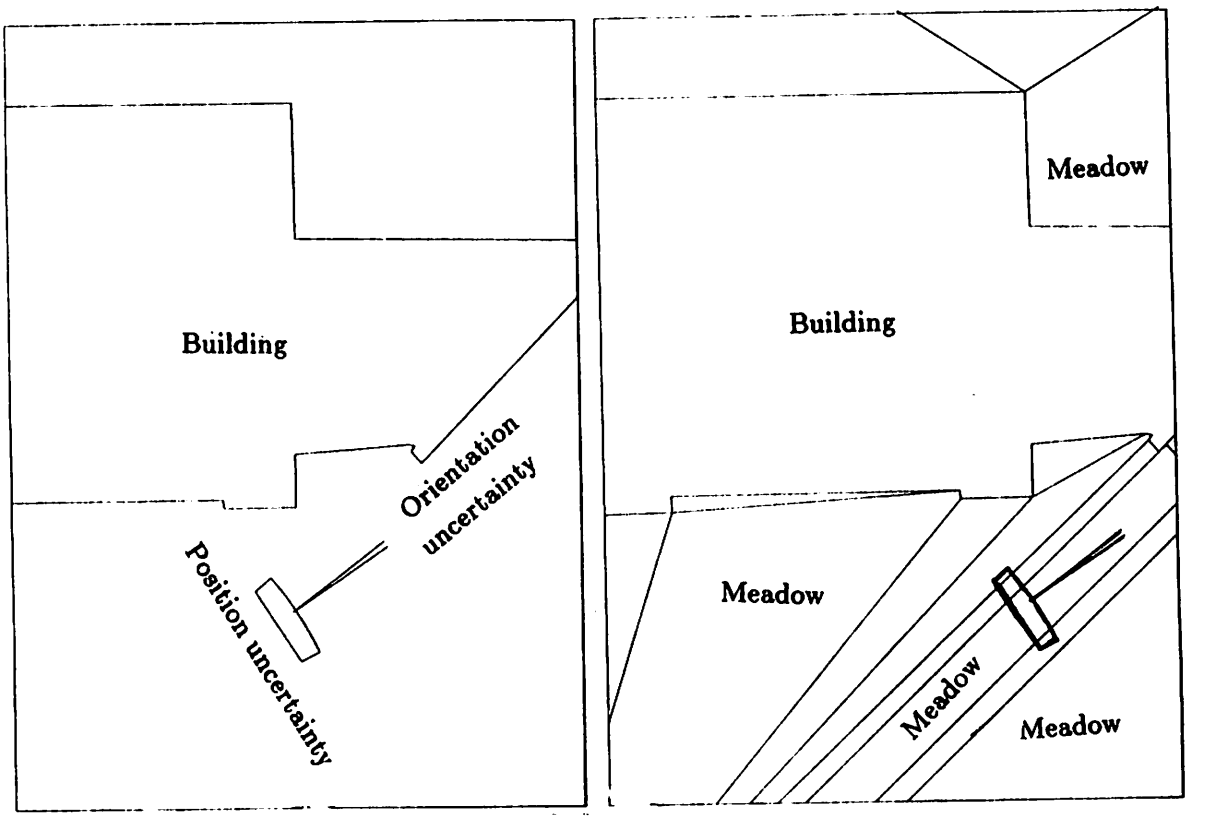
effort to encounter something familiar and recognizable relative to its world map. In the normal sequence of events, however, the robot does not change the camera pan, tilt or focal length during leg traversal (in contrast to the Maryland system [4] described in Section 1).

The two frames of reference that need to be reconciled are the egocentric perceptual representation provided by the video images and the global meadow map itself. The spatial uncertainty map provides the mapping from one frame to the other. UMS uses an approach to uncertainty growth based on empirical terrain statistics. Consequently, there is a finite, but relatively small, probability that the robot will be located outside of the bounds predicted by the uncertainty map. Back-up re-orientation procedures are important if the robot is to regain its bearings if this occurs.

Thus far, we have assumed that the uncertainty of objects located within the global map is nil. Although this is technically an invalid assumption, as there will always be some non-zero amount of error in the positional representation of a landmark, it is safe to assume that if these data came from accurate blueprints or maps that the amount of uncertainty is small to the point of being negligible when compared to the error resulting from the robot's motion. Nevertheless, it is feasible to explicitly represent each landmark's positional uncertainty relative to the global map and to use that information in the Expecter process and in the uncertainty reduction techniques within the uncertainty map manager. This will be discussed further when these processes are described in Sections 3 and 5 respectively.

The **find-landmark** schemas also play a role in producing motor behavior (see Chapter 5). In particular a **move-to-goal** motor schema needs a **find-landmark** schema to recognize the goal towards which it is directing the movement of the robot. In this instance the **find-landmark** schema serves a twofold purpose - to provide perceptual guidance for a specific motor behavior while concurrently providing information to reduce positional uncertainty by the UMS.

Figure 67 illustrates the relationship of the spatial uncertainty map, representing both position and orientation uncertainty, and the global map. Other examples are presented in Section 6. The sections that follow describe the details of the data structures and processes that are found with UMS.



(a) (b)

Figure 67: Spatial uncertainty map in context of global map

a) Convex polygon represents robot's positional uncertainty. Two lines extending from center of maximum likelihood of robot's location indicate extent of rotational uncertainty.

b) Close up of same view showing meadows in meadow map.

§3. Spatial uncertainty map and uncertainty map manager

The spatial uncertainty map and uncertainty map manager represent and maintain the spatial uncertainty present in the robot's position relative to the modeled world. The spatial uncertainty map manager is the only process which can alter the uncertainty map itself, although the uncertainty map has read-only availability for other processes such as the Expecter and pilot.

§3.1 *Spatial uncertainty map*

The uncertainty map consists of two components, one representing the spatial extent of positional uncertainty, the other the limits of heading uncertainty. The positional component is modeled as a convex polygon which is produced by the repeated application of uncertainty transforms on earlier uncertainty maps. The coordinate system for the points of the polygon is the same as that of the global map against which it is matched by processes like the Expecter. The spatial extent of the positional area represents the likelihood that the robot's position is located within this region to a fixed probability (typically 95-99% - two or three standard deviations assuming a Gaussian distribution). Although it is theoretically possible to model this region as a three-dimensional surface, this was not done, both for reasons of computational and mathematical tractability as well as the lack of a perceived advantage to such an approach. Nonetheless, a center point representing the robot's single most likely position is maintained within this uncertainty map.

The positional component described above tells us nothing about the direction that the robot is facing within that area, only the likelihood that it is to be found there. The second component of the spatial uncertainty map represents the uncertainty in orientation. A compass wedge indicating the limits of heading uncertainty relative to the global map constitutes this model. As before, a center of probability is maintained, in this case representing the most probable orientation. The wedge is modeled by assigning limits to the extent of both clockwise and counter-clockwise rotational uncertainty from this center point.

Previous approaches have typically used circular disks to model positional uncertainty (see Section 1). This was deemed inappropriate due to the asymmetric nature of uncer-

tainty growth and landmark recognition as described in the following subsections. To use a disk would require worst case uncertainty modeling for all situations. An example which illustrates this point nicely is the uncertainty in position as a robot moves down a path (Fig. 68). Visual feedback indicating that the robot is located on the road substantially restricts the amount of spatial uncertainty in the direction perpendicular to the road. Protracted movement causes large amounts of uncertainty to accumulate along the dimension parallel to the road itself.

Differences in the image window produced by landmark location predictions are quite pronounced if a disk model rather than a polygonal model is used. Expectations produced from the uncertainty map for landmark recognition (described below) can result in much smaller image windows if an asymmetric map is used (Fig. 69). Significant computational savings can be achieved by restricting the search space for a particular landmark to as small a region as possible. By properly orienting these windows based on the relationship of the asymmetric uncertainty map and the landmark information available in LTM, these savings can be realized.

For the remainder of this section we first describe the growth procedure used by the spatial uncertainty map manager. This is followed by an analysis of the types of landmarks used and their application in the reduction of uncertainty by the spatial uncertainty map manager. These two techniques complete the feedback cycle used for the establishment of expectations for landmark position, and after identification, reduction in the size of the uncertainty region that was used to produce those expectations.

§3.2 *Uncertainty map growth statistics*

The spatial uncertainty map's growth arises from the difference between the commanded motion of the robot and the distance it actually traversed. The extent of growth (i.e. the amount of slippage and drift) is dependent to a high degree on the terrain that the robot is traveling on. Experimental data have been collected for each of the terrain types in the robot's current world (tile, concrete, grass, and gravel) and are presented in Table 3. It should be recognized that these data will vary based on daily conditions (e.g. long wet grass will have different values than newly mown dry grass, similarly a waxed floor will have values different from a dirty floor). If a more extensive table is built based on these varying conditions, a more representative error model of the world

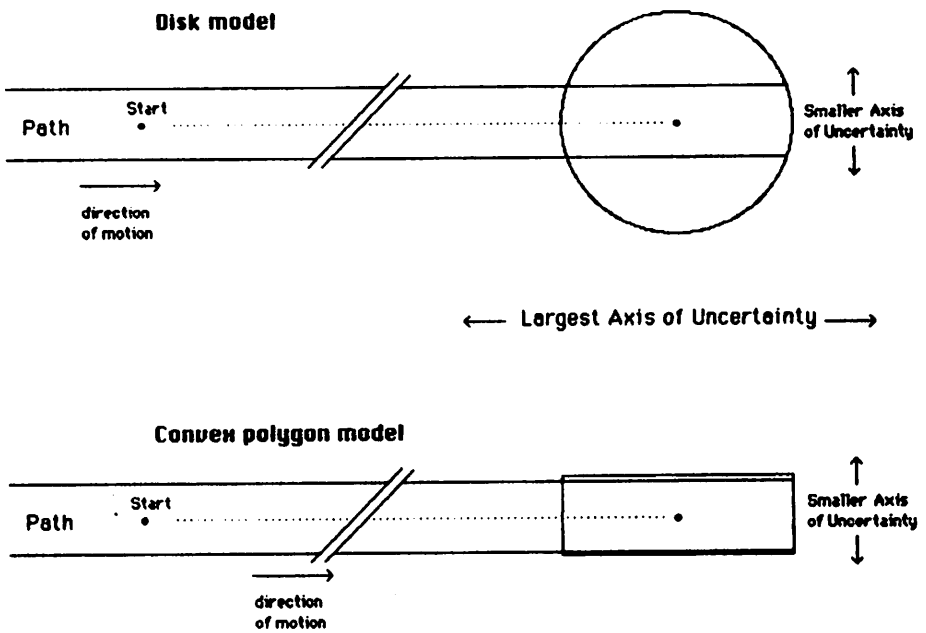


Figure 68: Disk model versus convex polygon model

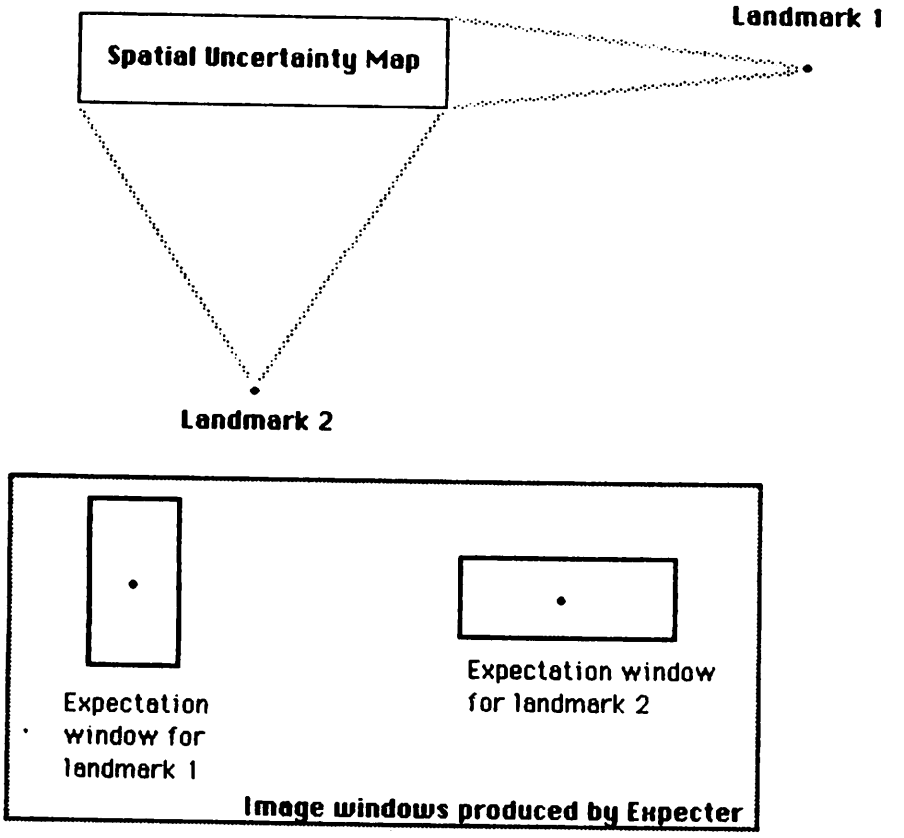


Figure 69: Image windows from convex model

For a polygonal map, smaller windows can be obtained for expected landmark location than for a circular model. Assuming the angular uncertainty is the same in all cases, Landmark 1 produces a window with a greater vertical extent, while Landmark 2's is more horizontal.

The window shape for a circular disk would generally be square instead of rectangular, and although the window itself would be simpler to compute, the overall computational cost would be greater due to the larger area required to be processed by the perception and matching algorithms.

would be available. For the sake of this dissertation however, we will assume that the statistics presented in this table are valid for all runs and all conditions for the particular terrain type concerned.

The data collected includes the following:

- Translational error (for straight-ahead motion)
  - mean translational error (loss)
  - mean inertial translational error (independent of distance traveled)
  - standard deviation of translational error
  - drift (degrees/foot)
- Rotational error (for turn commands about the robot's axis)
  - mean rotational error
  - standard deviation of rotational error
  - skitter (feet/degree)
  - standard deviation of skitter

The contribution to the uncertainty of the values used for the mean translational or rotational error depends on the distance the robot travels or turns. In all cases, an assumption is made that the robot will never go further than it is commanded. This implies that the robot does not slide significantly when it is told to stop (an invalid assumption for high-speed braking). For example, if the mean translational loss (error typically due to wheel slippage) is 0.10 (10%) and the robot is commanded to move 20.0 feet, the mean distance actually traveled would be 18.0 feet (similarly for degree errors in turning commands). The standard deviation serves to limit the likelihood of the robot being found within a given region to a specific probability (assuming a Gaussian distribution).

Inertial losses, which arise from slippage during the robot's acceleration and braking, are independent of the distance traveled. This quantity is a small fixed value that is independent of whether the robot has traveled 1 foot or 100 feet.

Table 3: Uncertainty data

	Tile	Concrete	Gravel	Dry grass
Translational Errors				
Mean Translational error (ft/ft)	0.042	0.044	0.038	0.026
Mean Inertial error (ft)	0.0084	0.0084	0.0084	0.0084
Total Standard deviation (ft/ft)	0.0028	0.0019	0.0017	0.0042
Mean Drift error (degrees/ft)	0.27	0.30	0.29	0.28
St. deviation Drift error (deg/ft)	0.12	0.14	0.14	0.14
Rotational Errors				
Mean Total Rotational error (deg/deg)	0.018	0.017	0.016	0.016
Total Rot. Standard deviation (deg/deg)	0.006	0.003	0.003	0.003
Skitter error (ft/deg)	0.0013	0.0014	0.0014	0.0012
St. deviation Skitter error (ft/deg)	0.00013	0.00029	0.00014	0.00014

The terrain statistics gathered are certainly dependent on the speed and acceleration of the robot over a given terrain type. For the relatively slow velocities used with HARV, the observed differences were negligible.

The problem of drift arises both from inaccuracies in the drive train of the vehicle and the terrain itself. As the vehicle moves, it "pulls" to the left or right depending on the particular configuration of the wheels. Drift is dependent on distance traveled. Although the drift value can be made a function of the current orientation of the wheels and handled by a simple look-up table, by overestimation we can treat this quantity as a fixed factor.

Skitter, the last error factor and the rotational analog of drift, is the robot's tendency to translate across the floor when given a command to turn. This quantity is dependent on the amount of rotation commanded. It is also orientation specific; the robot skitters (translates) to the right during a clockwise rotation (when viewed from the rear) and to the left during counterclockwise rotation. Although this asymmetry can be preserved in the growth procedures described below, the skitter component is quite small when compared to the translational error and is treated as a symmetric error.

A final comment on the validity of the statistics is in order. Although many man-days were spent on the collection and analysis of the statistical data presented in Table 3, it should be recognized that the accuracy of these figures is limited. A major effort would be required to fully model terrain characteristics. The time limitations for the development of this dissertation precluded this. Any errors in these figures have no effect whatsoever on the theoretical development of the UMS and the underlying growth routines for the uncertainty map. They have also served adequately for the experiment described in Chapter 8. If this system of uncertainty management is used for applications outside of the UMASS environment, a comprehensive study of the terrain characteristics of the domain in question is recommended.

§3.3 *Uncertainty growth procedure*

The uncertainty growth procedure resides within the uncertainty map manager. This routine draws on the statistical data described above and the robot's commanded motion in applying an uncertainty transform to the current uncertainty map. Each transform consists of a "turn and run" movement. If no rotation is commanded, the turn component

is zero. If no move command is issued, the run component is zero. In general, most robot commands will take the form of first turning to a new direction and then proceeding a given distance.

Let us assume that the uncertainty map is initially a point. Figure 70 shows an application of an uncertainty transform to a point. This point is transformed by the average translational and rotational errors (plus the inertial components) to yield a new center of uncertainty. The positional extent of the uncertainty map grows depending on the standard deviation of translational error. The assumption is made that overshoot is negligible. This sector-shaped region is then enclosed in a polygon, with any skitter or drift component added. The new uncertainty map's positional component is represented in Figure 70.

The orientation component is handled in a similar manner (Fig. 71). The center of orientation uncertainty is updated based on the amount of turn commanded and the rotational losses. The standard deviation is used to asymmetrically update the clockwise or counterclockwise uncertainty limits. Any orientation drift due to the translation is then added to these limiting values.

We now have a polygonal approximation of the positional uncertainty and a compass wedge representing orientation uncertainty. The technique for a new application of the uncertainty transform based on the next "turn and run" movement is straightforward. For positional uncertainty, the geometric transform as described above for a point, is applied to each of the points of the newly formed uncertainty map polygon. The center of uncertainty is updated exactly as before. A convex hull algorithm [44] is applied to the resultant set of points and a new positional uncertainty map results. The same approach is applied to the orientation compass wedge, but since the wedge is only one-dimensional, only the maximum value for each uncertainty limit need be retained. This process is illustrated in Figure 72.

The initial approximation of the robot's position need not be a point, but can be a bounding polygon. This is desirable as it is difficult to be certain of the exact location of the starting point of the robot in the global map unless we use a surveyor's transit.

In summary, the growth procedure operates as follows. For each "turn and run" movement, an uncertainty transform is applied to the existing position and orientation components of the spatial uncertainty map (the center point and all vertex points of the

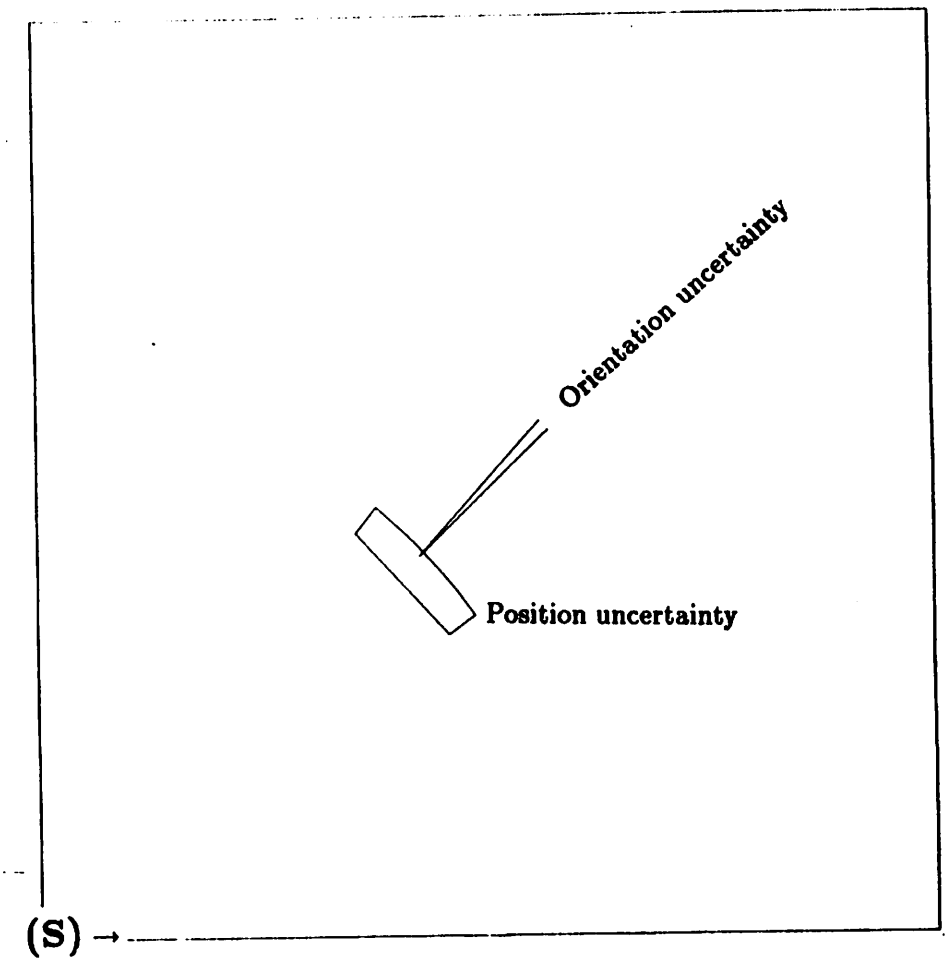


Figure 70: Point uncertainty transform

The robot starts at point S at the origin, facing directly to the right. A 45 degree turn is executed followed by a 30 foot move. This causes the spatial uncertainty map to grow and introduces orientation uncertainty as well.

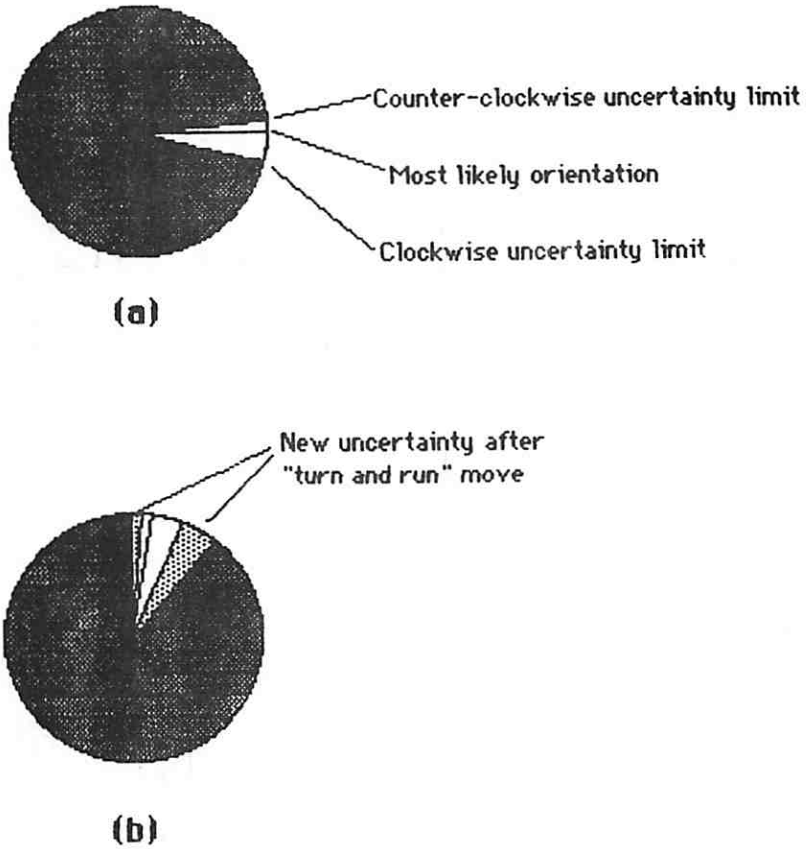


Figure 71: Angular uncertainty

The circle represents a compass with the unfilled areas representing possible headings of the robot relative to the global map.
a) Original uncertainty. The center line is the most probable orientation while the two side lines limit the uncertainty to a known probability. The center line reflects the accumulated mean errors in rotation while the side lines are produced from the standard deviations of the rotational error.
b) New uncertainty after a "turn and run" command. The direction of the turn was 90 degrees in the counter-clockwise direction, in this case producing a large increase in clockwise uncertainty. The small increase in counter-clockwise uncertainty is due to the drift that occurs during the run

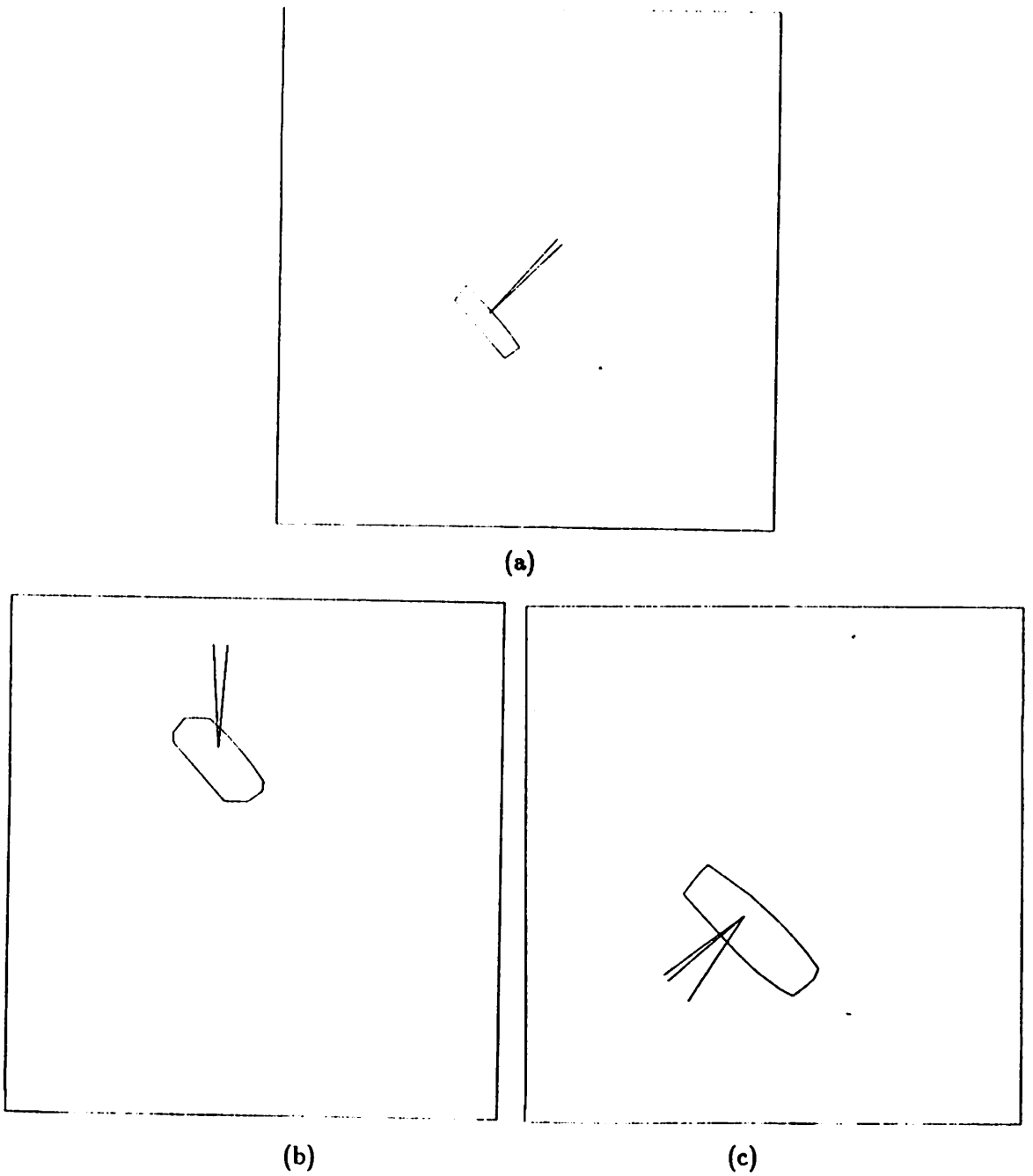


Figure 72: Map uncertainty transform
a) Starting spatial uncertainty map (from Fig. 70).
b) 45 degree turn and run applied to spatial uncertainty map.
c) 180 degree turn and run applied to spatial uncertainty map.

map itself). This transform uses as inputs: the current terrain characteristics represented in a statistical format; the vehicle commands sent to the motor controllers of the robot; and the current spatial uncertainty map. It produces as output a new uncertainty map that has been increased in uncertainty in a manner reflecting the input data. This in turn can serve as input for the next "turn and run" movement.

Without environmental feedback, this map would grow to eventually fill the global map. The following subsection describes the methods that can be used to reduce both components of the spatial uncertainty map based on successful landmark recognition.

§3.4 *Uncertainty reduction procedures*

The uncertainty represented in the spatial uncertainty map is reduced by means of landmark recognition. As AuRA is predominantly a vision-based system, the types of landmarks used are those capable of being extracted from video images. Other uncertainty reduction techniques can be applied to landmarks recognizable by other sensors, but they will not be considered here.

Three distinctive landmark classes are available to the spatial uncertainty map manager for the reduction of uncertainty. The first class consists of landmarks, typically walls or poles, that produce vertical image lines that have been truncated by the top of the image. The depth to these lines is not directly ascertainable from the image data. Although their location in world coordinates is known, the distance from the robot to the landmark cannot be computed from this information alone. There is no way to tell if the fraction of the line detected in the image is a major or minor portion of the landmark's total physical extent. What can be obtained is a ray from the landmark's world position to the robot. The robot must be located somewhere along this ray. This ray thus can be used to constrain the spatial uncertainty map. The second class of landmarks consists of identifiable image features that can be used, when combined with *a priori* knowledge of the physical landmark's height and location available from LTM (i.e. their world coordinates), to tightly constrain the uncertainty map. The third class consists of long lines typically found in the ground plane such as road edges. These lines help control the uncertainty in the direction of motion. These three classes are illustrated in Figures 73-75 and are discussed in more detail below.

Reiterating, Class I landmarks are typically based on the recognition of a vertical

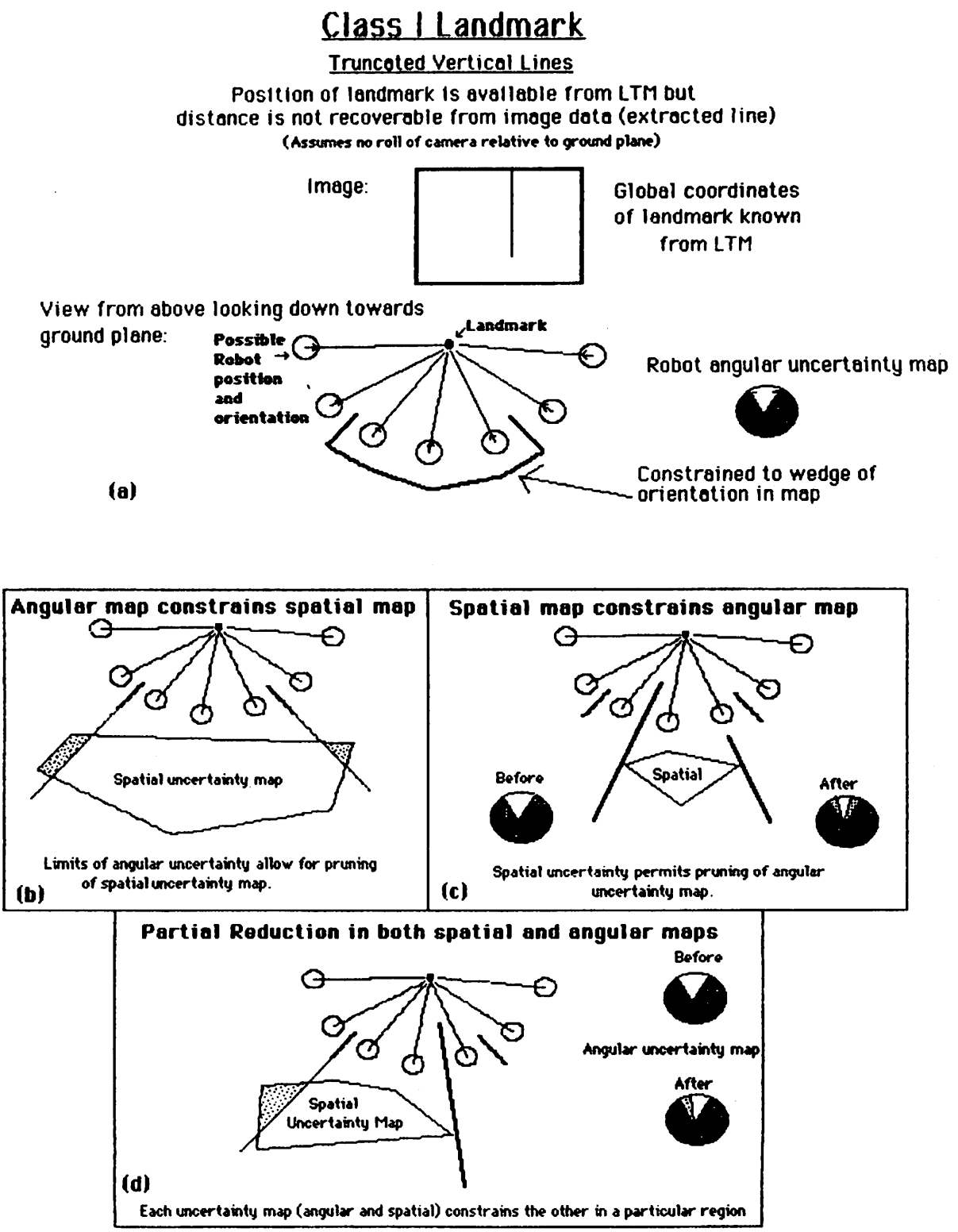


Figure 73: Class I landmarks - Vertical lines

Actual landmark yields a truncated vertical image line.

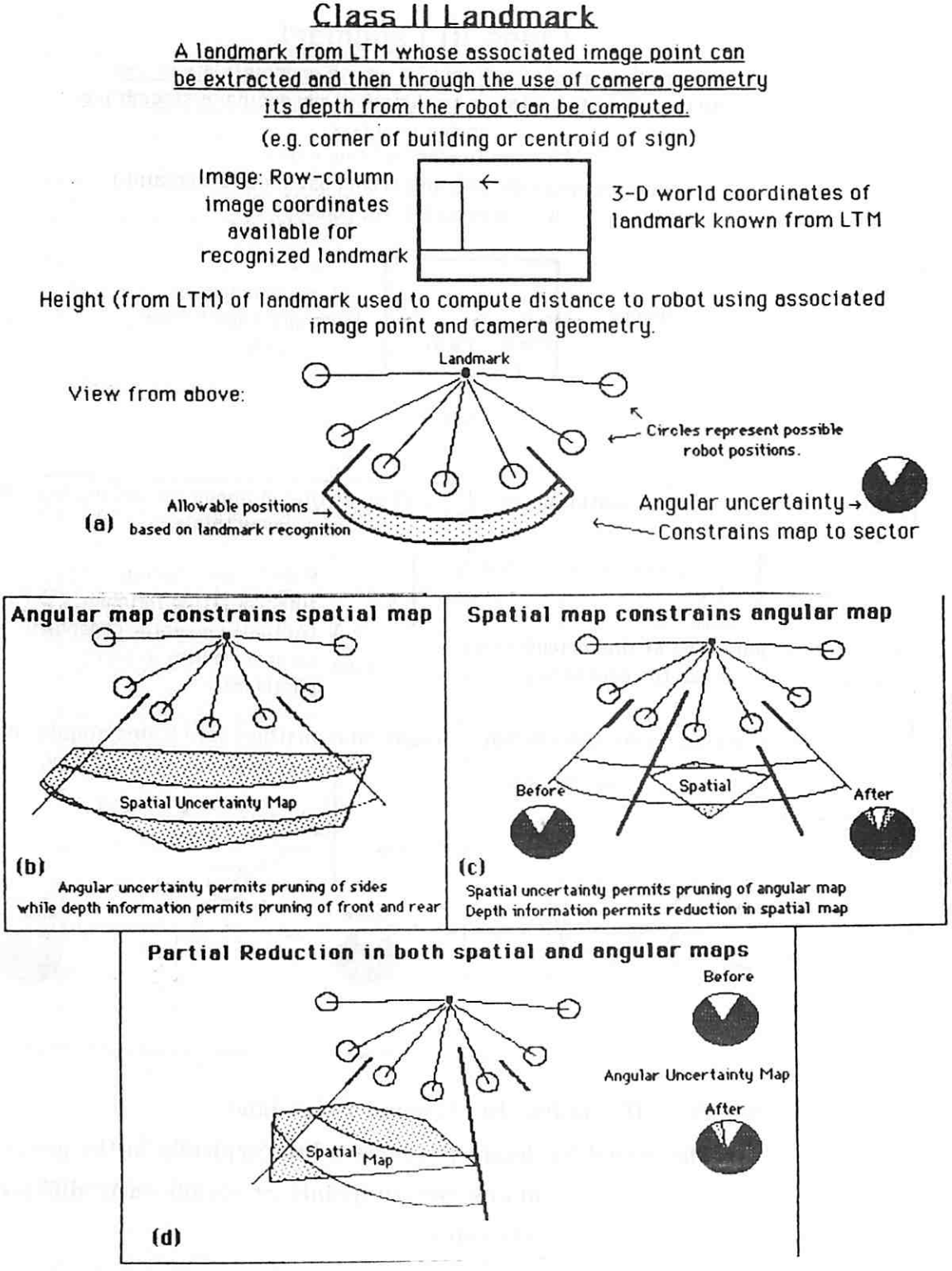


Figure 74: Class II landmarks - Points

Match made between 3D actual landmark world coordinates and 2D image plane coordinates of landmark's image.

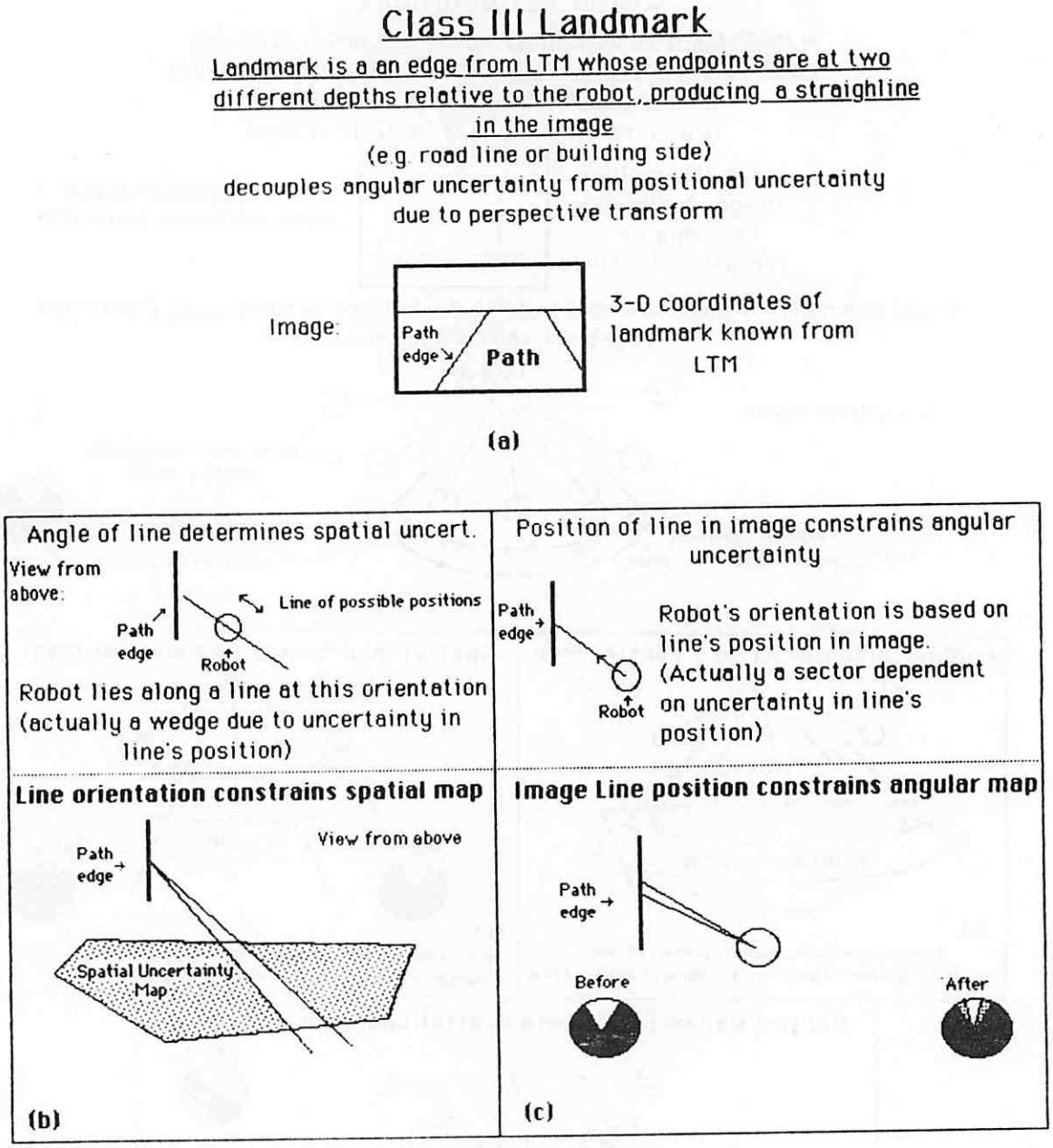


Figure 75: Class III landmarks - Ground Plane Line

The actual landmark produces a line, typically in the ground plane, that contains two endpoints at significantly different distances from the robot.

image line that has been cut off at the top of the image. This type of landmark appears, for example, with a closeby doorway in an interior scene or a nearby building outdoors. Even with *a priori* knowledge of the height of the associated feature (the doorway or building), it is impossible from this evidence alone to extract the distance to the landmark feature from the image. Only a ray emanating from the associated world map point can be used as information for spatial uncertainty map pruning purposes. This is a consequence of the fact that we do not know how much of the image line represents the total actual side of the door or building; if we are close, it may be a small fraction; if far, it may be almost the entire side. Although depth information is lacking, by combining the information available in the spatial uncertainty map regarding possible positions and orientations of the robot, uncertainty can be significantly reduced. The evidence obtained from a class I landmark consists of a ray emanating towards the camera. The position of the ray in the image plane (Fig. 73) restricts the possible orientations of the robot relative to it. The positional component of the uncertainty map when combined with the known global position of the recognized landmark restricts the possible locations of the vehicle. In some cases the extent of orientation uncertainty enables a reduction in the positional component of the uncertainty map (Fig. 73b). In others the spatial map allows reduction of the orientation uncertainty (Fig. 73c). Finally, in some instances, a reduction in both position and orientation uncertainty can be made (Fig. 73d).

Class II landmarks are landmarks whose image produces a recognizable point that can be directly matched to a landmark feature in LTM, yielding information regarding its world coordinates, height, etc. Class II landmarks (Fig. 74) provide significantly more information than do class I. Typical class II landmarks include building corners, road signs, or any recognizable feature that can be represented with its 3D coordinates in the meadow map. Both the 3D world coordinates and the matched 2D image plane coordinates are used in position estimation. Through the use of camera geometry and the perspective transform, the distance from the robot to the recognized landmark can be established within some known error limit (assuming the vehicle's relationship to the ground plane is understood). The distance error is based on camera calibration error, digitization resolution, actual landmark location uncertainty, etc. The detected landmark's relationship to the spatial uncertainty map (and hence robot) is best described as a fuzzy point location, as the actual location of the landmark is known only within

limits and not exactly. It is not a sharp point due to the inherent uncertainty in the imaging process and in the actual location of the landmark. The net effect is that with this approximate depth information, we can accomplish everything that a class I landmark offered, but also reduce the forward and rearward components of spatial uncertainty (Fig. 74b-d).

Class III landmarks (Fig. 75) are similar to class I landmarks in that they do not directly provide depth information. This class of landmarks arises from landmarks in LTM that are typically located in the ground-plane in a direction that is oriented away from the camera (i.e. not parallel to the image plane) and that produce lines in the 2D camera image. The best example is a path edge. With this information it is possible, as a consequence of the perspective projection, to decouple orientation and positional errors. The angle of the line in the image plane corresponds to the relative translational position of the robot to the line (in a ray-like manner). The position of the line in the image (left or right) provides feedback on the orientation of the robot. This same type of information is used to servo the robot for path-following based on line finding as described in Chapters 6 and 8. The reductions in uncertainty that are possible with class III landmarks are illustrated in Figures 75b-c.

An important feature in the use of these classes of landmarks is that triangulation (the recognition of two landmarks in widely separated locations) is not required to improve the vehicle's estimate of its position and heading. Triangulation certainly can be subsumed by this method (e.g. identification of two class I or II landmarks). It is a goal of this system however to provide concurrent landmark recognition without forcing the camera to search through the countryside, using a pan-tilt-zoom mechanism. In this manner only relevant landmarks are sought in the direction of the robot's motion. If the robot becomes sufficiently disoriented, as recognized by exceeding a certain area threshold for the spatial uncertainty map or by failing to detect several predicted landmarks, it can then stop and look around for familiar and easily discernible landmark features.

The actual coded implementation of uncertainty reduction involves slicing the uncertainty map as specified in Figures 73-75. In the case of class II landmarks, where sectors are produced (e.g. Fig. 74b), the curves produced are enclosed with lines, maintaining the convex polygon representation for spatial uncertainty as described in Section 3.3.

How these landmarks are specified, selected, sought after, and recognized within the

UMS is the topic of the next two sections.

§4. **Find-landmark** schema selection

Uncertainty reduction cannot be accomplished without the availability of recognizable visual landmarks. These landmarks must be stored in memory and selected for possible recognition when necessary. Perceptual recognition strategies must then be associated with each chosen landmark and activated when appropriate to complete successful landmark recognition. This section describes the roles of long-term memory (LTM), short-term memory (STM), the pilot, and the motor schema manager in the context of uncertainty management. All of these AuRA components also serve other useful functions: long-term and short term memory for planning purposes, the pilot and the motor schema manager for motor behavior selection and activation. These systems will only be discussed here in the context of uncertainty management.

Long-term memory

Landmark information must be stored somewhere. As AuRA generally assumes the existence of a partially modeled world, it is a logical consequence to embed landmark data in LTM. Two choices are possible:

1. The landmarks could be created automatically based on available 3D world model data and a visibility analysis.

2. Specific landmarks can be chosen by the designer of the system and consist only of those landmarks that are expected to be particularly useful (i.e. a tuned subset of the collection in 1).

Several advantages of the second method are apparent. First, the landmarks are precompiled for a particular region and so it is merely a memory access operation to obtain them, which does not burden the system with additional computation. It is possible that this precompiling could be automated from available 3D world models. Second, and perhaps of more significance, it is also easier to ensure proper selection of landmarks and their activation ranges for the experimental testing of the system. In addition, the number of **find-landmark** schemas can be kept down to a reasonable

number. AuRA stores a 3D model of the world (that is a partial wire frame world view), so it is feasible to implement a visibility search to determine which landmarks could be of value. In the interest of a reasonable completion date for this thesis however, a specific subset of landmarks was manually selected and tied into each meadow.

Each meadow contains a pointer to a landmark list. This list contains pointers to useful landmarks visible from that meadow, not only those within that meadow. For some meadows this list will be empty. For others there will be one or more landmarks available. The information stored will depend on the specific landmark but typically includes some of the following:

- Symbolic class (e.g. lamppost)
- Symbolic label (e.g. Lamppost_107)
- Landmark Class (I,II,or III)
- Activation criteria
  - Minimum and maximum effective recognition distance of the robot to the landmark
  - Acceptable viewpoints - maximum and minimum heading
- Means of identification - perceptual strategies to use
- Spectral data (if applicable)
- 3D structure (if applicable)
- Positional uncertainty of landmark in global map
- Pointers to subparts
- Pointer to parent object

A single landmark may be present in the landmark list of several meadows. The landmark need not be located in the meadow, but only visible and potentially identifiable from some point within that meadow.

Short-term Memory

When the navigator specifies a particular leg for the pilot to execute, the cartographer recognizes this and instantiates a group of related meadows from long-term memory into STM. These instantiated meadows consist of the meadows that the robot is expected to traverse during this particular leg of its journey. Adjacent and other nearby meadows (where nearness is measured by the proximity to the computed navigational path) are also instantiated. This information, which is chosen by the cartographer, restricts the number of landmarks for the pilot to search in its quest for suitable **find-landmark** schemas.

Pilot

Prior to initiating motion, the pilot accesses the landmark lists from the instantiated meadows in STM. Using the available schema library, the pilot parameterizes the **find-landmark** schemas with information available from the landmark data. It also sets activation criteria based on each landmark's location and identifiability. Appropriate perceptual schemas are parameterized with values suitable for the landmark in question. For example, if the line finder is to be used, filters and/or buckets will be tuned to specific line orientations. If the region segmentation algorithm is to serve as the basis for identification, the spectral and region characteristics will be set by the pilot. The pilot then passes this set of slot-filled schemas to the motor schema manager for instantiation.

Motor schema manager

The **find-landmark** schema instantiations (SIs) are created by the motor schema manager immediately upon their receipt. In general, many are immediately placed in a state of hibernation until the robot moves into range for potential identification. At that time they are activated and make specific calls to the Expecter process to determine their anticipated position in the image. This portion of the image is then processed by the **find-landmark** SI. If the landmark is deemed identified (see next section) that fact is placed in the identified landmark buffer with a time-stamp for the uncertainty map manager reduction processes to use. The **find-landmark** schema continues to make periodic reports, tracking the landmark over multiple frames.

The role of the Expecter process and the means for landmark identification are described in the following section.

§5. Landmark identification

Actual landmark identification consists of matching the information acquired by the perceptual schemas, using the expectations established by the Expecter process, and the landmark model itself. This section first describes the operation of the Expecter process and then the identification mechanisms available within UMS for landmark recognition.

§5.1 *Expecter*

The Expecter process restricts the search for a landmark to a particular region in the incoming image. This reduces the possibility of erroneous identifications and affords better utilization of available computing power. The initial implementation uses a single Expecter process outside the realm of the motor schema manager. It is also possible to create individual expectation schemas for each and every **find-landmark** schema running within the motor schema manager. This strategy was not employed in AuRA's first-pass implementation as the time required to produce the expectations for a perceptual schema is small compared to that for the processing of the perceptual schema itself.

By correlating the spatial uncertainty map against the known position of a landmark in LTM, a window in the image can be established which, to a known probability, contains the landmark's image projection. This is accomplished by analyzing the spatial extent and angular uncertainty of the uncertainty map in light of the landmark's global position (see Fig. 76). Worst case analysis establishes the window. The top of the window is determined by computing where the landmark would appear in the image if the robot was located at the closest point on the spatial uncertainty map, while the farthest point is used to determine the bottom of the expectation window. The leftmost and rightmost boundaries of the image window are determined by applying the clockwise and counter-clockwise uncertainties in heading respectively to the individual spatial uncertainty map vertices and determining where the landmark would appear in each case. The predicted image locations furthest to the left and right complete the rectangular window bounds. The resultant window must be adjusted so as to include an adequate area to produce the intermediate representations of image features necessary for the landmark identification processes described below. For example, if the corner of a building is to be located using the line finder, an adequate window size must be provided. This would involve a

window much larger than the corner so that accurate lines could be extracted to determine the intersection that yields the corner sought for (Fig. 77a-b). On the other hand, if the Moravec operator is to be used for the same task, a much smaller window can be established for this perceptual schema (Fig. 77c-d). Figure 78 shows typical windows produced for different landmark classes.

In order for the Expecter process to reliably predict the landmark location, it must have information obtained from camera calibration procedures. The computed perspective transform is then applied to the position of the landmark relative to the possible locations of the camera in the world as determined by the uncertainty map. This yields the image window to be searched. The calibration process is described in Chapter 6.

§5.2 *Landmark discovery and tracking*

It is difficult to determine just when a landmark has been positively identified in scenes as unconstrained as those to be found in AuRA. A judicious choice of landmarks that produce relatively unambiguous data under normal circumstances is a key factor for successful recognition. Two distinct phases for landmark recognition are present: discovery and tracking. These two phases bear a strong relationship to the bootstrap and feedforward methods described in Chapter 6 in the context of perceptual analysis.

The discovery phase uses the model provided by LTM to locate a landmark within the image window. This phase is vulnerable to failure due to obscuration, poor or changing lighting conditions, etc. It is advisable that discovery of a landmark be confirmed over several images to ensure that a transitory event did not produce a mismatch in a single frame. The schema activation level mechanism readily accommodates this. Nevertheless, discovery is the most difficult component of the landmark identification process.

The tracking phase adjusts the LTM model of the landmark used for discovery to provide a newer model (within the image) that is used to guide landmark identification in images acquired after the initial phase. The model is continually adjusted as successive images are acquired. The newly produced model is compared with the original LTM version to ensure that the landmark was not incorrectly identified in the first place. Tracking generally assumes reliable discovery. If a landmark is shown to be incorrect during the tracking phase, the robot must assume that its uncertainty map is partially invalid and institute special methods to regain its bearings. The concepts involved in

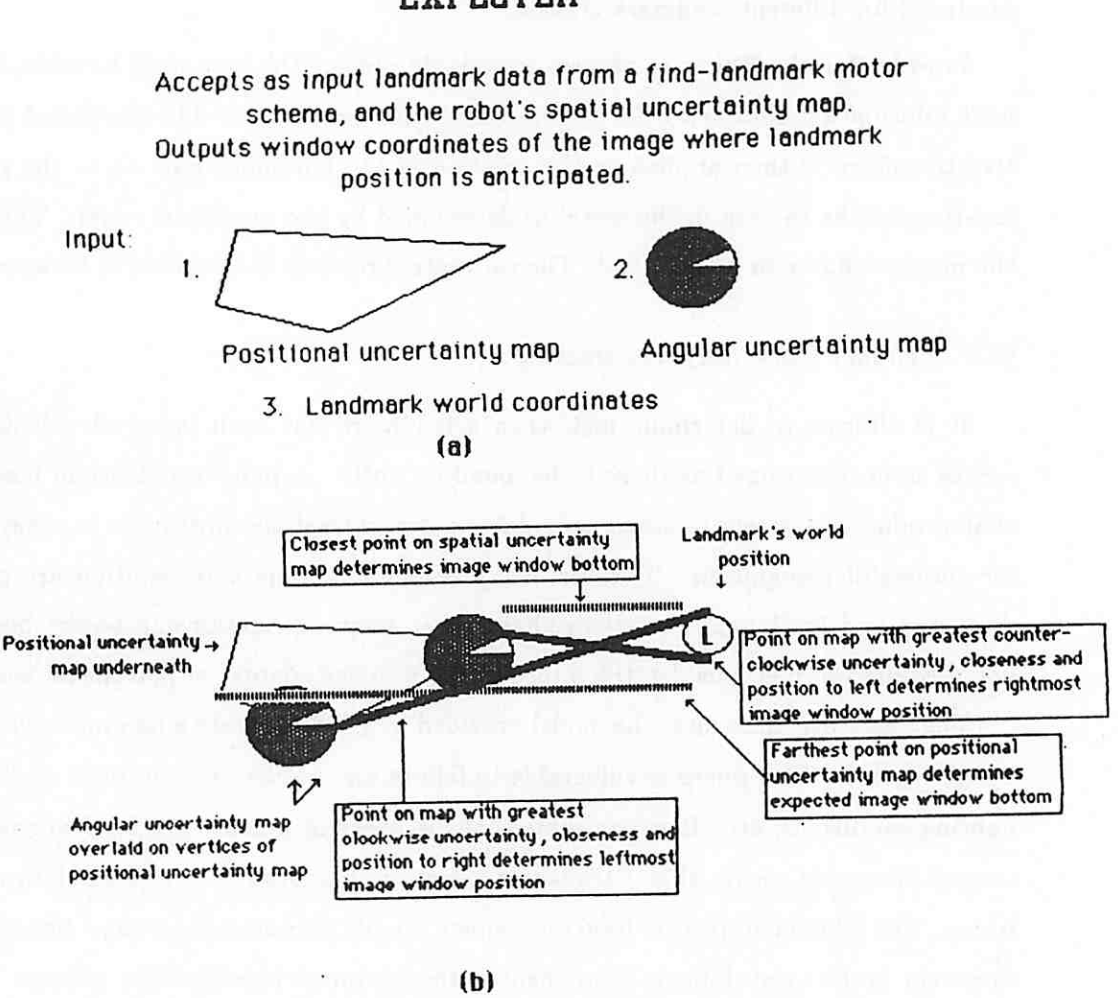


Figure 76: Expecter

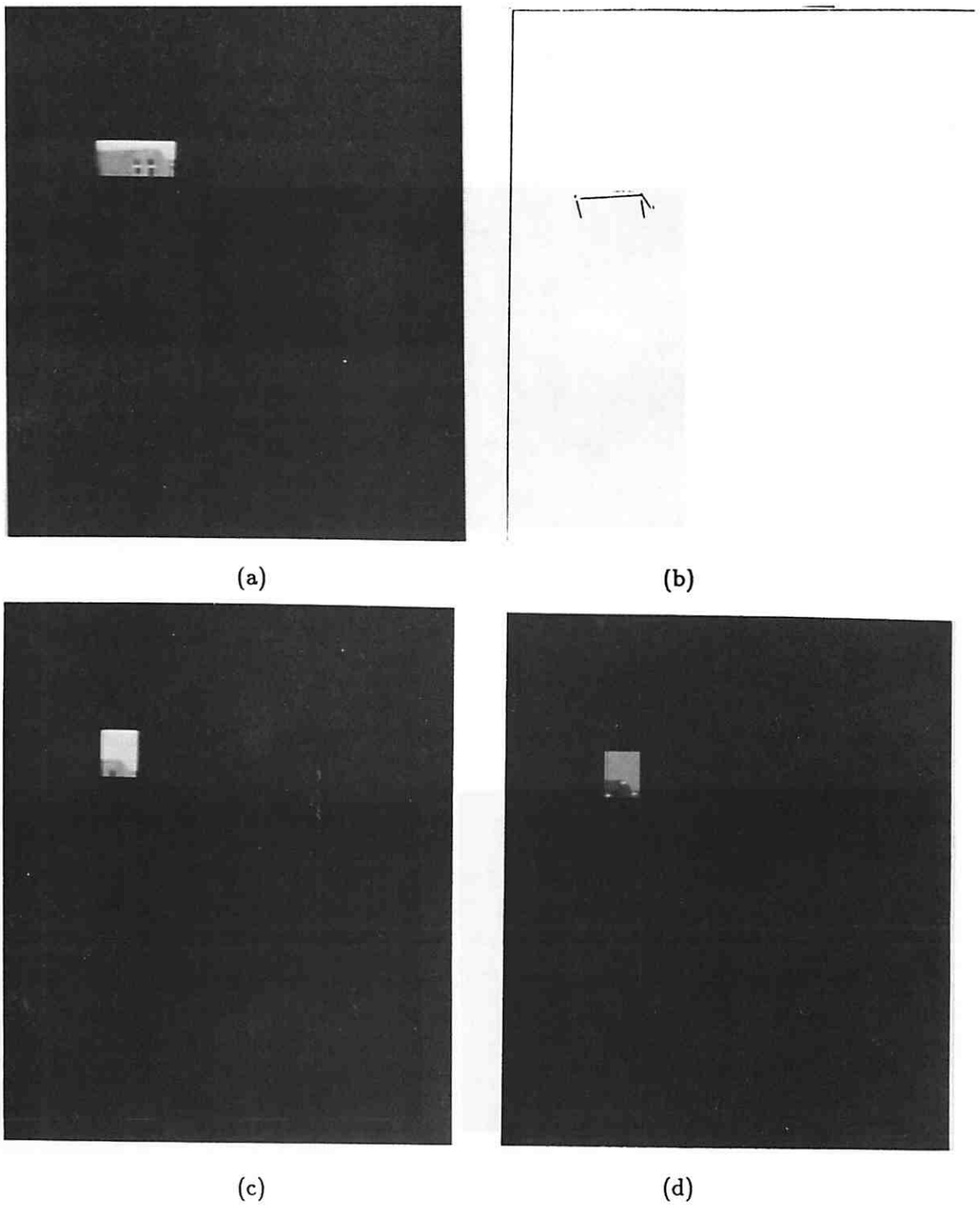
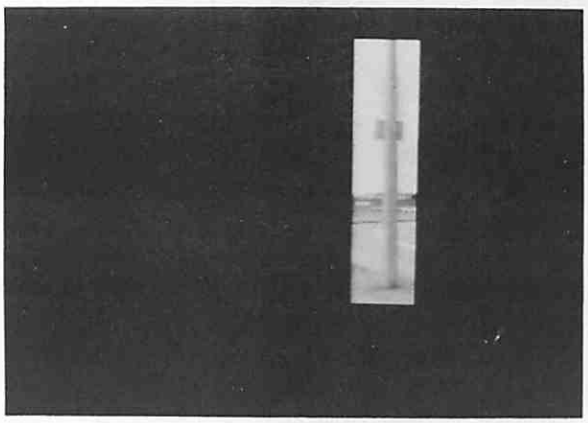
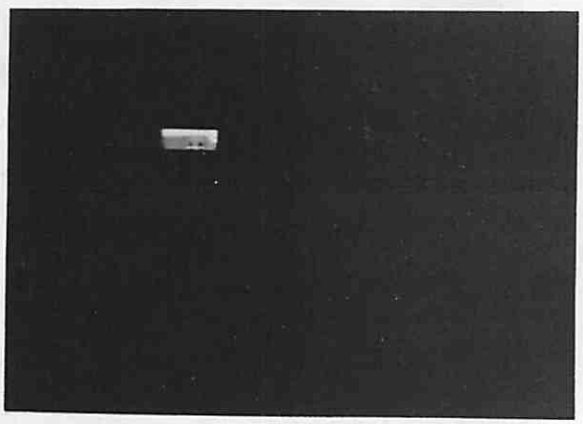


Figure 77: Expecter windows

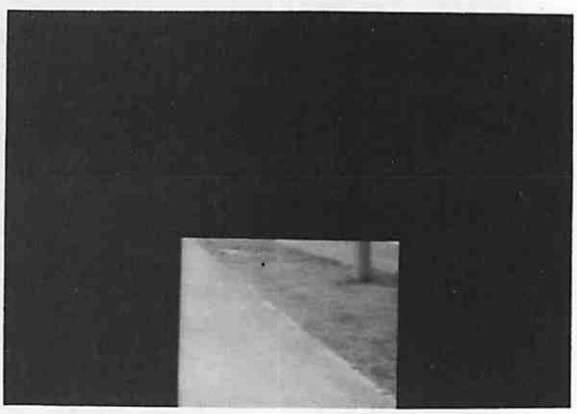
a) Window size (search area) for building corner using fast line finder.
b) Results of fast line finder within window a).
c) Window size (search area) for building corner using Moravec operator.
d) Results using Moravec operator within window c). The interest point furthest to the top and right identifies the corner.



(a)



(b)



(c)

Figure 78: Typical landmark class windows
a) Class I - Lamppost
b) Class II - Corner of building
c) Class III - Path edge.

tracking (feedforward) are explained in Chapter 6. The remainder of this section will deal with the more difficult problem of discovery.

A separate thesis could be written about the problem of discovery of landmarks in natural scenes. The UMASS schema group [42] is addressing the problem of object recognition which closely overlaps landmark discovery. Other projects, described in Section 1, report possible mechanisms to handle this difficulty [4,17]. Burns and Kitchen [32] are developing means for recognizing 3D objects in 2D images using prediction hierarchies for potential use within UMS. "Generic views" serve as a means for compiling 3D information into a useful format for 2D recognition. Techniques for indexing into a large database of objects and their object views are being developed. For the implementation version of AuRA's UMS, somewhat naive approaches will be used for the discovery process. These include line matching, region extraction, and corner identification.

Class I landmarks, typified by strong vertical lines, are identified through the use of the fast line finder described in Chapter 6. The fast line finder is run within the window produced by the Expecter. If a sufficiently strong line of proper orientation is encountered, where strength is measured by length and gradient magnitude, the landmark is deemed identified and the **find-landmark** schema converts to the tracking phase.

Class II landmarks, which yield the depth of a modeled landmark, are extracted using the line finder, the region segmenter, the interest operator, or some combination of these perceptual strategies. The multiplicity of algorithms available allows the opportunity to explore cooperating schemas as a confirmation mechanism. Landmark discovery is declared when specific certainty thresholds for a single algorithm are exceeded and/or mutual concurrent discovery occurs from different vision algorithms.

Class III landmarks, most commonly path edges, are identified by the fast line finder path-edge extraction method and/or by the region segmenter (see Chapter 6). Whenever a landmark is identified, either by discovery or tracking methods, its time-stamped location relative to the robot is stored in the identified landmark buffer. This is done independently of the class.

The techniques used for actual landmark discovery in the current version of AuRA do not use sophisticated 3D models for landmarks. Although this limits the current versatility of the system, in particular during conditions that produce partial or complete obscuration of landmarks, more sophisticated strategies for landmark discovery can be

easily embedded in AuRA when they become available. This is achieved by using the feature editor (Chapter 4) for the meadow map as the mechanism for the addition of new models and by taking advantage of the modularity afforded by the use of schema-based control (Chapter 5). Ultimately, as the VISIONS system approaches real-time scene interpretation with the advent of the UMASS IMage Understanding Architecture, more robust methods for landmark discovery will be available.

§6. Examples and implementation

We describe here examples showing the growth of the spatial uncertainty map as a consequence of motion over differing terrain types, and its resultant pruning due to landmark recognition. The examples in this section do not deal specifically with the landmark discovery process. Landmarks are entered in the simulations below through the identified landmark buffer, without concern for the specific perceptual processing that placed them there.

The implementation of UMS is in Common LISP and runs on a VAX-750. The supporting cartographic processes and representations are coded in C.

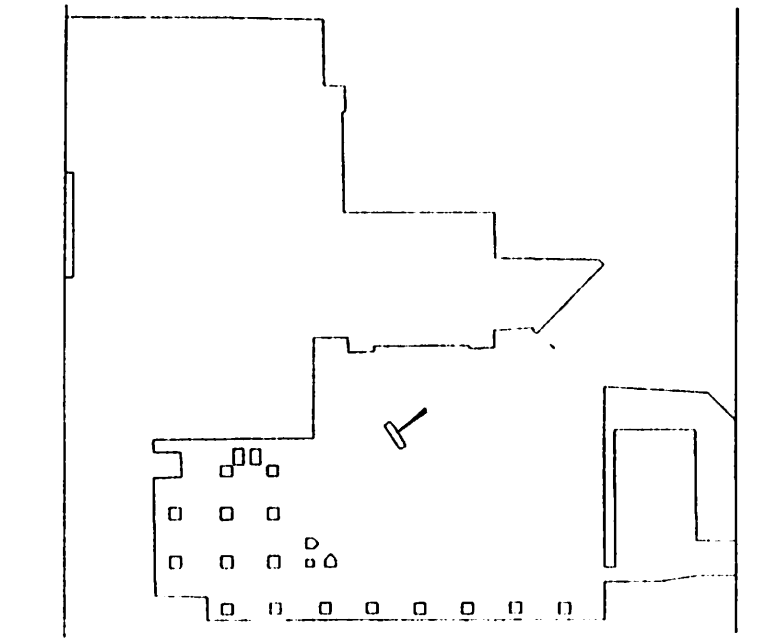
Figure 79 shows the growth of the spatial uncertainty map during repeated movement of the robot without landmark recognition feedback. Note that the unchecked growth rapidly covers a large portion of the global map.

Figure 80 shows the importance of landmark recognition in controlling the spatial and orientation uncertainty over the same path. Marked reductions in both positional and rotational uncertainty are apparent upon inspection of the figures.

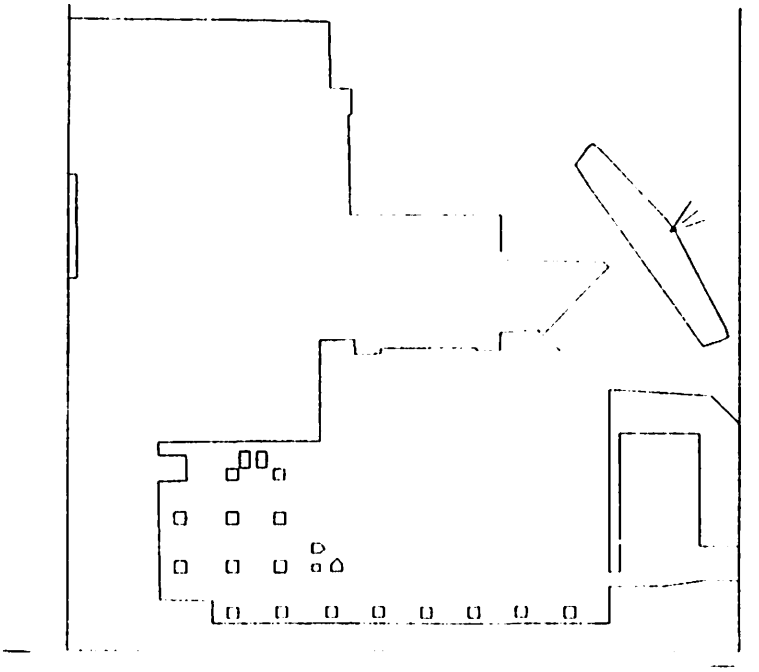
Chapter 8 presents a localization experiment involving the spatial uncertainty map using images acquired from our mobile robot HARV.

§7. Summary

An Uncertainty Management System has been developed for AuRA to provide for the efficient use of computational resources in the guidance of perceptual processing, and as a means to ensure successful navigation of a mobile robot within a partially modeled environment. To accomplish this, a spatial uncertainty map representing both



(a)



(b)

Figure 79: Uncertainty map growth without recognition feedback
a) Small amount of movement produces small uncertainty map.
b) Continued motion without landmark recognition produces large spatial uncertainty map.

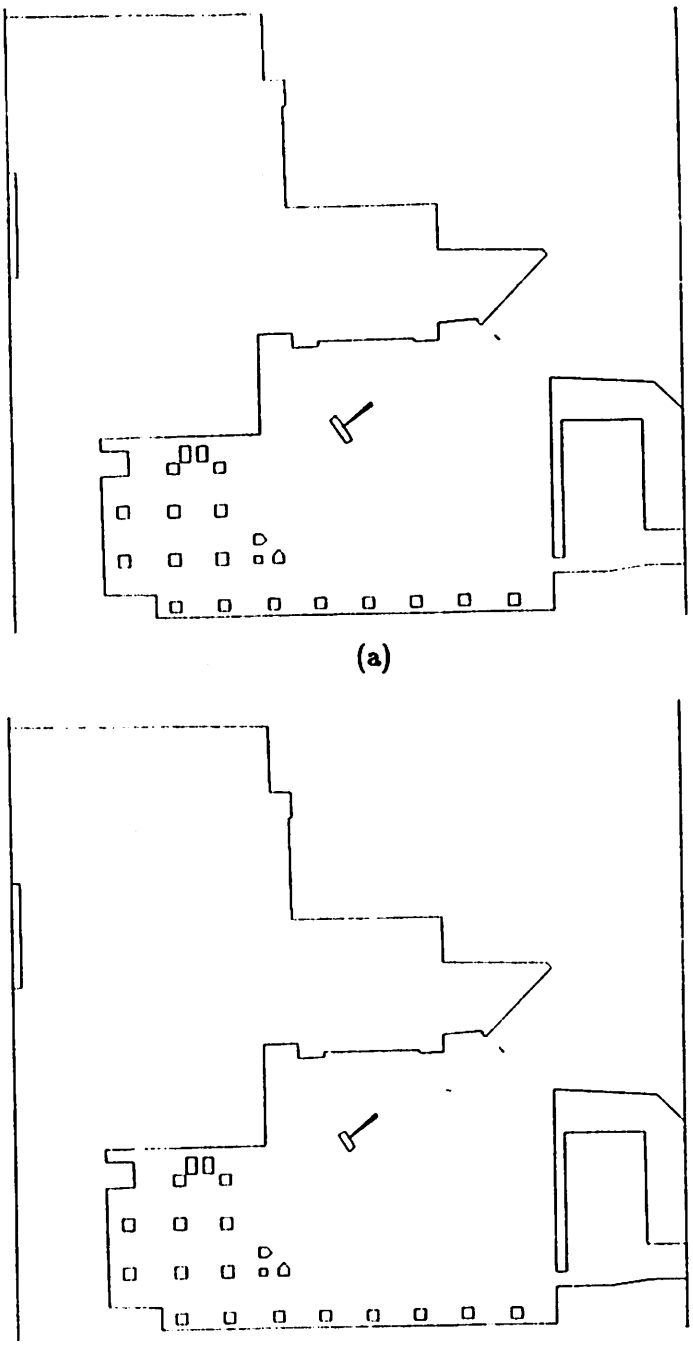
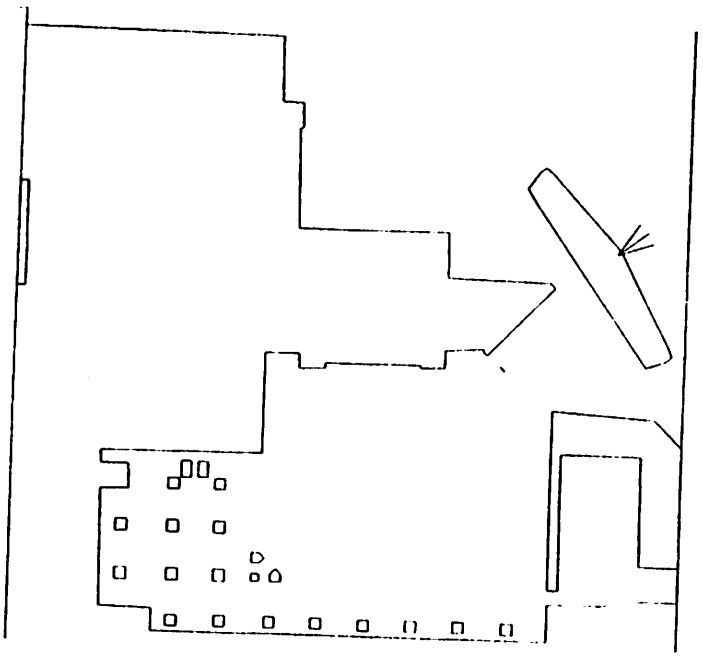
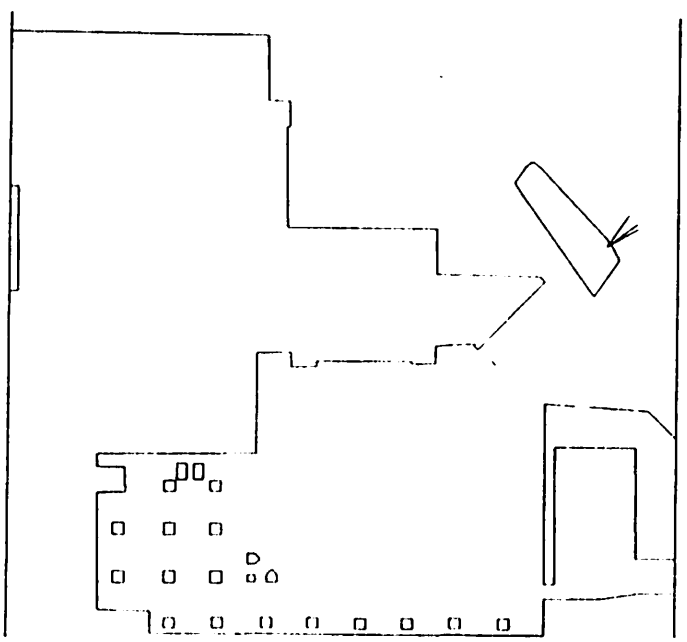


Figure 80: Uncertainty map with recognition feedback
a) Original small uncertainty map.
b) After pruning based on identification of building edge.
(Figure continued on following page).



(c)



(d)

Figure 80 continued.

c) After next movement but before next landmark recognition.

d) Reduction in both spatial and angular uncertainty due to recognition of lamppost. This case corresponds to Figure 73d, where uncertainty reduction is made to one side of the positional component (the rightside) and to one side of the orientation component (clockwise uncertainty).

positional and orientation uncertainty has been created. Specialized processes, used for both the growth and the reduction of this uncertainty map, have been implemented. Landmark selection to provide appropriate feedback for the control of uncertainty is handled by the pilot, based on the current navigational goals. An Expecter process is used to guide perceptual schemas in their interpretation of image events by restricting those interpretations to specific portions of the image.

AuRA's approach to uncertainty is based on the tenet of action-oriented perception. The advantage of this approach lies in the ability to restrict the computational needs of perceptual processes by limiting their operation to only portions of incoming images. This is important when many different processes are performing different perceptual tasks.

This particular approach for uncertainty management is somewhat narrow in focus and is not claimed to be generalizable. It is geared specifically for mobile robot architectures that contain significant amounts of reliable *a priori* knowledge. It would not work well in systems that acquire their world maps dynamically. The designer must be careful in the accuracy of his world map regarding landmark position. Significant errors in the *a priori* world map would force the robot to stop and initiate more costly means for localization (e.g. scene interpretation).

Most of the implementation of the UMS is complete, although it is not fully integrated with the navigational components of the AuRA architecture. Landmark discovery is one area that remains to be worked on. Hopefully when the UMASS Image Understanding Architecture is completed and running the VISIONS system, this problem will be easily manageable.

Another area for growth is the replacement of the shaft encoders with inertial navigation, eliminating the need for terrain modeling and restricting uncertainty growth to be based on the drift and other cumulative errors found with this more costly piece of hardware. Predictions from the Expecter would then be tighter, resulting in even lower processing demands.

Uncertainty management at the mission planner level has been largely ignored in this treatment as it addresses only spatial issues. Symbolic reasoning for dealing with unpredicted events will need to be studied when the mission planner takes on a fuller implementation.

Additional work on more sophisticated vision algorithms that recognize expected

landmarks from their 3D models is an important area of future research. By using available knowledge from the spatial uncertainty map and long-term memory, limitations on each landmark's pose and distance relative to the robot can be obtained. This scale and orientation information, when applied to the landmark's 3D model, can then be used to invoke the perceptual strategies that are most appropriate for the identification task.

CHAPTER VIII

EXPERIMENTS

§1. Introduction

This chapter describes the various experiments that were used to demonstrate the different components of AuRA. The test domains included the second floor of the Graduate Research Center (Fig. 12) and its immediate outdoor surroundings (Fig. 13).

Several different types of experiments were performed. Some involved the testing of individual components of the architecture in isolation. In particular, the relatively slow vision algorithms were exercised in this manner. Other components were tested in a system framework, e.g. the ultrasonic/encoder schema-based navigation. Management of the robot's uncertainty was also experimentally tested. The last experiment tests several of the different AuRA components working on the same task.

§2. Motor schema-based navigation

A real-time schema experimentation/demonstration system was developed based on sensing using ultrasonic and encoder data. The hooks for tying in visual sensing are in place but are currently not implemented (due to the slow processing speeds for vision).

Five different motor schemas have been implemented: **move-ahead** (encoder based), **move-to-goal** (encoder based), **avoid-static-obstacle** (ultrasonic based), **noise** (sensor independent), and **follow-the-leader** (ultrasonic based). The user is able to select the collection of motor schemas to use and associate a perceptual schema with each. After schema selection is complete, robot motion is initiated. The vehicle then behaves in an intelligent manner in response to its environmental stimuli. Several of the more interesting behaviors are described below. Although the available videotape of mobile robot behavior

shows more than words allow, the following subsections and still photograph sequences describe these behaviors as well as we can. It should be noted that the schemas for this system emulate distributed processing, but actually are evaluated sequentially.

§2.1 *Avoidance*

By instantiating the **avoid-static-obstacle** schema with ultrasonic perception, the robot manifests an interesting behavior. The schema instantiation can be controlled by altering both the gain, which affects the velocity of the vehicle, or the sphere of influence of detected obstacles, which increases its sensitivity to the environment. When activated, the robot seeks out a potential field minimum and remains there unmoving (or slightly oscillating in cluttered environments due to the limited sampling rates and noise in the sensor data).

If a change in the environment occurs, (e.g. a person approaches), the robot is repulsed and seeks out a new potential minimum. The robot can be "herded" by following behind it, forcing it to move to a desired location. It avoids obstacles during its journey and then settles into its new location when the environment stabilizes. Figure 81 illustrates this process.

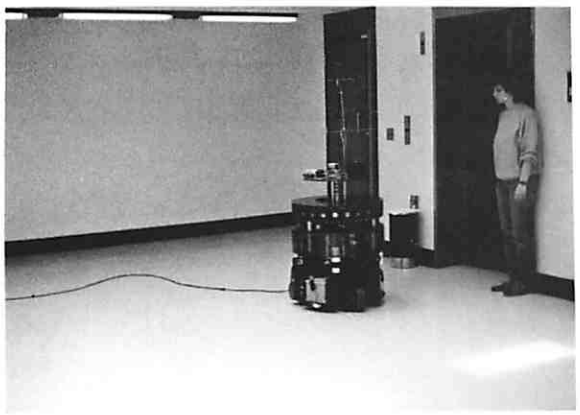
§2.2 *Exploration*

By combining the **noise** schema with the **avoid-static-obstacle** schema, exploration behavior can be observed. The **noise** schema's gain (strength) and persistence (how frequently the direction changes) can be set at startup by the experimenter. The robot meanders about the lab exploring different regions while avoiding collision with the obstacles. (Fig. 82).

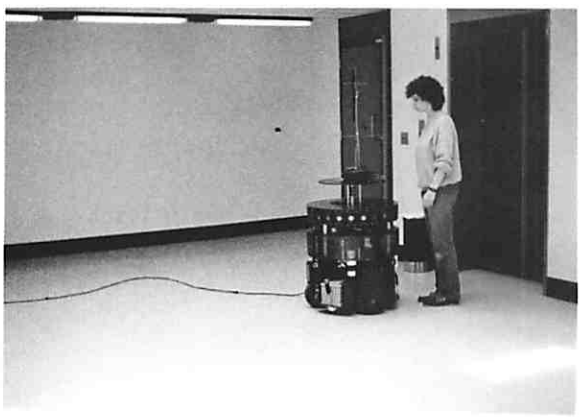
The robot responds quite well in response to changes in its environment, as when people surround it during a demonstration. The biggest problem HARV faces is the slow sensor sampling which makes its reflexes quite slow. The vehicle can also be herded when running in this behavioral mode, but it is not quite as obedient, moving in the general direction forced upon it but occasionally making some slight sidesteps due to the presence of noise. Actually this form of herding is more reminiscent of an animal's behavior, and HARV has been likened to a sheep by some observers when it's running in this mode.

Figure 81: Avoidance behavior

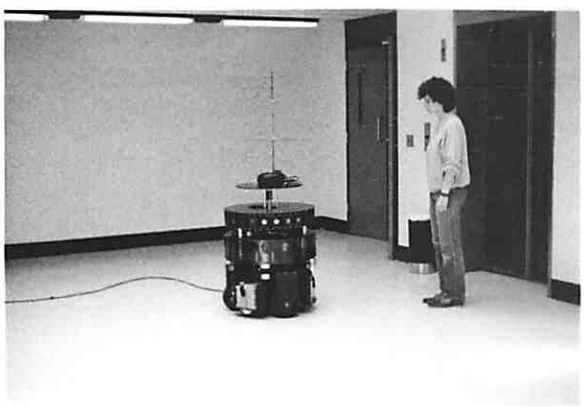
In this sequence, the robot is initially stationary (a). When the robot is approached (b), it is repulsed and moves to its new position and then stops (c). This behavior is produced by the instantiation of an avoid-static-obstacle schema.



(a)



(b)

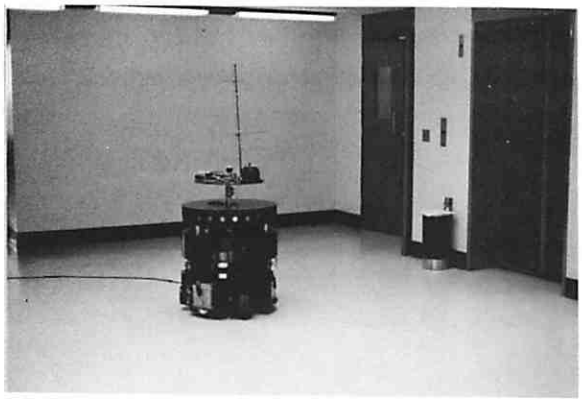


(c)

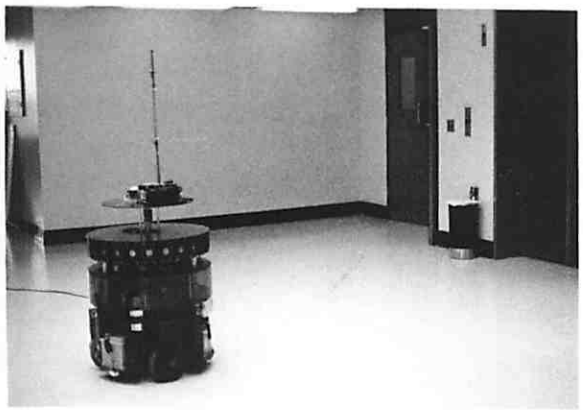
Figure 82: Exploration

Exploration behavior is produced by combining avoid-static-obstacle and noise schemas. The robot wanders randomly while avoiding collisions. (a-c). The last 3 photos (d-f) show the robot being "herded" towards the wall.

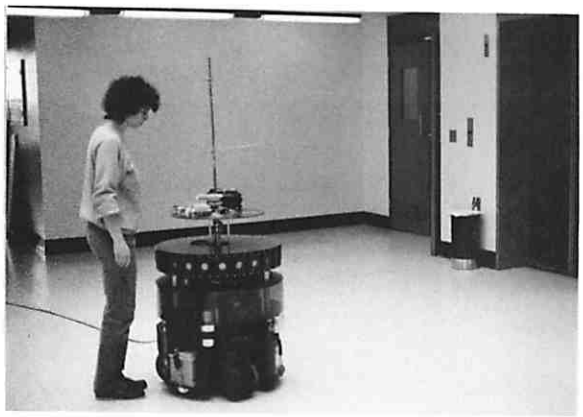
(a)



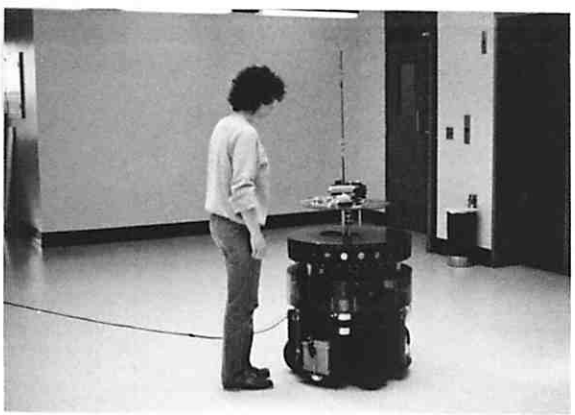
(b)



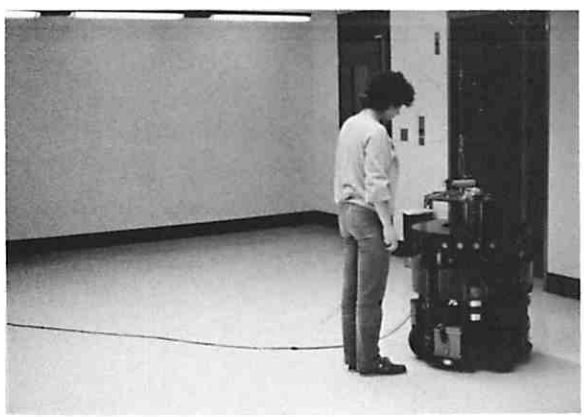
(c)



(d)



(e)



(f)

§2.3 *Hall following*

By instantiating the **move-ahead** schema with an **avoid-static-obstacle** schema, the robot is able to safely travel down a hall (Fig. 83). Although there is no model of the hall (at the schema level), the robot senses the two sides as obstacles and positions itself near the center. The **move-ahead** schema drives it forward.

HARV responds to changes in the hall structure itself such as pillars, seeking out the middle of the hall. Small obstacles placed near the wall's sides are interpreted as part of the wall by the ultrasonic sensors. Open doorways pose no problem as long as the **move-ahead** direction is roughly parallel to the hall. If this direction points steeply into the wall, the robot might pass through an open door (see door entry below), but since the angle into the wall typically must be at least 45 degrees or more for normal doorway entry, this allows a latitude of about 90 degrees (± 45) of error in the **move-ahead** direction for successful navigation of halls using this method.

§2.4 *Navigation in the presence of obstacles*

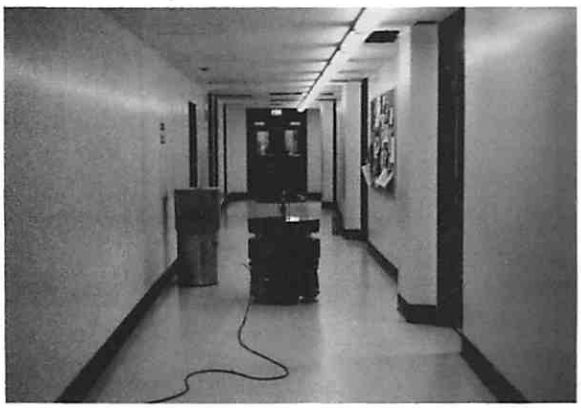
The same combination of schemas allows for navigation in cluttered hallways or outdoor situations. The **move-to-goal** schema can be substituted freely for the **move-ahead** schema if desired. Figures 84 and 85 show the robot's course through a series of cluttered obstacles both indoors and outdoors. The **move-to-goal** schema relies on the shaft encoders to move the robot to a particular point. This is accomplished by specifying the distance to and direction towards the goal and then monitoring the encoder data to provide the input to the **move-to-goal** SI.

The robot's minimum detectable ultrasonic sensor reading is 0.9 feet. For this reason, whenever the ultrasonic sensor returns a value of 0.9, the robot must consider a collision to be imminent. This effectively increases the robot's diameter by almost two feet, making it more difficult to squeeze through tight spaces. This is particularly evident in the indoor hallway examples. A consequence of this fact, when coupled with the slow sampling rates for ultrasonic data (2-3 seconds for a complete scan, transmission, and interpretation by the VAX), is the production of motion oscillations when the robot is operating under continuous motion in constricted areas. The vehicle moves from side to side in a dance-like motion as it squeezes through the congested spot. The robot still

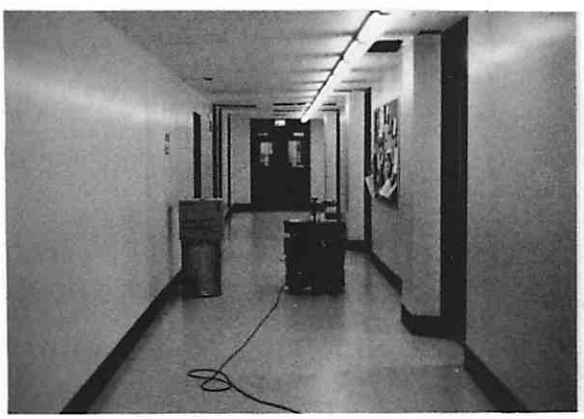
Figure 83: Hall following

This 5-photo sequence shows the robot moving down the hall under the control of move-ahead and avoid-static-obstacle SIs. Note that it detects and steers away from obstacles along the way. The robot maintains its center position both by virtue of the move-ahead SI and the fact that the walls of the hallway itself are repulsive due to the avoid-static-obstacle SI.

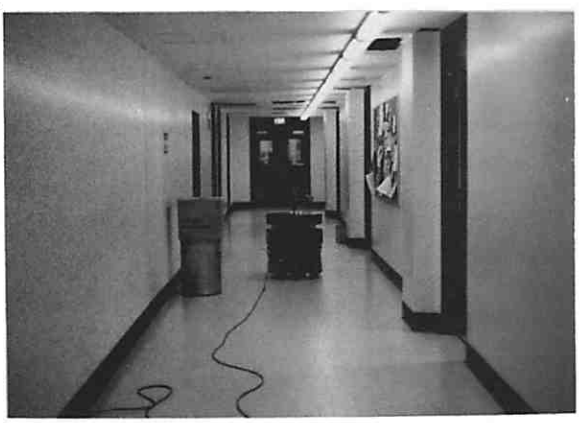
(a)



(b)



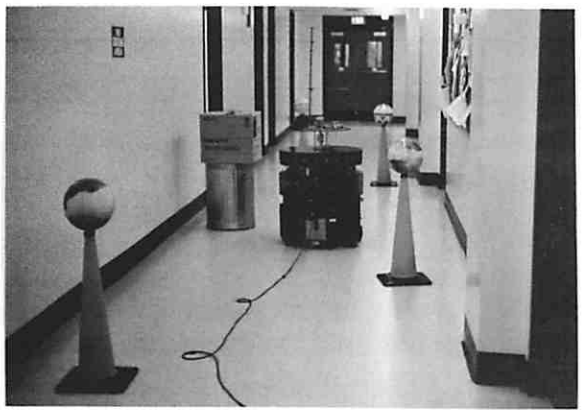
(c)

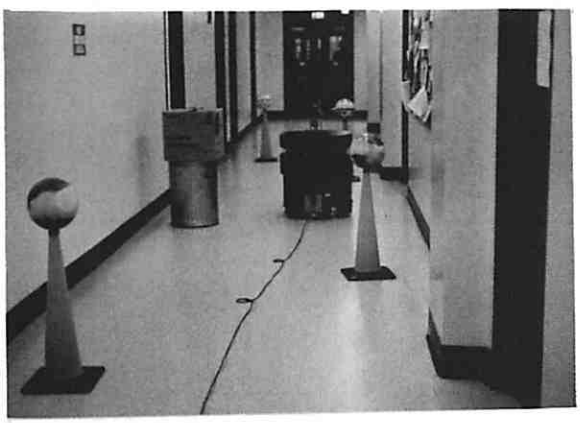


(d)

Figure 84: Navigation (Indoor)

In a more complex environment than that of Figure 83, the robot still succeeds in its navigational task of moving down the hall while avoiding collisions in the considerably more cluttered environment. (The balls on top of the cones make them better sonar targets, as the height of the traffic cones is just short of the height of our ultrasonic sensors in their particular geometric configuration).





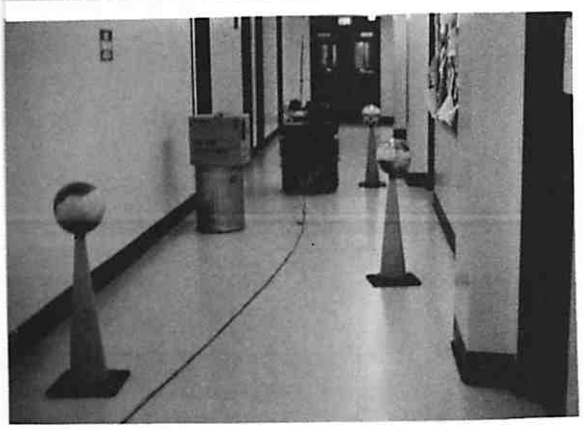
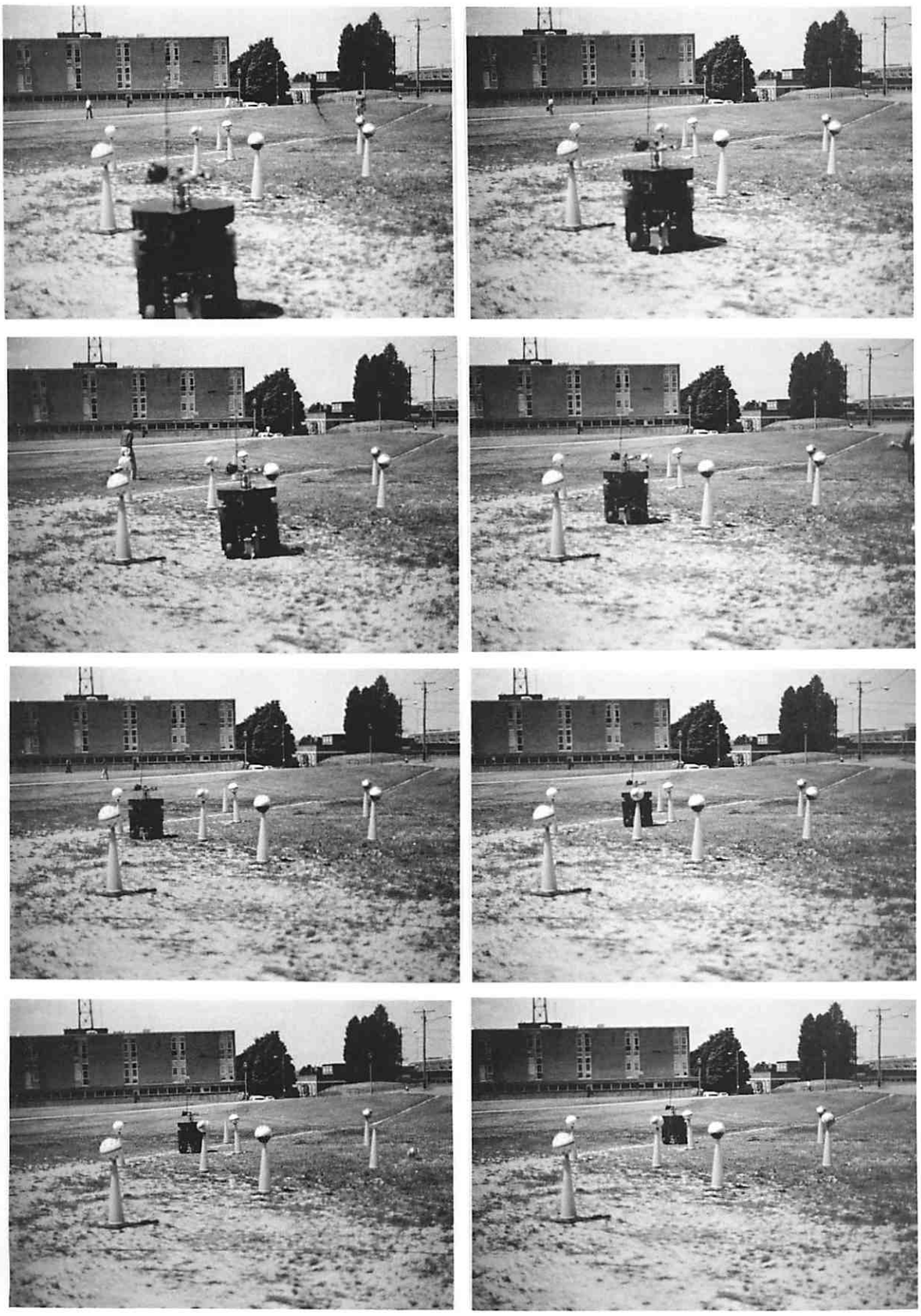


Figure 85: Navigation (Outdoor)

The same type of navigation can be performed outdoors as well. In this case, a move-to-goal SI (shaft-encoder based) was used instead of a move-ahead SI, the goal being a location near the distant cone that does not have a ball on it. The robot's course winds through the obstacles while making its way towards the goal.



attains its navigational goal and does not crash into obstacles, but it takes more time than would otherwise be necessary. This problem will vanish when the processing and communication speeds are improved. On the other hand, if the velocity of the robot is increased over the 0.3-0.4 feet per second used for these examples, the problem would be exacerbated. To attain higher velocities, faster processing is necessitated.

The experimental schema system offers three types of motion: step-by-step mode where each motor step must be approved by the operator (used chiefly for debugging new schemas), lurch mode where the robot waits approximately 2 seconds per step while sensor processing is completed between moves, and continuous motion where the robot acquires sensor data while moving. The problem with continuous motion is that the vehicle changes its position in the two or three seconds it takes to process the sensor data. Generally this is not a problem, but it can give rise, especially in tight quarters, to the oscillatory situations described above. All of the behaviors shown in this section work well in continuous motion with the possible exception of door entry (described below), again due to the tight quarters.

§2.5 *Single wall following*

A "drunken sailor" walk can be produced by directing a **move-ahead** schema into a wall with the obstacle avoidance behavior active (Fig. 86). The robot slides along a repulsive field a specified distance from the wall, reacting to obstacles as it moves. This allows the robot to enter doorways if the vector pointing into the wall is sufficiently large in magnitude and its angle is sufficiently steep. (Fig. 87).

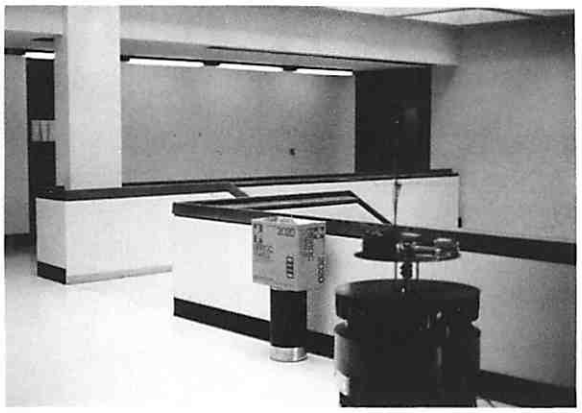
With normal doorways, it is difficult to get the robot to enter when it is operating in continuous mode. As stated above, this is a consequence of the data being old relative to the robot's current position. It also requires some finesse in proper selection of gains and angle of attack to produce smooth door entry behavior. With a 45 degree angle of attack and 1.3 gain (1.0 is baseline) for the **move-ahead** schema, and a sphere of influence of 2.5 feet and gain of 0.7 for the **avoid-obstacle** schema, good results have been obtained.

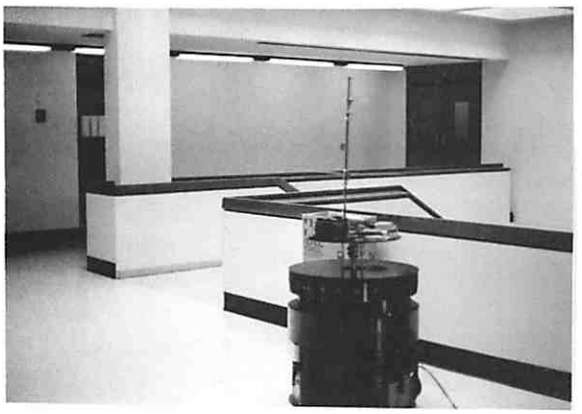
§2.6 *Impatient waiting*

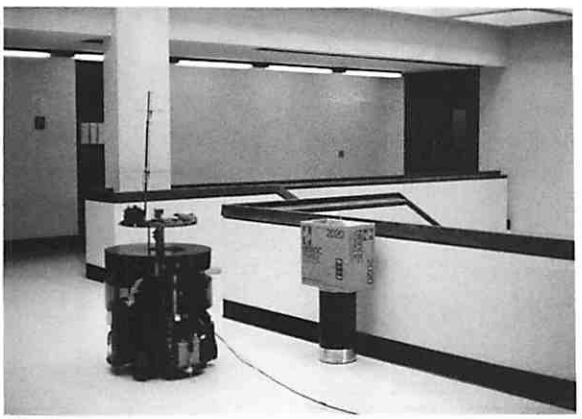
The potential fields methodology is strongly in evidence when the robot moves into a box canyon. The robot ends up in a potential well and is not able to make meaningful

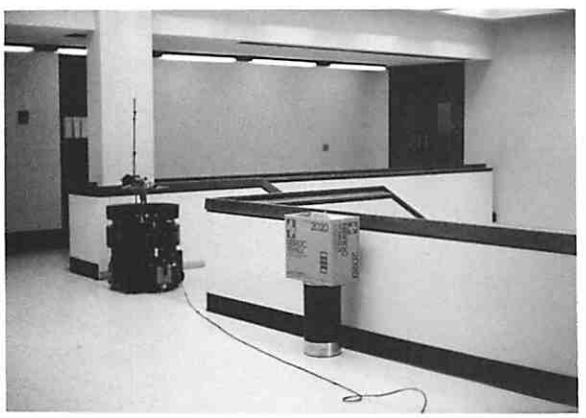
Figure 86: Wall following ("drunken sailor")

This behavior is produced by the combination of a move-ahead SI directed obliquely into the wall while keeping the obstacle avoidance behavior active. In the sequence shown, the robot follows the single wall, moving around the obstacle, and then staggers towards the staircase. The behavior appears as if HARV was leaning against the wall while moving forward, conforming to its contours. HARV would have fallen down the stairs (to the right), if it was not shut off by our technician (classic "drunken sailor" behavior).









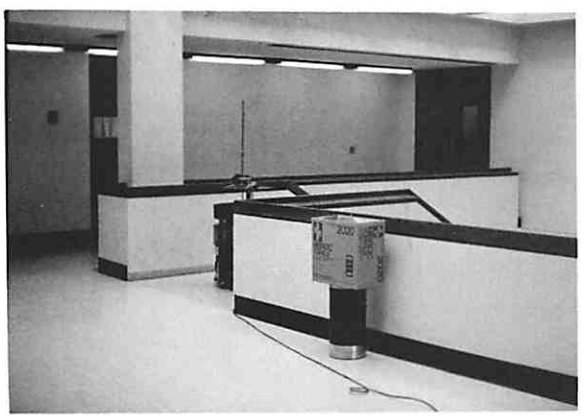
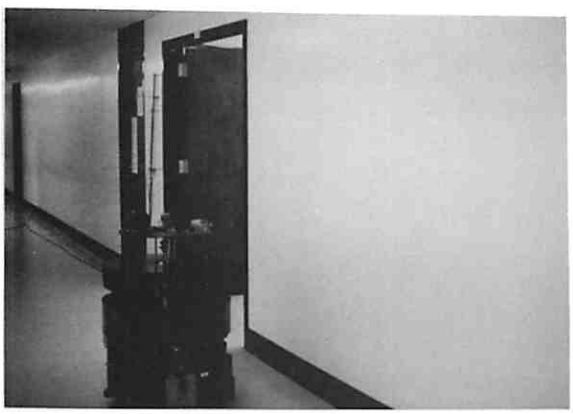
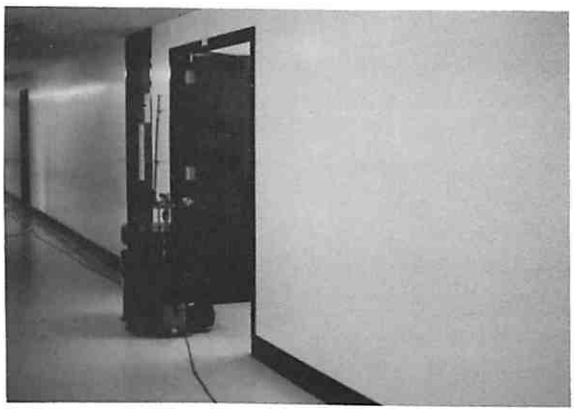
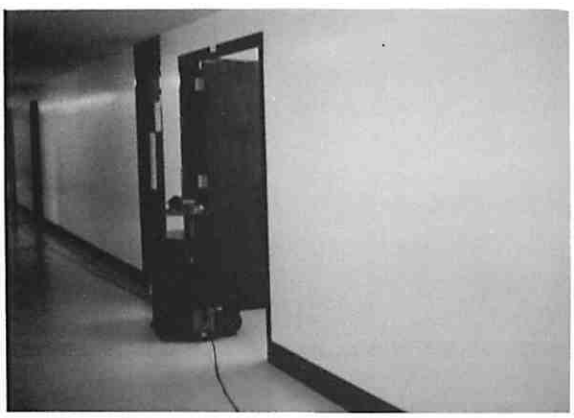


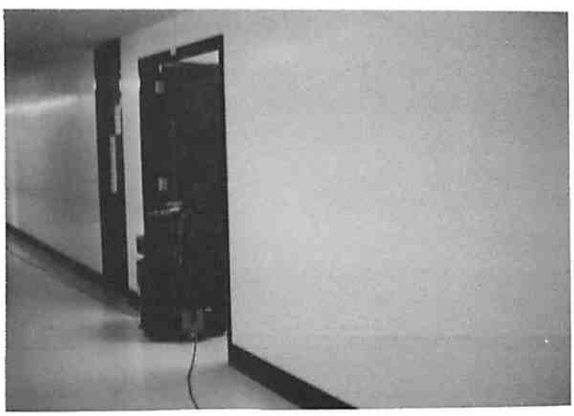
Figure 87: Door entry

The tendency of the robot to move through openings using the combination of behaviors described in Figure 86 can also be used to advantage in going through doors. In this sequence, the robot follows the wall and then enters the open door.









progress. Normally, after a time-out occurs based on a hard real-time deadline, the pilot and/or navigator would be reinvoked to compute an alternate path. Suppose, however, the blockage is temporary, due to closed doors at the end of a hall or an elevator. The robot can wait, rocking around in its potential well, until the obstruction is removed. This behavior is similar to what might be evidenced by a fly at a window. When the path becomes unblocked, the robot continues ahead, trying to satisfy its initial goals. Figure 88 depicts this process.

It is also possible to embed a temporal delay schema that turns off the robot's motion for a specified time interval, letting the robot deliberately wait before restarting. This incorporates aspects of patient waiting in addition to the impatient waiting described above.

§2.7 *Follow the leader*

HARV can track a moving object (with additional obstacle avoidance behavior to aid maneuvering in tight situations if desired). The appearance is similar to walking the robot on an invisible leash (Fig. 89). It turns and tracks the nearest object within a given ultrasonic sensor spread, moving more rapidly as the distance between robot and object increases, but avoiding contact with the tracked object. The potential well (the ideal distance separating the tracked object and the robot) is user specifiable as is the separation distance at which the robot abandons following. Vision algorithms, well suited for this particular type of tracking, will hopefully be exploited for this purpose when suitable real-time image processing hardware arrives.

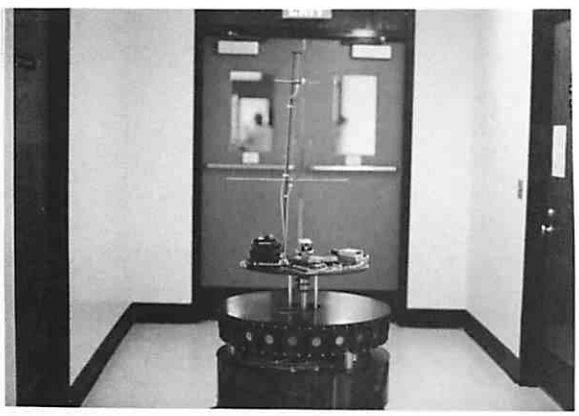
§2.8 *Motor schema experiment summary*

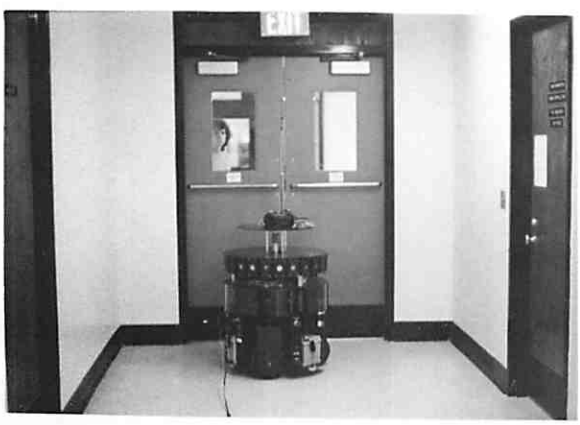
The motor schema experimentation/demonstration system enables visualization of motor schema-based potential field navigation. The greatest drawback lies in the relatively slow sampling rates for processing the ultrasonic data on a shared VAX-750 (typically 2-3 seconds for 24 sensor readings) causing minor oscillations of the vehicle in highly dynamic or extremely cluttered environments when continuous motion is allowed, since the sensor data becomes old relative to the robot's position.

Enhancements would include the addition of other perceptual schemas using vision, particularly if they can be made to function more rapidly than the current implementa-

Figure 88: Impatient waiting ("fly at a window")

This sequence shows what can occur when the robot enters a potential well. A move-ahead SI directs the robot to the end of the hall, while the avoid-static-obstacle behavior keeps it from colliding with the door. The robot roves impatiently at the end of the corridor until the door is opened. At that point, it moves through the door and continues on its way.





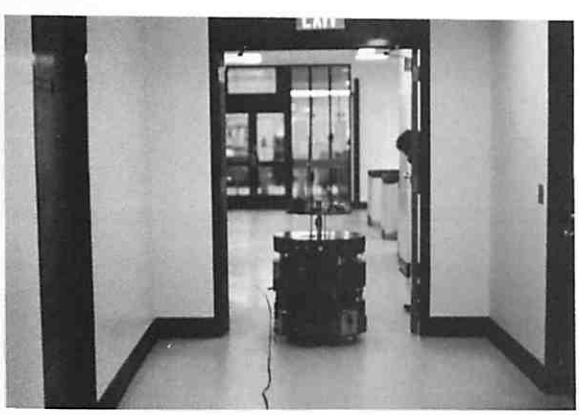
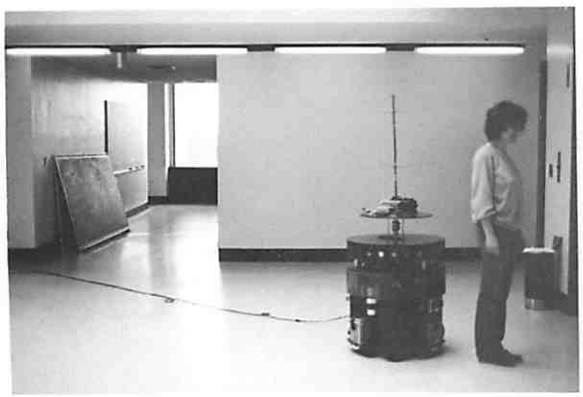


Figure 89: Follow-the-leader behavior

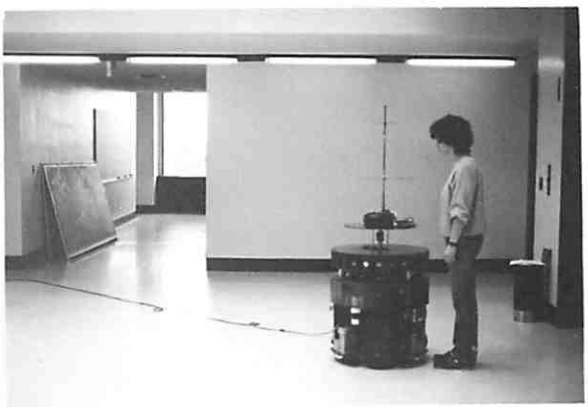
Although difficult to show in a sequence of still photos, the robot can be seen to follow Robyn across the floor, turning when she turns. When she stops (d-e), the robot settles into a potential well a short distance from her and then stops. This behavior mimics walking a dog, but with an invisible leash.

(a)

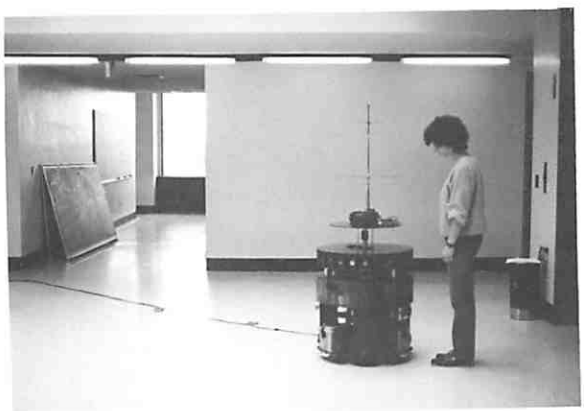
(b)



(c)



(d)



(e)

tions of the vision algorithms described below. Converting the system to a distributed environment will reduce the delays incurred by the sequential evaluation of schema velocity vectors when using the VAX, further enhancing the robot's sensitivity to its world. It can be seen however, that by using combinations of relatively primitive behaviors, sufficiently intelligent action can be undertaken by the robot to successfully navigate through its world.

§3. Visual navigation experiments

The vision algorithms were tested individually, principally because of the longer times required for their processing. Three algorithms were tested including path following using the fast line finder, path following using the fast region segmenter, and obstacle detection using the depth-from-motion algorithm. Path following cannot be implemented using the DRV's ultrasonic sensors, thus the visual path detection algorithms significantly increase the robot's ability to navigate in an outdoor environment. The details of these algorithms are described in Chapter 6 and Appendix C. This section presents the experimental results.

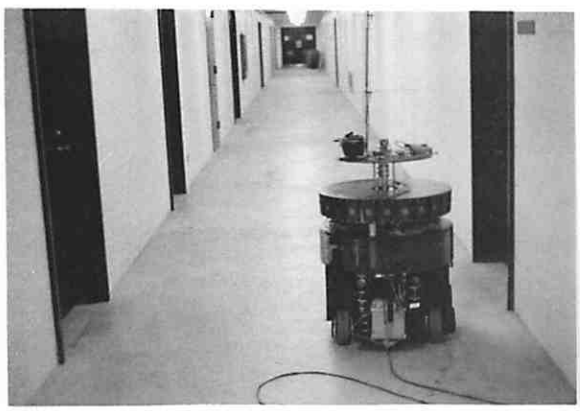
§3.1 Path following using fast line finding

The robot was tested on both an interior hallway and an asphalt sidewalk outside the GRC. In both cases the robot successfully extracted the path edges and servoed itself to remain on the path. (See Figs. 90-93). The bootstrap position of the path edges was provided for the the robot by the experimenter in each case. All the feedforward path edge positions were computed automatically. In the indoor run (Figs. 90-91), the robot was initially started at the side of the hall, quickly found the centerline, moved to the hall center, and then continued to follow the hall successfully. Near the end of the corridor, the image contained little path edge for the robot to follow and the run was terminated.

In the outdoor run (Figs. 92-93), similar behavior was observed. Since the UHF transmitter was not operational for these runs, the testing was limited to those paths that were accessible using our 500 foot cable tether. Whenever a path had reasonably well defined edges, the fast-line finder was able to provide the information necessary for visual servoing. It failed however for the case of the gravel path which had quite

Figure 90: FLF path following - indoors

This sequence of photos shows the robot finding the centerline of the hallway and then maintaining its position on the centerline. The algorithms described in Chapter 6 (FLF, line fragment grouping, centerline computation, servoing, etc.) produce this well-defined behavior. Figure 91 shows the robot's view of the hall.



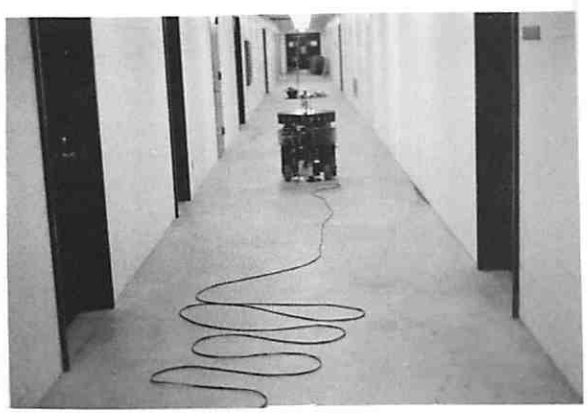


Figure 91: FLF path following - indoors (continued)

The lefthand images show the results of the fast line finder with the buckets tuned to the anticipated hall edges, and the righthand images show the extracted left and righthand path edges and the computed centerline used for servoing (see Fig. 90). Images (a-b) show the initial position of the robot, images (c-d) show the view after the first servo sequence is completed, (now in the center of the hall), and (e-f) show an image taken from much further down the hall, demonstrating the robot's continuing ability to extract the path edges.

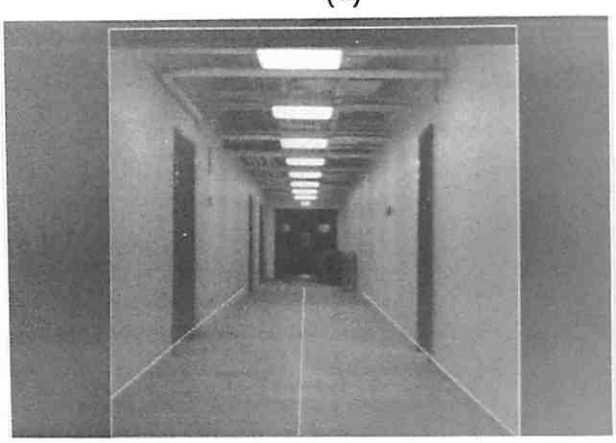
(a)

(b)

(c)

(d)

(e)



(f)

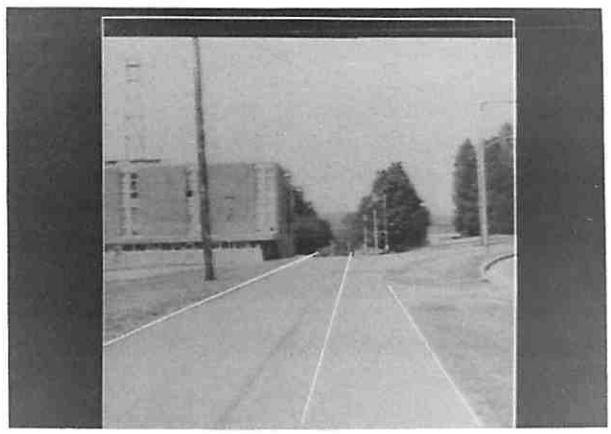
Figure 92: FLF path following - outdoors

The robot, using the same algorithm as in Figures 90-91, navigates along an asphalt sidewalk. The robot is started off to the left, then after finding the path, moves to the center of the sidewalk, and then remaining in the center of the sidewalk for the rest of the run.

Figure 93: FLF path following - outdoors (continued)

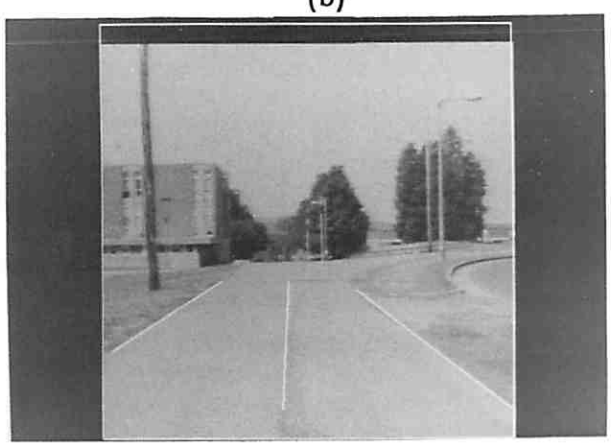
The lefthand images show the results of the fast line finder with the buckets tuned to the anticipated sidewalk edges, and the righthand images show the extracted left and righthand path edges and the computed centerline for servoing (see Fig. 92). Figures (a-b) show an image taken from the initial position of the robot (off to the side of the path) that is used for the first step in path following, Figure (c-d) shows the view after the first visual servo sequence is completed and the second step of FLF processing done, (the robot is now in the center of the sidewalk) and Figures (e-f) show the results from an image taken from much further down the sidewalk, showing the robot's continuing ability to extract the path edges.

(a)



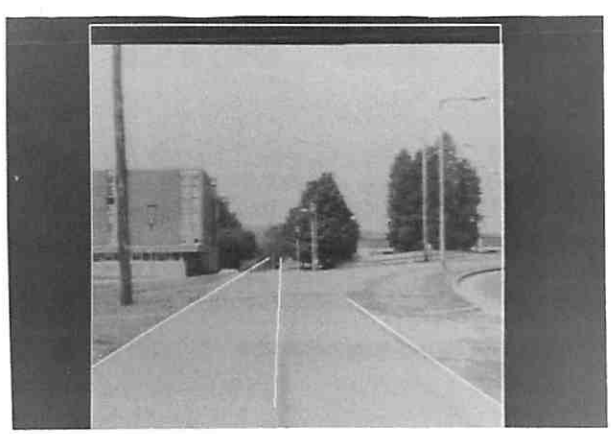
(b)

(c)



(d)

(e)



(f)

indistinct edges. Visual path following was still attainable for this case however, using the fast region segmenter as the perceptual process (see section following).

Topography plays a role in outdoor navigation using the FLF. If the ground drops off due to a change in the terrain's slope, the vanishing point of the path appears above the actual horizon. As a consequence, lines may be extracted in trees, buildings, etc., which are added to the line fragment collection and may cause the extracted path edges to deviate from their actual position. This problem can be obviated by pointing the camera downwards, so the horizon is above the field of view, but this will make landmark recognition from the same image difficult, as the road becomes the most prominent feature. Another alternative is to use high-level knowledge that reflects the topography to lower the horizon and remove any extraneous line fragments before combining them to produce the path edges. A more sophisticated approach would involve geometric grouping of long collinear lines as in [143].

The FLF algorithm as applied to path following, however, is quite robust and requires little tuning to produce good results. For the runs above, a 40 degree range for line orientation proved more than adequate. The expectations in general do not have to be highly constrained to produce good, working results. This robustness is largely a consequence of the use of gradient orientation as opposed to gradient magnitude as the means for producing the line fragments.

§3.2 *Path following using fast region segmentation*

The robot was placed in a similar test environment as for fast line finding. In this case, however, a gravel path with quite indistinct path borders was used. Under these irregular conditions, at the image resolution used, the fast line finder had difficulty extracting the path borders (smoothing the image prior to processing may help the FLF find the path). Using the fast region segmenter (FRS), the robot successfully negotiated this course (Fig. 94). Figure 95 shows the operation of the FRS algorithm on the images used during this run.

This algorithm, being sensitive to the gradient magnitude as opposed to the gradient orientation as in the case of the fast line finder, is considerably less robust than the FLF to changing lighting conditions and thus is not particularly well suited for outdoor navigation. Changing lighting conditions (e.g. passing clouds) make this algorithm, as

(a)

(b)

(c)

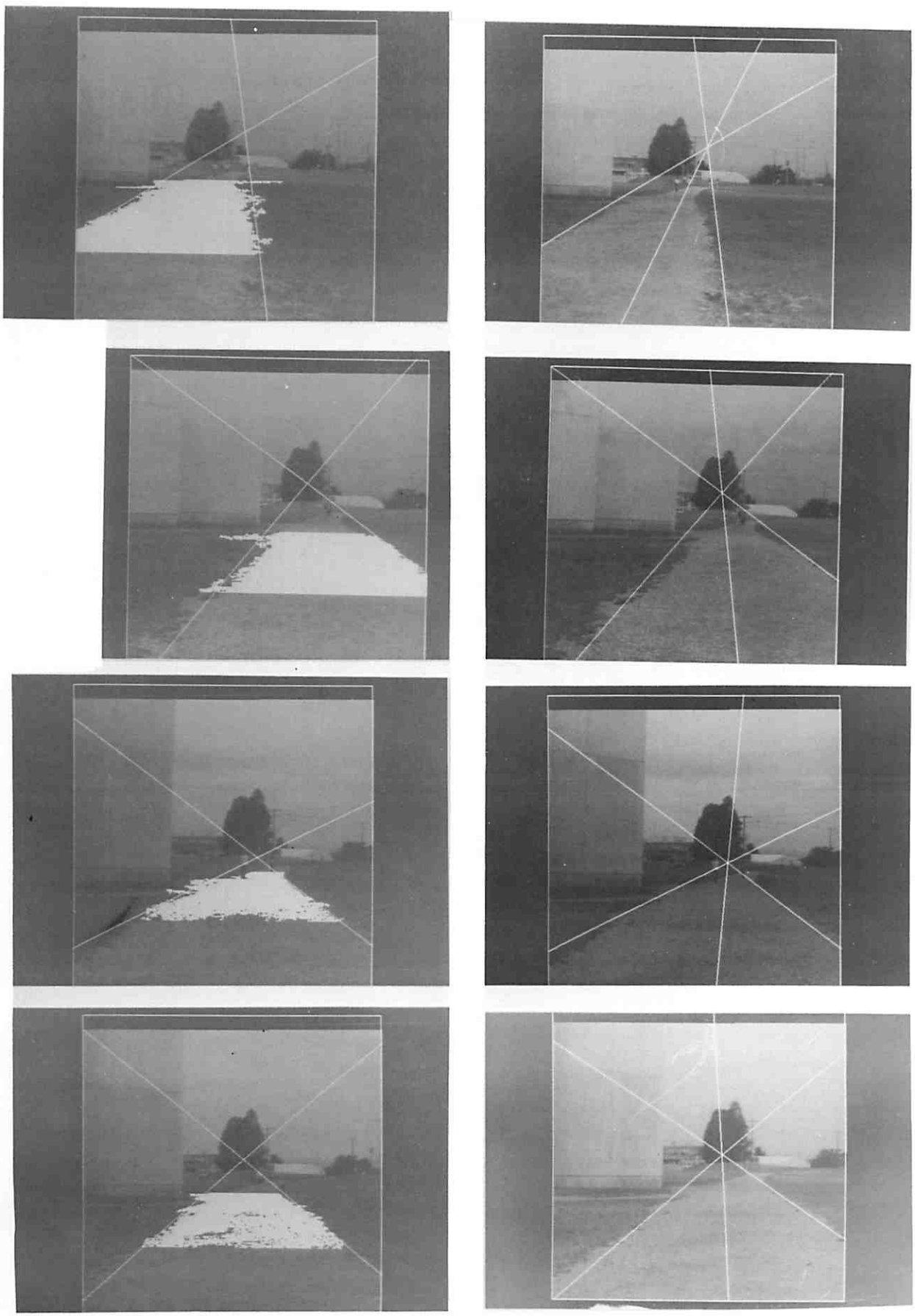
(d)

(e)

Figure 94: FRS path following
This sequence shows the robot's path when navigating under the fast region segmentation algorithm over a gravel path.

Figure 95: FRS path following - (continued)

The lefthand images show the results of the fast region segmentation. Lines are fitted to the left and right sides of the region using the least squares method. The righthand images show the computed centerline used for servoing purposes. Only the central portion of the path was extracted to obtain more consistent and uniform intensity values for the segmentation.



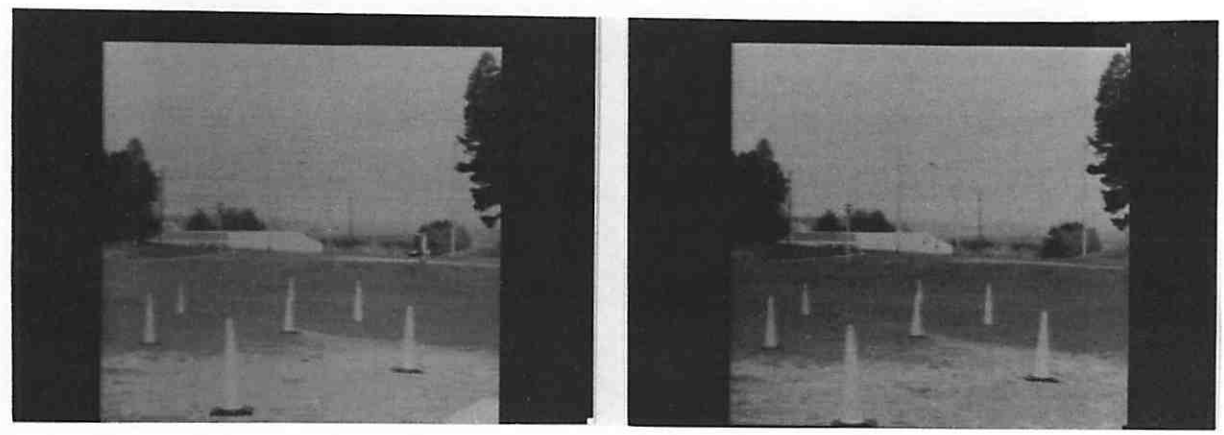
it currently stands, susceptible to failure. The use of color images would significantly increase its versatility as additional spectral information would aid in the separation of the gravel region from the surrounding grassy areas. For indoor environments with carpeted halls and predictable lighting conditions, the FRS algorithm can be expected to perform well.

§3.3 *Depth-from-translational-motion results*

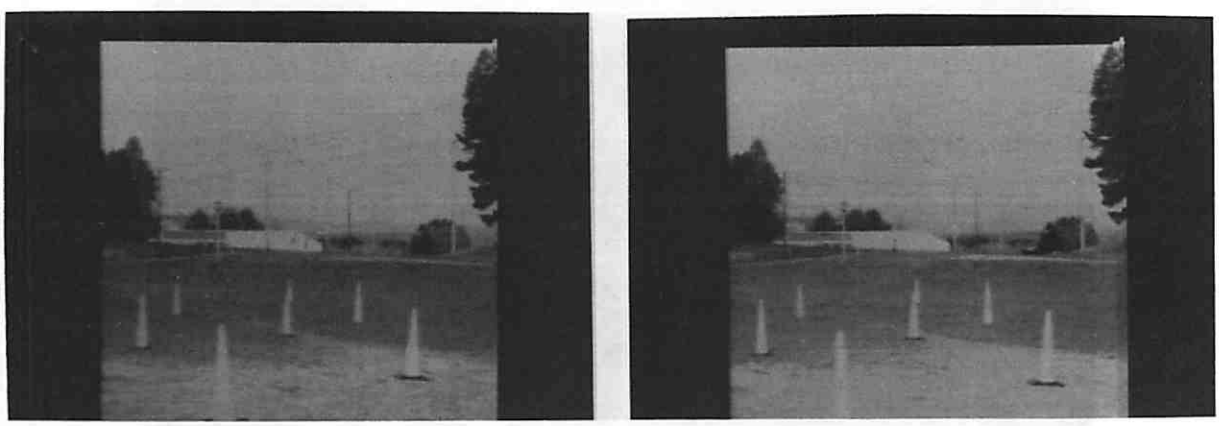
The depth-from-motion algorithm was tested with images acquired from HARV, but was not used to drive the vehicle. This algorithm in its current hardware implementation is far too slow to provide reasonable control of the robot. Two image sequences were used, the first outdoors and the second indoors. The outdoor images were hand-registered (an automated registration process is being developed [106]) while the indoor sequence was used as is (no registration was performed). Proper image registration is crucial to accurate recovery of depth. Although successful point tracking can occur even when image sequences are not well registered, the actual depth values may be inaccurate. In both cases the FOE was automatically extracted. The three image sequences themselves are shown in Figures 96 and 99. The tracked points and extracted obstacles are shown in Figures 97 and 100. Depths to several of the extracted and tracked obstacle points are tabulated in Tables 4 and 5. Motor schemas were then attached to the obstacles in the outdoor case in a simulation run and produced the path shown in Figure 98.

The major problem associated with accurate depth recovery, as stated in Chapter 6, is the difficulty in obtaining accurately registered images. HARV acquired the outdoor image sequence while traveling over a concrete pad used as a parking lot at the GRC. The pad itself is not completely level and there is considerable gravel and dirt covering it. This caused the camera to roll somewhat from one image acquisition to the next. The manual registration process used for the outdoor sequence did not attempt to correct for this rotation in the image plane. Only image translations were used to minimize the misregistration. Pitch and yaw movements of the camera can only be partially removed by image translation [106]. Roll of the camera, nonetheless, still remains the biggest problem in registration.

The numerical results (Tables 4,5) are somewhat discouraging. Anomalous behavior, such as the depth to an object increasing as the robot approaches, can be seen in the



(a) (b)



(c) (d)

Figure 96: Outdoor depth-from-motion image sequence

(a)

(b)

Figure 97: Outdoor depth-from-motion obstacles

(a) The tracked points, associated with each cone top through all four frames, are shown superimposed on the first frame.

(b) The results of the obstacle extraction algorithm (thresholded at 50% of the points above the least squares-line fit - see Chapter 6). The three closest cones are successfully detected as obstacles.

Table 4: Outdoor depth-from-motion results

object	feature	nominal depth			true depth (feet)		
		1-2	2-3	3-4	$frame_2$	$frame_3$	$frame_4$
cone1	0	33.1	64.4	65.6	35.1	33.2	31.3
	1	30.3	55.3	66.2	35.1	33.2	31.3
	4	36.5	31.1	38.4	35.1	33.2	31.3
	10	36.3	38.0	33.0	35.1	33.2	31.3
	18	33.7	—	—	35.1	33.2	31.3
cone2	9	37.1	41.7	39.8	45.1	43.2	41.3
	11	38.8	46.7	39.7	45.1	43.2	41.3
	15	46.2	51.3	49.3	45.1	43.2	41.3
cone3	5	47.6	54.6	51.4	55.1	53.2	51.3
	6	51.8	56.1	58.2	55.1	53.2	51.3
	14	61.9	60.8	56.1	55.1	53.2	51.3
cone4	8	90.8	85.7	100.1	65.1	63.2	61.3
	13	83.2	88.6	101.1	65.1	63.2	61.3
cone5	2	73.9	85.3	109.4	75.1	73.2	71.3
	12	88.9	107.5	94.6	75.1	73.2	71.3
cone6	3	107.1	106.9	98.2	85.1	83.2	81.3
	7	113.2	146.4	—	85.1	83.2	81.3
	23	94.0	126.8	92.7	85.1	83.2	81.3

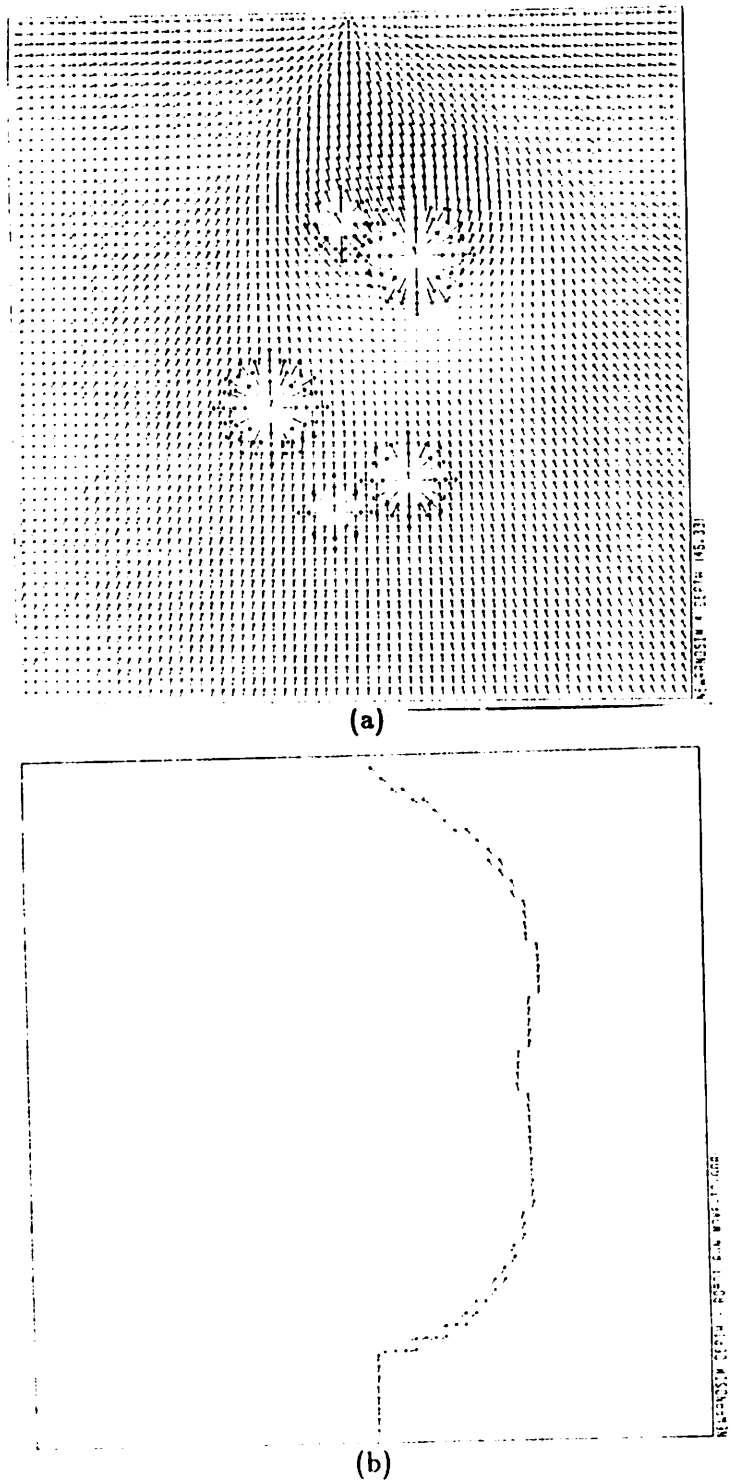


Figure 98: Depth-from-motion schema simulation

a) The position of 5 of the tracked obstacles plus a goal produce the velocity field shown here. The robot is located at the bottom of the vector field with the goal at the top.

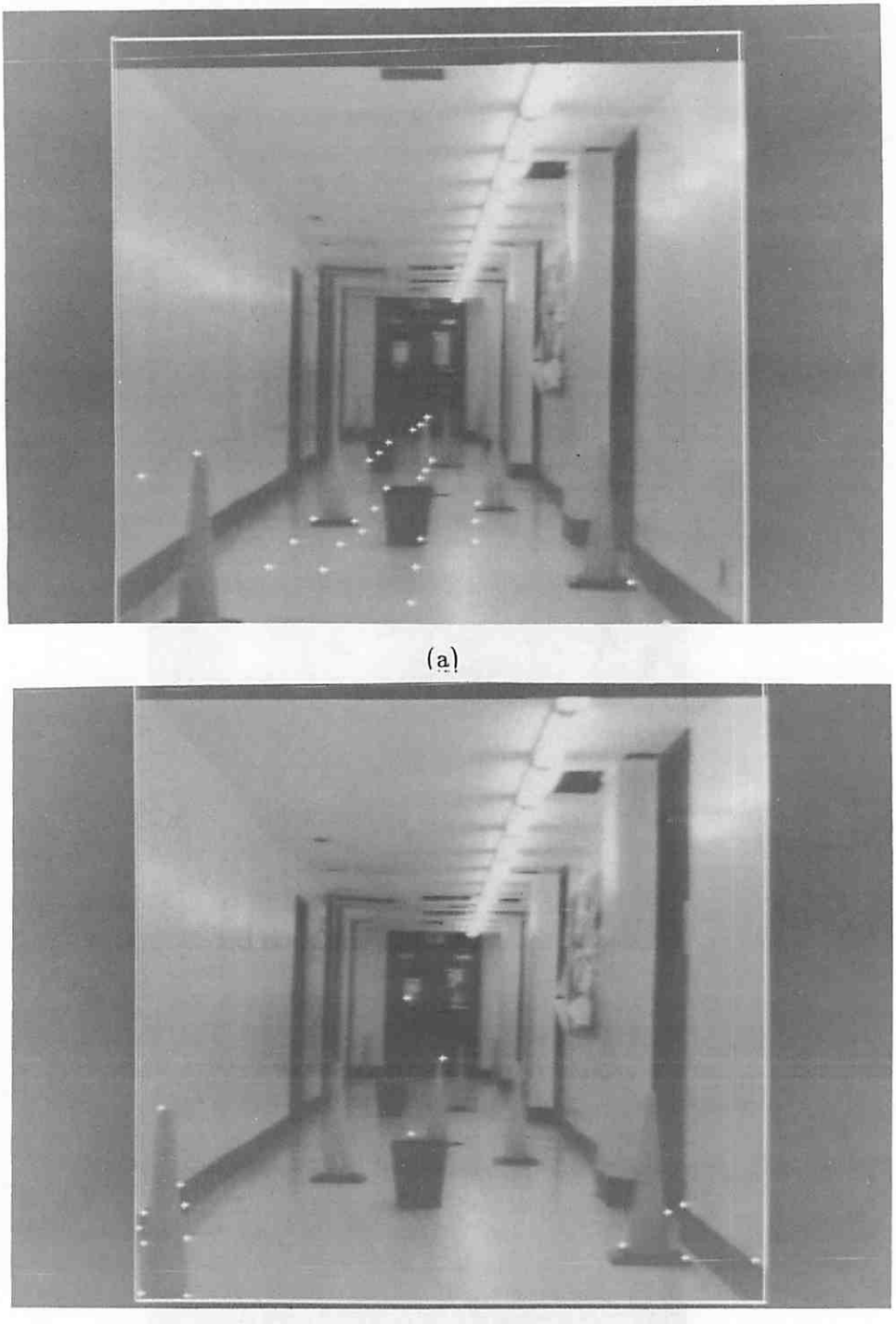
b) The robot's course through the obstacle field to the goal.

(a)

(b)

(c)

Figure 99: Indoor depth-from-motion image sequence



(a)

(b)

Figure 100: Indoor depth-from-motion obstacles

(a) Tracked points, associated with several obstacles through all three frames, are shown superimposed on the first frame.

(b) The results of the obstacle extraction algorithm (thresholded at the top 50% of the points above the least squares line fit)). The two closest cones, the closest can, an additional cone, and the door were all detected as obstacles.

Table 5: Indoor depth-from-motion results

object	feature	nominal depth		true depth (feet)	
		1-3	3-5	$frame_3$	$frame_5$
cone1	18	25.1	—	16.1	12.2
	42	23.0	—	16.1	12.2
	45	23.2	18.7	16.1	12.2
	55	23.5	—	16.1	12.2
	68	22.4	24.4	16.1	12.2
	71	22.2	—	16.1	12.2
	72	22.3	—	16.1	12.2
	91	22.5	25.8	16.1	12.2
	92	22.6	33.3	16.1	12.2
cone2	11	26.4	36.5	21.1	17.2
	15	24.3	25.9	21.1	17.2
	16	28.7	29.1	21.1	17.2
	46	26.4	23.0	21.1	17.2
	50	27.1	26.3	21.1	17.2
	63	28.0	28.4	21.1	17.2
cone3	41	49.9	44.5	31.1	27.2
	44	39.8	38.8	31.1	27.2
	47	41.5	40.2	31.1	27.2
cone4	13	51.1	49.2	36.1	32.2
	43	50.6	54.3	36.1	32.2
cone5	33	57.5	82.3	41.1	37.2
	51	71.7	85.5	41.1	37.2
	58	57.1	61.3	41.1	37.2
cone6	60	64.6	84.4	56.1	52.2
	65	66.1	84.2	56.1	52.2
can1	3	38.476	34.34	26.1	22.2
	8	40.827	38.9	26.1	22.2
	9	38.171	34.5	26.1	22.2
can2	19	54.2	63.9	51.1	47.2
	32	52.0	62.1	51.1	47.2
	52	54.0	66.5	51.1	47.2

analysis of later frames in a sequence. However, the tracking of the object features themselves is quite robust and leaves some basis for optimism in further development of the algorithm (Figs. 97 and 100). The causes for the numeric error become apparent on inspection of the tracked points. For example, note the two leftmost cones in Fig. 97a. Curvature of the point tracks can clearly be seen. This is attributed to rotation in the image plane, a known problem that can produce errors such as those above.

For the indoor sequence (Fig. 100) the images were not manually registered as the floor was level, smooth, and clean. Drift of the vehicle and eccentricities in its drivetrain are sufficient in themselves to produce registration errors nonetheless. Hand registration was impractical since there are no truly distant points to use as invariants from frame to frame, and only those points extremely close to the FOE can be assumed not to undergo displacement as motion proceeds (these are the points needed to register the images accurately). However, the FOE cannot be accurately found unless the images are registered and there is no easily extracted set of points available from this monocular image sequence with which to carry out the registration process. The most distant trackable points were clustered at the end of the hall and would not work well with the software registration algorithm described in [106].

Although the numeric results were not as accurate as we would have liked to see, many of the obstacles were still extractable. In the outdoor sequence (Fig. 97b) the tips of the three closest cones were detected as obstacles. For the indoor run (Fig. 100b) the two closest cones, the nearest can, one further cone, and the door were extracted. Although the numeric values for depth returned were poor (indoor case), the obstacle detection algorithm worked well enough to extract the closer obstacles. The depth values themselves are not usable in this current form. Partial blame can be assigned to misregistration. The results obtained for the example in Chapter 6 were much better.

An interesting observation is in the tracking of points formed by occlusions of cones with the path edge or the road edge. The point trackings are generally well behaved but their image displacements are not dependent entirely on either the depth to cone or to the road. An occlusion of the edge of a nearby cone with a nearly vertical and distant line in the image could produce large pixel displacements for relatively small vehicle translations due to the slope of the cone's side, giving rise to substantial errors in depth. Fortunately, most of the points tracked in these examples did not exhibit this type of behavior.

What is of the most significance for this algorithm is that although the depths recovered were not always as accurate as hoped for, the tracking of the feature points was in general reliable (Figs. 97a and 100a). By improving on our FOE detection and image registration methods, particularly by introducing a stabilized platform for image acquisition, the accuracy in the depth recovery is predicted to increase dramatically. Research on these issues is continuing at the University of Massachusetts for application to HARV. This collection of algorithms is being refined and will be sent to Carnegie-Mellon University for use on their NAVLAB vehicle. There we expect to be able to obtain images that have additional hardware support for solving the registration difficulties described above.

§4. Localization and uncertainty management

Localization, correlating the robot's position with the long-term memory representation, was tested by placing the robot in a known outdoor location and using the information available in the environmental model, in conjunction with the terrain measures and distance traveled, to control the spatial uncertainty map. The motion of the robot through its world map is depicted in Figure 101.

The robot started at a position known to an accuracy of ± 1 foot relative to the landmarks. This was probably an overestimate of the positional accuracy. Four images were captured (Fig. 102), first from the robot's starting point and then as it moved 10 feet, then 20 feet and finally 40 feet over the sidewalk. Figure 103 shows how the spatial uncertainty map changes as the robot moves from position to position. In this particular case the orientational uncertainty is reduced appreciably after each landmark recognition.

Three landmarks (a telephone pole, a concrete lamppost, and the corner of the East Engineering building) were used. All were treated as Class I landmarks. The building corner initially was treated as a Class II landmark, but it consistently appeared lower in the image than it was predicted to. On further analysis, this was found to be a consequence of the failure of the ground plane assumption in two ways. First, the corner's height from the ground as determined from architectural drawings did not indicate to us that the building's foundation was about 10 feet lower than the ground plane of the path that HARV was on. After compensation was made for this (only an approximation),

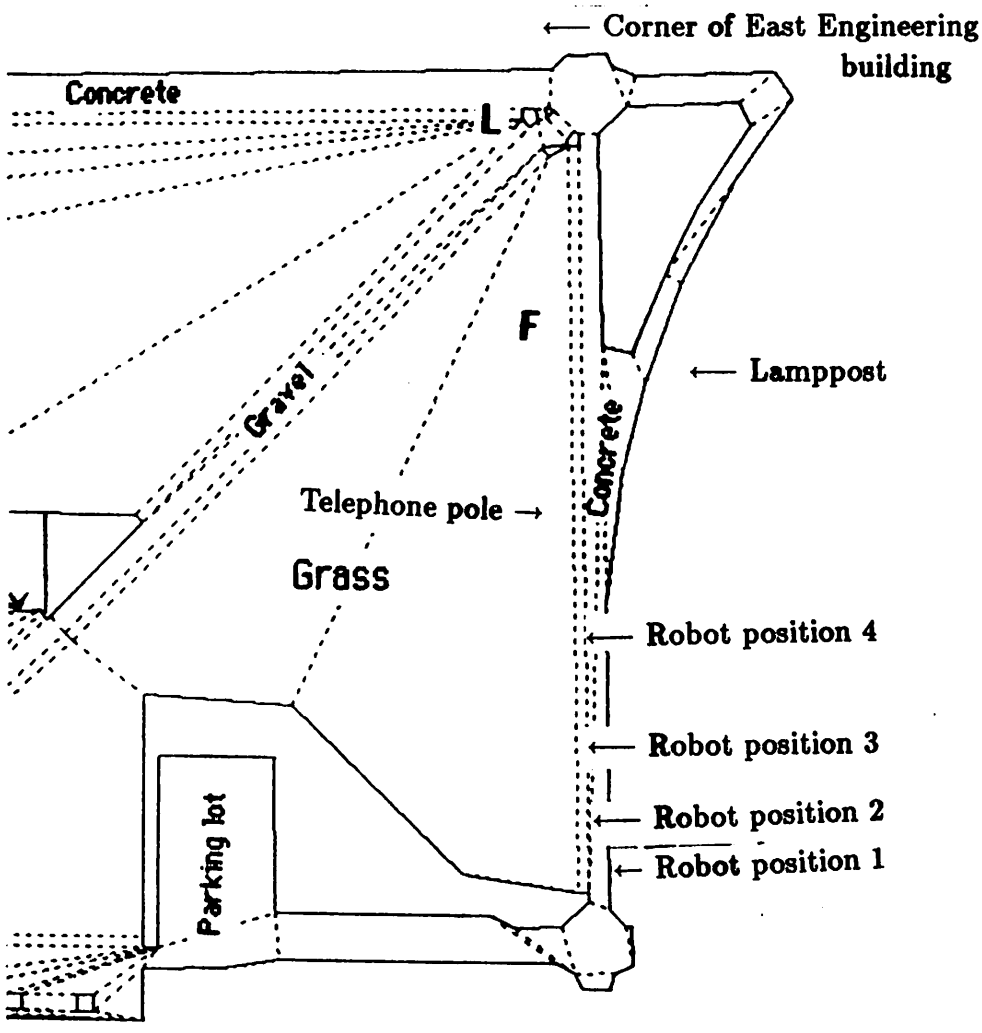
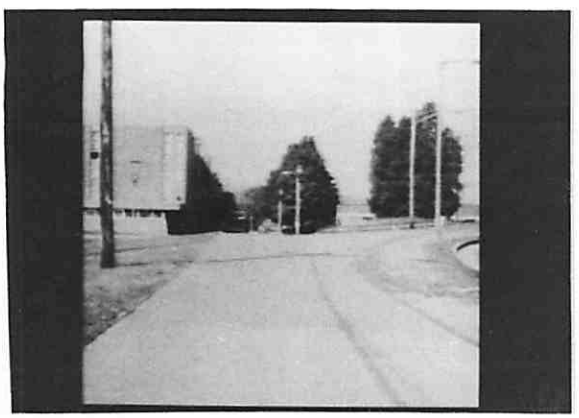


Figure 101: Localization experiment - Robot's position in LTM

Position 1

Position 2



Position 3

Position 4

Figure 102: Localization experiment image sequence

Figure 103: Localization experiment spatial uncertainty map

Here the spatial uncertainty map is pruned via landmark recognition. Most of the improvement occurs in orientation uncertainty which is reduced considerably. The left hand column shows the map before recognition and pruning. The right hand column shows the reduction accomplished by landmark recognition. The concrete pole landmark is used to reduce the uncertainty for maps (a) and (e), the telephone pole is used to reduce the uncertainty in map (c), and the corner of the East Engineering building is used for map (g).

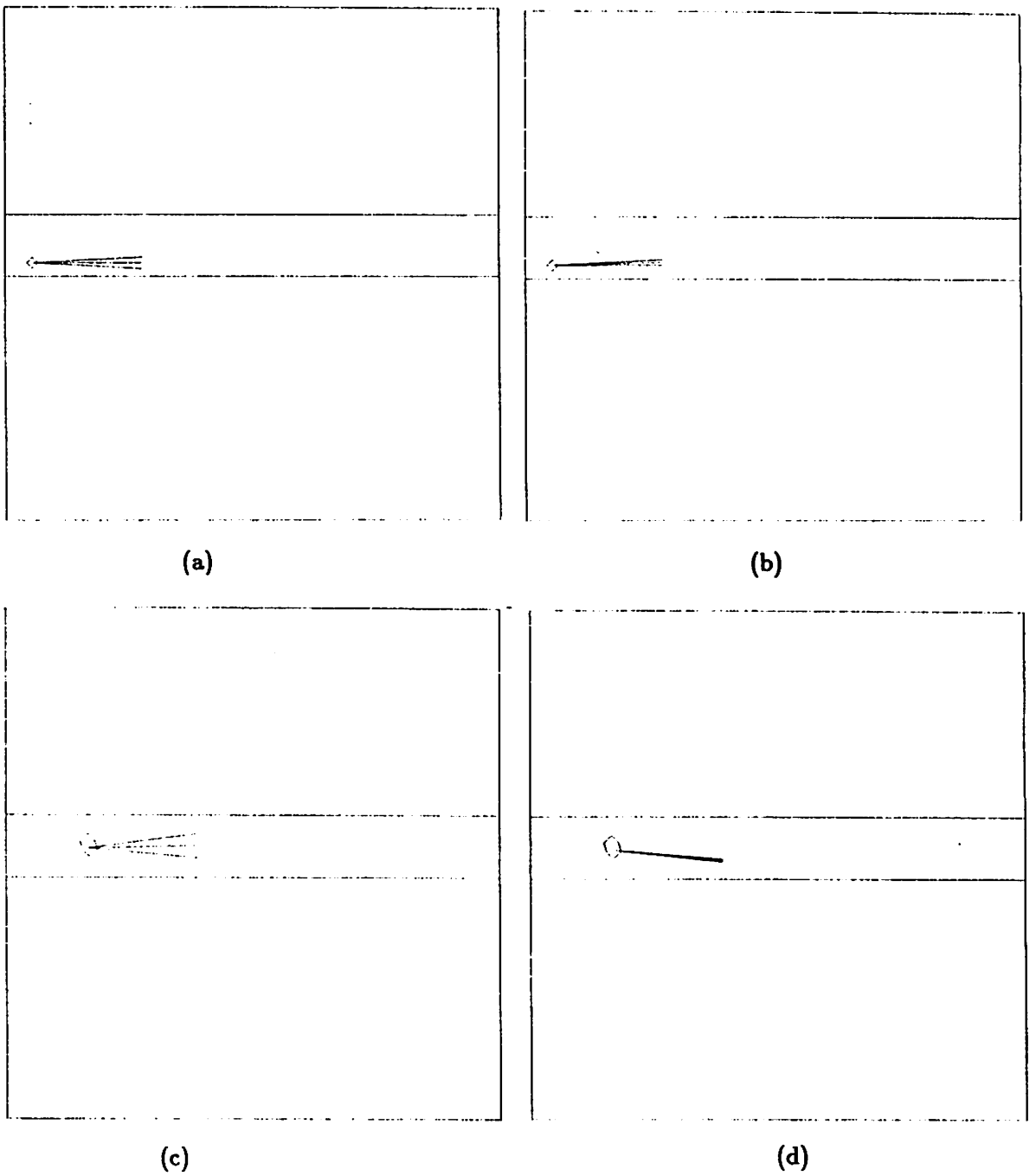
(a-b) Position 1, before and after pruning.

(c-d) Position 2.

(e-f) Position 3.

(g-h) Position 4.

(Figure continued on following page).


(a)
(b)
(c)
(d)

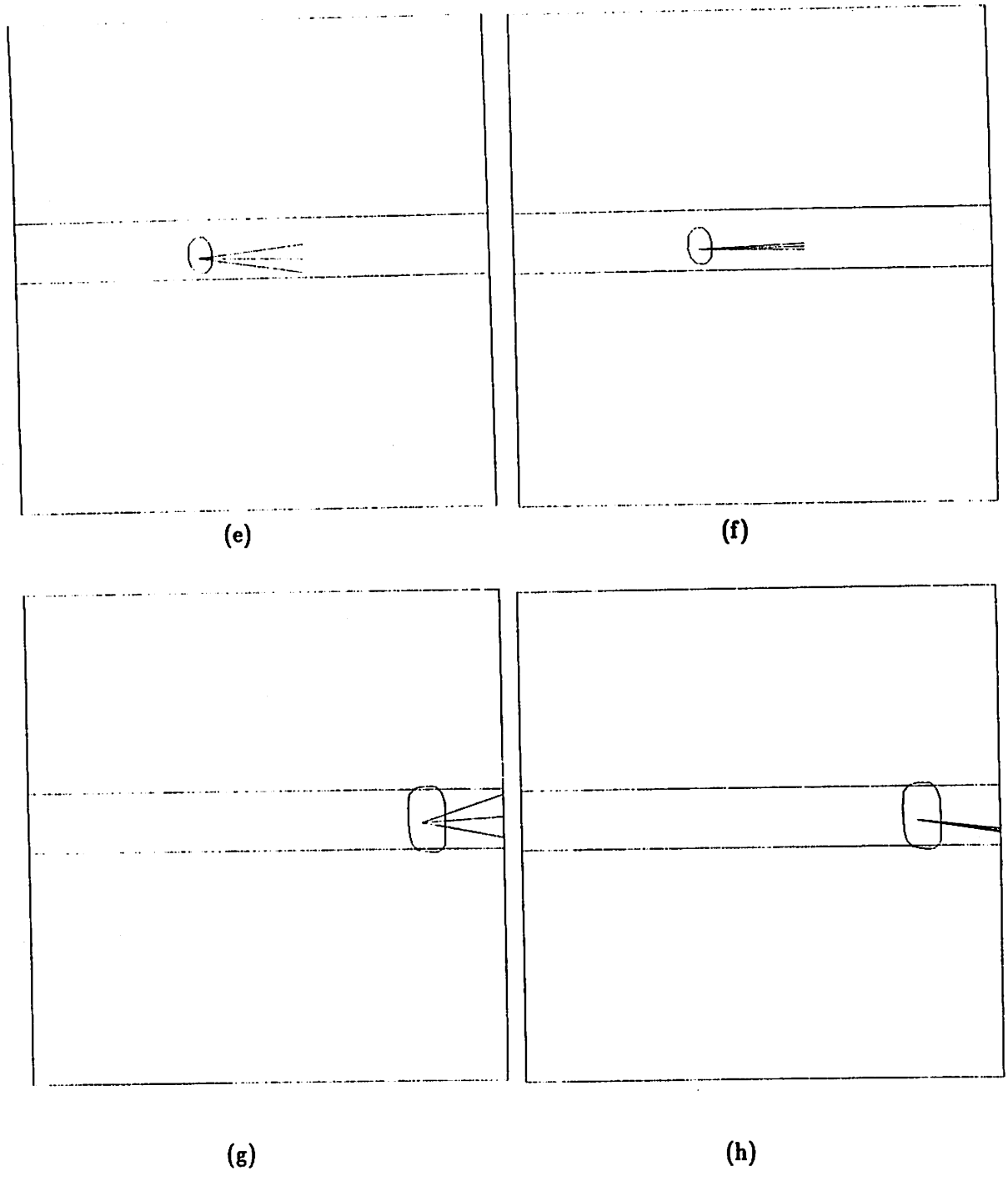


Figure 103 continued.

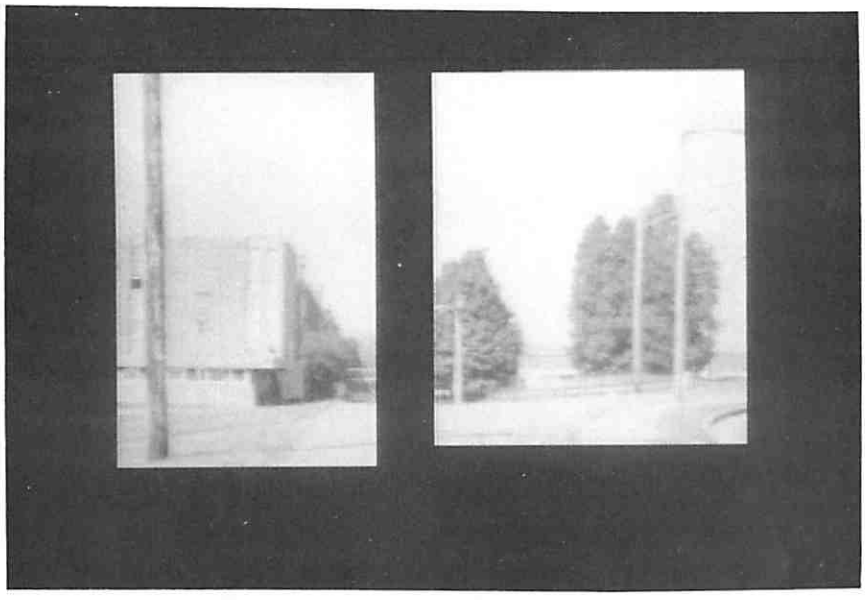
the pitch of the vehicle (due to the crowning of the sidewalk surface) still distorted the landmark's position in the image. By this time, without any inclinometer data or a stabilized platform, we had lost confidence in the ability to directly extract the depth of this feature. The roll was minimal (by observation) so we used this feature as a Class I landmark for the rest of the experiment.

Expectation windows were produced and used to restrict the search for the particular landmark being sought. If no pruning was accomplished the expectation windows would be larger than if pruning was undertaken. Figure 104 shows the size of the expectation windows for three landmarks when pruning is performed. The production of these windows involved the use of the camera calibration matrix described in Chapter 6. Although the line finder was run on these windows (Fig. 105) and produced identifiable structures related to the landmarks, the actual image position of the landmark was determined manually. After identification occurred, the landmark image coordinates were input along with its world coordinates to automatically reduce the spatial uncertainty. Only orientation uncertainty reduction was in evidence as the uncertainty in the world coordinates of the landmark and the error in the calibration matrix permitted only coarse adjustments. By *surveying* the three-dimensional world accurately, and using inclinometers or inertial navigation to give an estimate of the roll, pitch, and yaw of the camera relative to the world, greater control of the uncertainty could be produced.

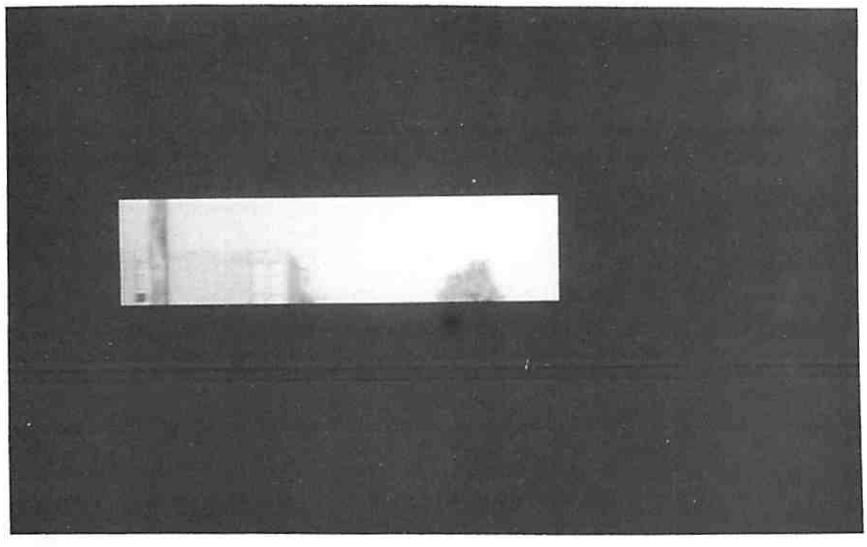
§5. Multi-component test

The multi-component test combines motor schema based navigation, ultrasonic obstacle avoidance, path following using vision, and STM building. The navigation was performed on-line, while the short-term memory representation was generated off-line. The STM context is shown for both levels: the representation for the instantiated meadows (Fig. 112) and the sonar-filled regular grid (Fig. 110).

This test initially involved navigation using the fast-line finder, with the robot moving towards the center of the walk. Figure 106 shows the robot's course. While performing line finding, the robot monitored its heading for obstacles using ultrasound. When a blockage was detected as evidenced by at least one of the three leading ultrasonic sensors returning a value below a specified threshold (3.5 ft), the robot switched over automat-



(a)



(b)

Figure 104: Localization experiment expectation windows
These are the windows produced for the image taken at position 3 by the spatial uncertainty map incorporating the reductions in orientation uncertainty as the robot moves.
a) Window for telephone pole and the concrete lamppost.
b) Window for building corner.

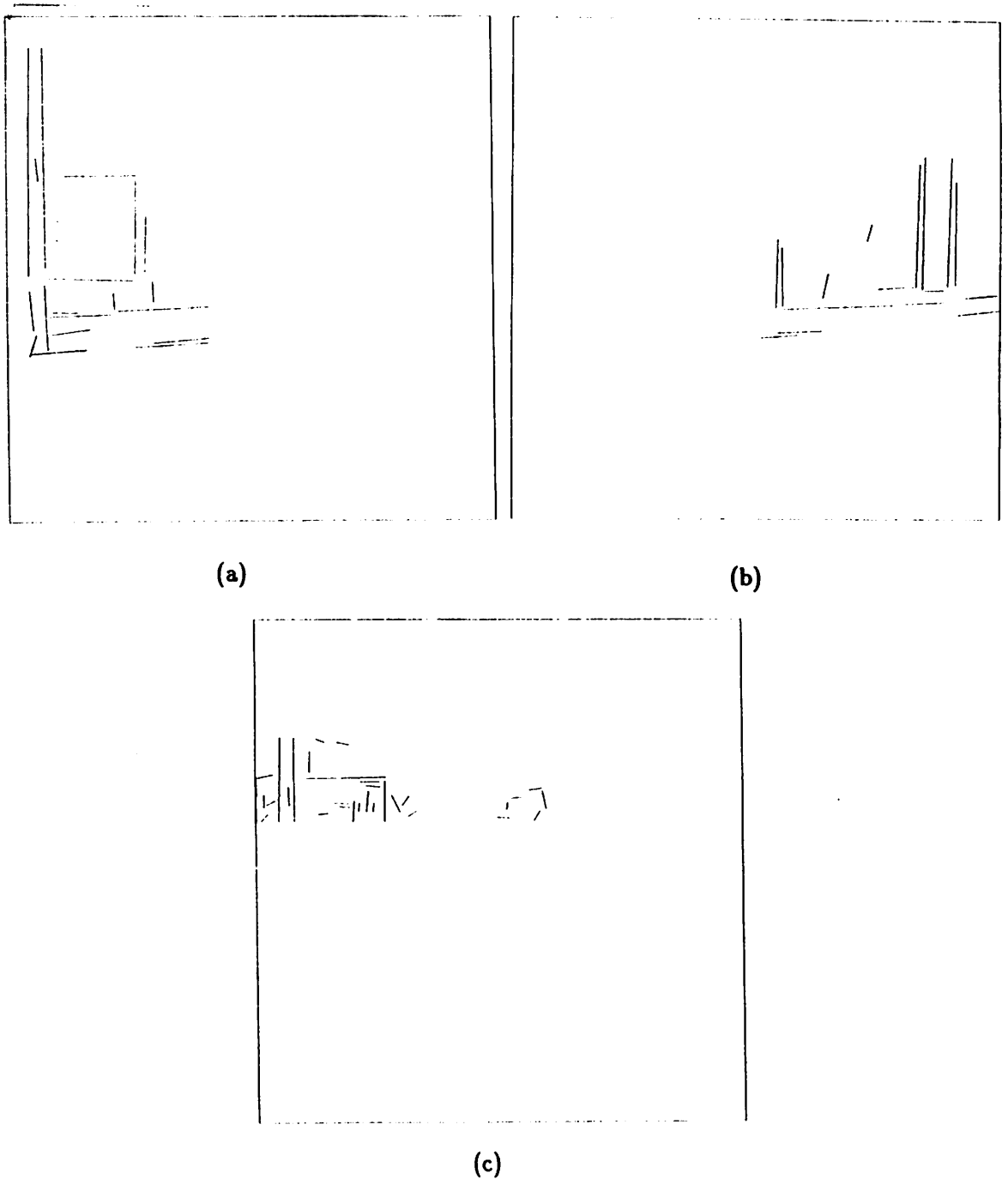


Figure 105: Line finder results on landmark windows
These are the results for the image taken from position 3.
a) Results for telephone pole.
b) Results for concrete lamppost.
c) Results for building corner.

ically to schema-based navigation, using **move-ahead** (encoder-based), **stay-on-path** (encoder-based), **avoid-static-obstacle** (sonar-based), and **noise** SIs in the framework of the schema-based navigational testbed (described in Section 1) to negotiate the course. The obstacles were successfully circumnavigated, at which time the robot reverted to line-finding navigation. The run went extremely well, with the exception of the robot's desire to strangle itself on its camera tether. The cable tether, due to the frequent rotations during the obstacle avoidance phase, became entangled around the robot's upper (rotating) platform and the lower (stationary) body. When the UHF transmitter becomes operational this will no longer pose a problem. Aside from this difficulty, the run went smoothly.

Figures 107-109 show the fast line finding component in action. It is interesting to note that the line being extracted on the right side is actually a change in the surface type of concrete rather than a separate path. This surface feature is distinctive enough for the line finder to operate on.

Figure 110 shows the short-term memory representation built up from the sonar data. The robot built this model as it passed amongst the middle of the cone area. The scale of this grid is 32 feet by 32 feet. Whitespace denotes areas highly probably to be unoccupied while occupied area probability is denoted by numeric values. This information was not needed for this run as the motor schema based navigation system handled the obstacles without any difficulty. As mentioned in Chapters 4 and 5, the reliance on STM should be relatively rare and is only needed when the potential field methodology fails.

Figure 111 shows the planned course through LTM. The information in this map is used by the STM meadow instantiator process to extract only those meadows relevant for this particular navigational leg. Figure 112 shows the instantiated meadow component of STM for this run.

§6. Summary

HARV's experiments have given a practical demonstration of many of the concepts developed in earlier chapters. As expected, some experiments were more successful than others. Good results were obtained with motor schema-based navigation, the path following using the fast line finder and the multi-component experiment. The fast region

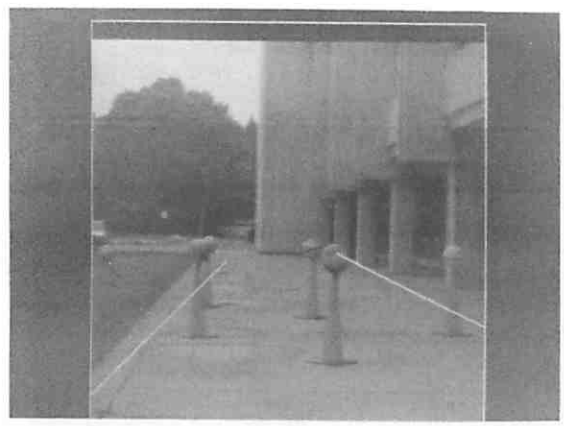
Figure 106: Multi-component test run

The robot initially moves under control of the FLF path following algorithm. When it detects the obstacle in frame 3 using ultrasound, it switches to schema-based navigation, using the shaft encoders to enforce the stay-on-path schema. After successful completion of the obstacle course, it reverts to FLF path following.



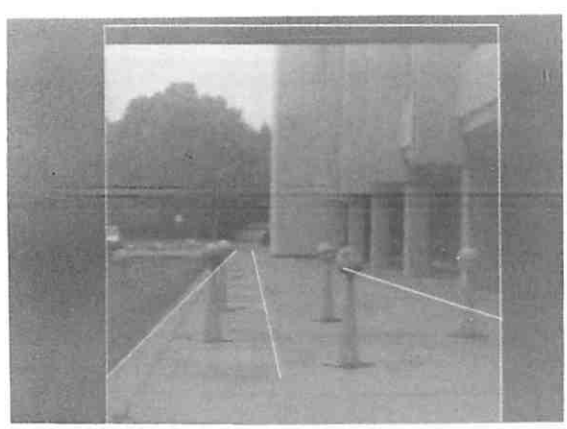




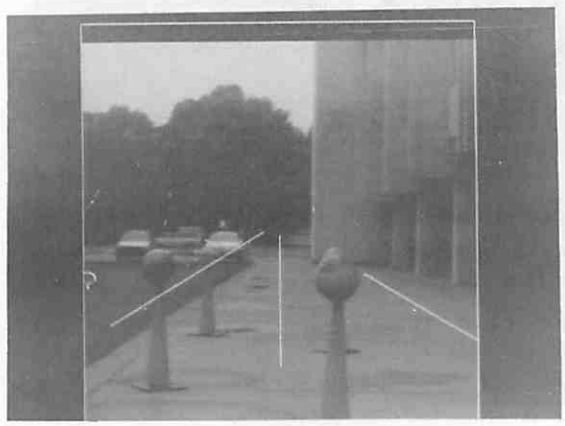
(a)

(b)



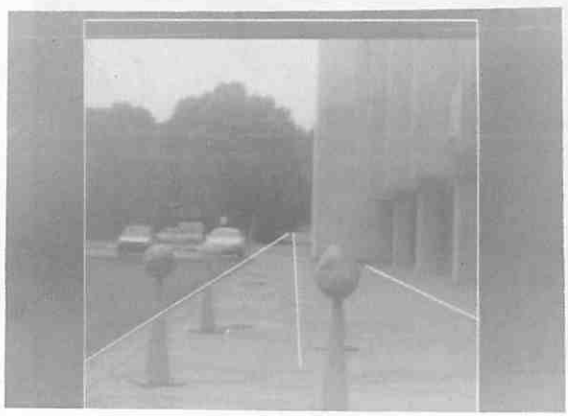
(c)

Figure 107: Multi-component fast line-finding (first image)
a) The bootstrap lines used to tune the buckets for the image. Note there is a significant discrepancy between the expectations and the actual image, but the lines are found anyway, indicating the robustness of this approach.
b) Extracted lines using orientation tuned buckets.
c) Resulting road edges and computed centerline for servoing.



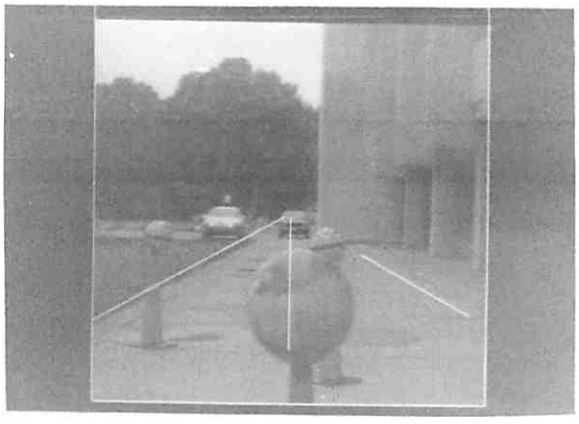
(a)

(b)

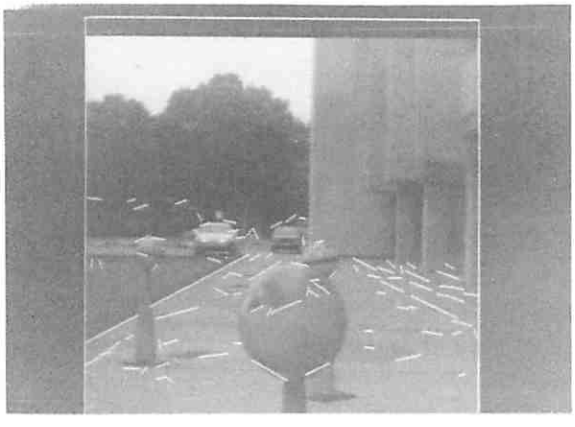


(c)

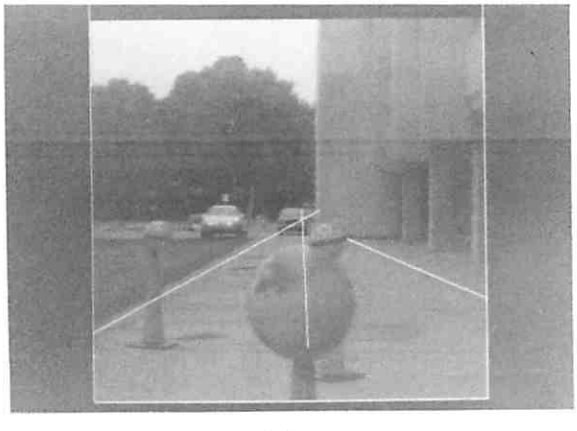
Figure 108: Multi-component fast line-finding (second image)
a) The feedforward lines produced from the first image that are used to tune the buckets for the image. Note again the discrepancy between the expectations and the actual image events.
b) Extracted lines using orientation tuned buckets.
c) Resulting road edges and computed centerline for servoing.



(a)



(b)



(c)

Figure 109: Multi-component fast line-finding (third image)
a) The feedforward lines used to tune the buckets for the image.
b) Extracted lines using orientation tuned buckets.
c) Resulting road edges and computed centerline for servoing.

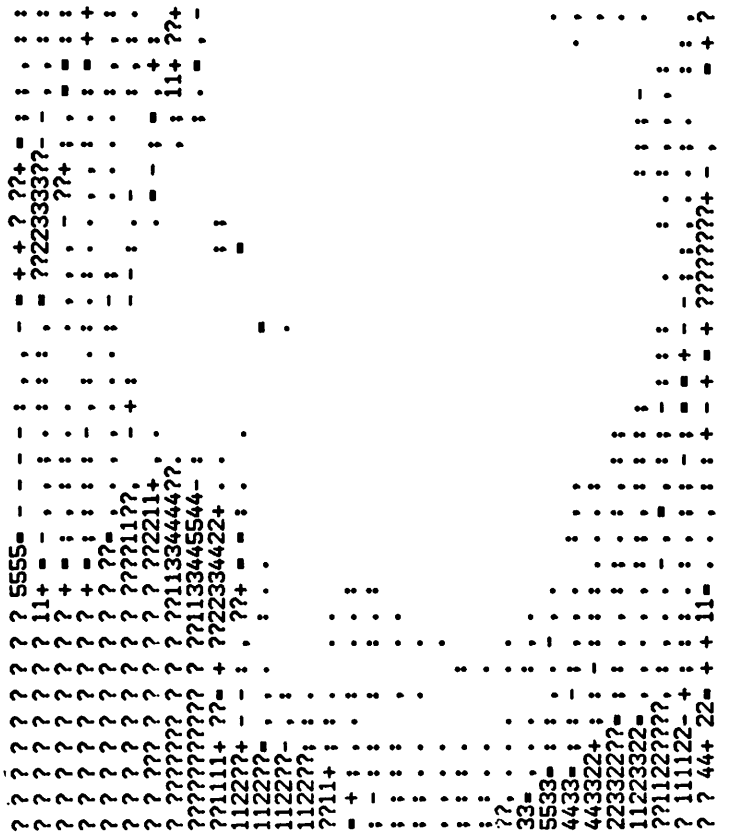


Figure 110: Multi-component STM building

A modified version of the Moravec-Elves code [93] is used to construct the top-level grid representation for STM (Chapter 4). Here the robot is in the free space amidst the obstacles (the view is from above). The clusters of numbers represent positions likely to be occupied by an obstacle (i.e. cone) while whitespace denotes areas likely to be unoccupied. (The scale of the grid is 32 feet by 32 feet).

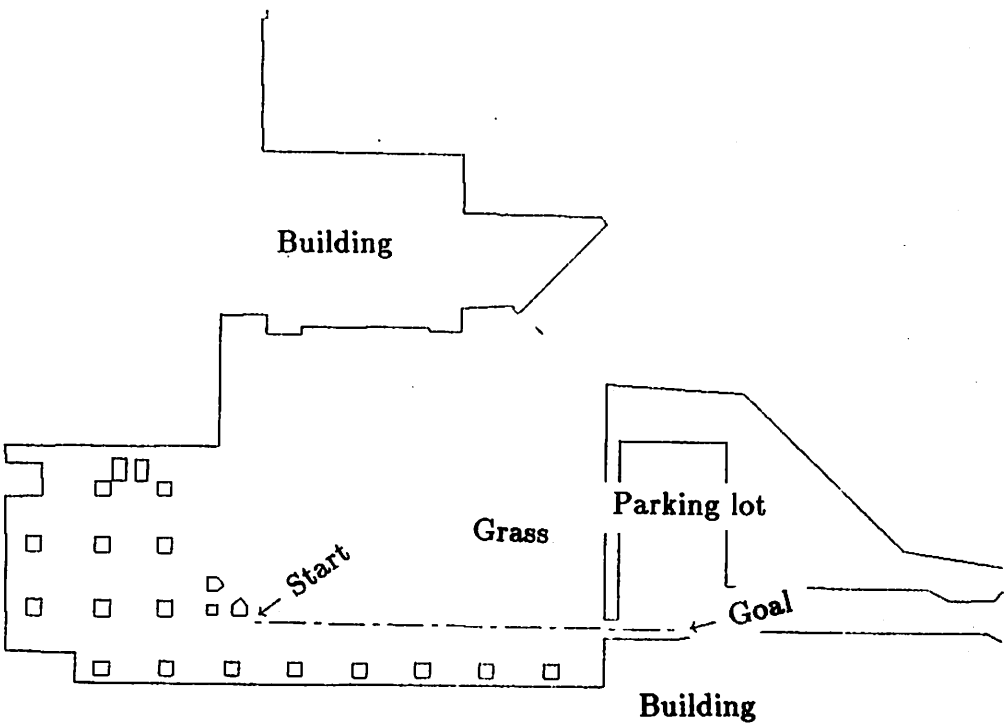


Figure 111: Multi-component LTM path

The robot's path takes it along the sidewalk past the GRC's concrete pillars. Note the unmodeled obstacles are *not* present during the path planning process.

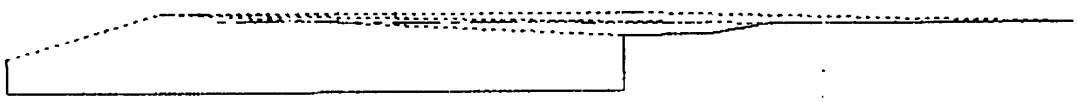


Figure 112: Multi-component STM instantiated meadows
These are the meadows selected by the cartographer to provide the context for STM (base level) during the robot's journey.

segmenter and depth-from-motion system produced less than perfect results. In all cases, the experiments yielded valuable insights into the problems of navigation in relatively unconstrained environments.

Schema-based navigation using ultrasound showed a high degree of reliability. Combinations of the primitive motor behaviors manifested the predicted more complex behaviors. When the vision algorithms become faster, it would be desirable to embed them as well as perceptual algorithms in this experimental testbed, especially for goal recognition and path following.

The vision algorithms were tested modularly and in one instance, in the multi-component test, combined with other navigational components of AuRA. The fast line finding system behaved particularly well for path following in both indoor and outdoor situations under a variety of environmental conditions. This algorithm will continue to be tested under more difficult conditions such as roads and paths streaked with shadows.

The fast region segmentation path following system is less robust, but this is expected. Using gradient magnitude for outdoor scenes only using intensity images limits severely what can be accomplished. Nonetheless, successful path following was accomplished using this algorithm. Color image acquisition is essential, in our estimation, to improve the reliability of this algorithm when applied to path finding.

The depth-from-motion algorithm was a mixed bag, yielding both successes and disappointments. The successes arose from the ability to track points reliably from frame-to-frame in a very high percentage of the cases. The disappointments came from the inability to produce highly accurate numerical depths. The reasons for this are believed to be well understood: misregistration and inaccurate focus-of-expansion recovery. These problems, although difficult, are not considered insurmountable and continuing research is being applied to the depth-from-motion system to obtain greater accuracy.

The role of spatial uncertainty management in AuRA was illustrated through the localization experiment. The Expecter, using camera calibration data, is used to predict where in an image a particular landmark will occur. Although this experiment used manual landmark discovery, the results of the fast line finder on these windows showed automation of this process is possible. Ultimately, landmark discovery will be relegated to the VISIONS system.

An interesting problem, especially for point landmarks (Class II - see Chapter 7),

occurs when the ground-plane assumption fails. This occurred in two ways. First, during the initial knowledge acquisition phase, it was insufficient to know the height of the building or other landmark feature above the ground. What was required was the height of the ground feature above the ground-plane the robot was currently located on. The LTM landmark representation must incorporate height relative to some global plane (e.g. sea level) if the robot is not working on a truly planar area. Secondly, it is important that the robot's pitch and roll relative to the ground plane be known. Inclinometers, currently unavailable on HARV, can provide this information. If the pitch or roll is non-zero relative to the ground plane, the vehicle is actually on a different plane relative to the LTM ground plane, and compensation must be made in the predictions issued by the Expecter as to where the landmark will appear in an incoming image. If the robot is pitching upwards towards the front, a landmark will appear lower in the image than otherwise expected. To further worsen matters, when the recognized landmarks's position is backprojected into LTM it will then appear farther away than it actually is producing errors in the spatial uncertainty map. As sidewalks and roads are typically crowned for drainage reasons this can pose real difficulties in obtaining accurate results.

The multi-component test demonstrates the relationships between many of the individual components of the AuRA architecture, including LTM, STM, navigational planning and schema-based navigation. This experiment went quite smoothly with all the different components integrating quite well. Future enhancements would involve the conversion of the visual FLF path following component to a schema and its integration into the motor schema testbed. This would avoid the somewhat artificial changeover from visual path following to ultrasonic and encoder based navigation and then back to vision again.

Many lessons were learned from these experiments. One of the most important is the value of performing the experiments themselves and not relying too heavily on simulations. In some cases the move from simulation to experiment was smooth as in the schema-based navigation system. In others, such as the depth-from-motion system, whose simulations worked well on synthetic images [21], many types of unanticipated difficulties appeared, such as misregistration and FOE changes. All in all, however, considerable progress has been made in having a mobile robot explore both indoor and outdoor environments given a partial model of the world.

CHAPTER IX

SUMMARY AND CONCLUSIONS

§1. Summary of Work

This dissertation has attempted to accomplish many things in autonomous robot vehicle navigation. These include:

- Global planning in the context of *a priori* knowledge
- Reactive/reflexive navigation in response to sensor information
- Perceptual strategies to provide navigational guidance
- Spatial uncertainty management
- A framework to tie these components together

This chapter first reviews the work performed for each of the above items. Future research areas are then described. A discussion of the general contributions resulting from the work in this dissertation is then presented. A brief commentary on what the future holds for mobile robotics concludes the chapter.

§1.1 *Global planning*

Global planning is accomplished through the use of digitized maps or blueprints, embedded with additional knowledge regarding terrain type, landmarks, etc. A convex decomposition algorithm operates on the original map to produce a meadow map, a connectivity graph tying together the individual convex regions. Two variations of an A* search algorithm are available to search for a "reasonable" global path through the modeled world. Chapter 4 describes the algorithms, rationale, and many advantages of

the meadow map representation. These include: the flexibility to produce short, safe, and other types of paths; the ease of embedding information to guide perceptual processing, the reasonable storage requirements; and the ability to conduct multi-terrain navigation in a straightforward manner.

The initial coarse path is improved through the use of strategies which tighten and straighten the unrefined path. Both this and the extension of this method to incorporate multiple terrain types are significant advances over previous work in this domain. The navigator's path planning techniques, both for single and multi-terrain navigation are described in detail in Chapter 4. The A* search strategies provide rapid initial path determination. The path strategies developed, for both tightening and straightening the raw path, produce high quality paths which previous methods, using similar representations, were unable to accomplish.

The global path planning component of the AuRA architecture is complete. The navigator, mapbuilder, feature editor, and long-term memory (LTM) component are fully implemented. The mission planner exists in a rudimentary version and is an area that holds great promise for future research (see below).

Good sources for navigational knowledge include building blueprints, aerial maps, and the various mapping agencies (e.g. Defense Mapping Agency). The data used to construct the landmark models for guiding perceptual processing can come from engineering drawings (e.g. for lampposts or fire hydrants). Spectral data pertinent to landmarks can be dynamically acquired from histogram processes running on images of the landmarks. In general, however, knowledge acquisition is a difficult and laborious task. Much additional research is needed to make this work easier (see Section 2).

Even if portions of the data within the *a priori* knowledge base are incorrect or inaccurate, in many instances the system performance will still be robust. Actually, navigation with little if any *a priori* knowledge is quite feasible, simply by having a means of goal recognition, a rough idea of the goal's location, and avoiding obstacles as the robot moves. In this manner, all obstacles are treated as unmodeled and high-level path planning is circumvented. Localization is of no concern as there is no world map to correlate against and schema-based navigation is the fundamental navigational principle. Efficiency, reliability, and safety, however, are sacrificed somewhat when the *a priori* knowledge is absent.

§1.2 *Reactive/reflexive navigation*

The true role of the global planner is to provide a context in which the pilot can select appropriate motor behaviors (schemas) to satisfy the navigational subgoals. These motor behaviors are exemplified by **move-ahead, move-to-goal, avoid-obstacle, stay-on-path** and **follow-the-leader** schemas. Each of these motor schemas is associated with one or more perceptual schemas/strategies that provide the sensor data essential for the successful completion of the particular motor task. Each individual motor schema, ideally operating in a concurrent manner, contributes a velocity vector germane to its particular motor task. These are then summed by the **move-robot** motor schema and transmitted to the motor control subsystem to update the robot's current direction and speed. Chapter 5 describes the motivation for schema usage, the potential field methodology used, and simulation results. Chapter 8 demonstrates, via experimentation, the utility of motor schema based navigation.

This control regime maps well to a distributed processing system and affords a high level of modularity that simplifies the addition and debugging of new perceptual or motor schemas. Complex behaviors are constructed through the combination of multiple simple behaviors. When suitable tools become available on the Sequent multiprocessor (i.e. the Schema Shell [41,42]) an implementation of the motor schema manager should follow. The experimental system described in Chapter 8 uses sequential evaluation of motor schemas to emulate concurrent schema activity.

Summing the output of the individual velocity vectors produced by each schema instantiation (SI) is a simplistic approach. The functions producing the velocity vectors themselves, in their linear implementation, are also simple. Nonetheless, good results for navigation have been obtained using these methods, both in simulation (Chapter 5) and in experimentation (Chapter 8). It is advisable to look at other methods for combining the outputs of the SIs and to consider other possibilities for the velocity formulations of the SIs themselves, to study just how much can be gained through the use of more complex functions and combination mechanisms.

The potential field methodology is not failsafe and reliance on a sensor-built world model constructed by the cartographer is important for recovery in instances of local potential minima or cyclic behavior. This intermediate planning level is invoked by the pilot when anomalous behavior is detected, typically by exceeding a hard real-time

deadline for path completion (cycle detection) or recognition that the robot's velocity has dropped to unacceptably low levels (local minima). Some of the navigational problems can be circumvented by the addition of small amounts of noise to the navigational process, moving the robot away from potential plateaus. Chapter 4 describes the roles of the pilot for both schema selection and local navigation. A simulation illustrating this point is presented in Chapter 5.

§1.3 *Perceptual strategies/schemas*

Action-oriented perception is the fundamental premise on which AuRA's perceptual strategies are based. By designing specific perceptual algorithms to solve particular navigational tasks, the amounts of computation required for intelligent navigation becomes tractable.

Vision is viewed within AuRA as a primary sensor modality. Chapter 6 details the different vision algorithms used to provide HARV with navigational capabilities. Unfortunately, the processing speeds for sophisticated algorithms on existing hardware currently preclude its use for real-time navigation. Nonetheless, experiments were conducted (Chapter 8), typically in lurch mode with the vehicle pausing during image processing, showing the ability of the robot to follow paths and to avoid obstacles using vision.

A fast-line finding algorithm was developed for both path edge extraction and vehicle localization purposes. The line-finder can be tuned to anticipated line orientation based on the availability of *a priori* knowledge or the results of previous frames. The use of a feedforward strategy to track features over time is typical of the vision algorithms.

As in all the vision work, this algorithm had to be accompanied by large amounts of software to accomplish the navigational tasks. This included:

- Grouping strategies to interpret the results of the image processing in the context of the navigational need (e.g. agglomeration of the line fragments to produce the path edges).
- Utilization of both *a priori* knowledge and data obtained from previous frames to reduce computational demand. (e.g. windowing the image).
- Interpretation of the output of the algorithm to produce intermediate results that reflect the navigational goals (e.g. determination of center road line).

- Conversion of the interpreted image into suitable motor commands (i.e. visual servoing).
- Communication of those commands to the vehicle.
- Production of new image expectations based upon the motor commands given the vehicle (e.g. where will the path appear in the next image now that the robot has moved).

A fast region segmentation algorithm (Chapter 6) is also available and is used to provide information for path following. Through knowledge of the spectral characteristics of the path or landmark in question, these features can be successfully extracted. The use of color will greatly enhance the ability of this algorithm. Chapter 8 describes an experimental run using this technique.

The depth-from-motion algorithm described in Chapter 6 provides the location of obstacles in the robot's environment. These detected obstacles can then be instantiated as **avoid-obstacle** motor schemas and subsequently conduct the robot's navigation. Chapter 8 presents experiments in both indoor and outdoor environments. This particular approach, unlike fast line finding and fast region segmentation, is particularly susceptible to problems in image registration. When this problem and others (such as the ability to extract the focus of expansion reliably and accurately) are solved, considerably better results are expected. Nonetheless, the ability of the algorithm to track feature points over multiple frames is indicative of the significant potential for this approach to obstacle avoidance.

Ultrasonic data is the most efficient means for demonstrating schema-based navigational abilities. Unfortunately, due to the limited resolution and discriminatory power of ultrasonic sensors, semantic interpretation is limited. Nonetheless, recognition of obstacles and tracking behavior are both readily implemented using this sensor form. Chapter 6 discusses the use of ultrasound in AuRA while Chapter 8 describes the experimental results obtained in the motor schema testbed used with HARV. Obstacle avoidance, wandering, impatient waiting, follow-the-leader, and other interesting behaviors have been demonstrated using this sensor modality.

Shaft encoders provide general directional ability as well as approximations of distance traveled. Although it would be a mistake to rely too heavily on the encoders due to

relatively large errors that arise during path execution, they can still provide, in most cases, a reasonable coarse estimate of the robot's whereabouts. This is especially useful in the context of the uncertainty management subsystem described in chapter 7 and as applied in the localization experiment in Chapter 8. Chapter 6 discusses some of the general issues in using HARV's shaft encoders.

§1.4 *Spatial uncertainty management*

In order to provide expectations for the robot's perceptions and to make the computational requirements for perceptual processing more tractable, a means of managing both positional and orientational uncertainty was developed. The spatial uncertainty map consists of two components – the first a convex polygon representing the extent of position uncertainty of the robot relative to the global map, and the second a compass wedge explicitly representing the orientation uncertainty. Growth in uncertainty occurs when the robot moves, based on statistical data reflecting the terrain type being traversed. Reduction in uncertainty occurs through the recognition of landmarks and the subsequent pruning of the spatial uncertainty map in a manner appropriate to the particular landmark. The spatial uncertainty map itself provides the basis for the Expecter process to provide image windows for landmark recognition by active perceptual schemas.

The uncertainty map manager, responsible for both the growth and reduction of the spatial uncertainty map, has been implemented in LISP. The ties connecting the LTM structure (coded in C) have been completed. Chapter 7 presents the spatial uncertainty management approach used within AuRA and Chapter 8 includes an experiment illustrating how vehicle localization is accomplished with this software.

Additional areas of uncertainty management, outside of the area of spatial uncertainty, undoubtedly will need to be addressed as the AuRA architecture grows. The treatment of symbolic uncertainty at the mission planner level, in particular, must be dealt with.

§1.5 *The AuRA framework*

AuRA, the Autonomous Robot Architecture, is the structure in which these individual components are linked. Chapter 3 presents an overview of the AuRA architecture and includes a discussion of the system issues involved. The overall architecture is quite ambitious, ultimately incorporating the UMASS VISIONS scene interpretation system

running on the UMASS Image Understanding Architecture. The first pass implementation of AuRA is much more modest, tying the available components together through blackboard-like global data structures (clipboards) or through the use of other expedient interim techniques. The AuRA's system architecture is not complete at this time, containing only a rudimentary mission planner, limited environmental knowledge embedded in LTM, a partial set of pilot rules for schema selection, some limitations on the current implementation of the uncertainty management subsystem, and limitations in the interfaces between components. The scope of this project, however, extends several years beyond this dissertation as the AuRA architecture becomes more fully integrated.

§2. Future research

Mobile robotics is a science in its infancy. Much remains to be done in almost all areas. There are several aspects of future research that would be quite appropriate for embedding in AuRA. These are discussed below.

§2.1 *Additional sensors*

AuRA's modularity makes it straightforward to embed new sensors and sensor algorithms. The use of inclinometers for vehicle orientation in a non-planar world would assist in the difficult problem of image registration. A literal "hill-climbing" schema could be embedded if the robot had the ability to measure its pitch and roll.

A laser range finder is highly desirable, but currently prohibitively expensive. Future funding may enable its acquisition thus enhancing the vehicle's ability to detect obstacles.

Color vision would greatly expand the possibilities for the fast region segmentation algorithm. Much remains to be done to apply the use of color in the domain of action-oriented perception.

§2.2 *Extension to three dimensions*

Much of this architecture is generalizable to robots capable of maneuvering in three dimensions. This would include space and undersea vehicles. Reformulation of the potential field equations used by the motor schemas to three dimensions is straightforward. The same would hold for the uncertainty management subsystem except for possible problems

in accounting for vehicle drift in the presence of ocean currents or the like. The meadow map representation, although capable of being extended to three dimensions may pose some mathematical difficulties, but certainly bears investigation. Some preliminary work on these extensions to provide for navigation in three dimensions is presented in [15].

Another area for exploration is a 3D ground-based world using the meadows as topographical facets with representations of their slope for path planning purposes. This would result in a piecewise planar world as opposed to a single ground plane.

§2.3 *Learning*

Several different levels of learning can be addressed within AuRA. The movement of information stored in short-term memory to long-term memory is a form of learning, as the representation persists after the robot has moved out of the area. In order to do this effectively however, semantic interpretation of the objects in question must be undertaken to distinguish whether the object to be learned is dynamic in nature (hence will move and is thus inappropriate for embedding in LTM as it may not be present the next time the robot is in the area) or is static and worth retaining in LTM. This is not a simple task by any means. Continued investigation into scene analysis (e.g. the VISIONS natural scene interpretation system) to provide both semantic interpretation of previously unrecognized environmental objects as well as construction of world models to guide sensor processing will potentially "open the eyes" of the robot.

Another form of learning involves the assimilation of motor skills. Just as robots are currently taught using a teaching pendant in industrial environments, perhaps through the combination of perceptual learning (e.g. by a reward/punishment regimen) with trained motor behavior, the need for mathematical formulae to represent potential fields at the programming level would disappear.

Adaptation of the gains on the motor schema outputs is also an important area for the application of learning theory. In this manner, the robot can learn from its experience as to which schema gains are best for a particular task. For a task requiring a delicate balancing of schema outputs, such as door entry, the tuning of these outputs is important.

In one sense, learning already exists in several of the perception algorithms used within AuRA. By utilizing the knowledge obtained from previous frames to guide subsequent processing (e.g. path edge position), the robot has acquired a model of its environment.

This information, however, is not currently used to update long-term memory representations and is discarded at the end of the navigational task.

§2.4 *Homeostatic control*

The niche is already present for homeostatic control within the AuRA framework (Chapter 3). Certainly the need exists for robots capable of managing their own internal resources, if they are to operate in environments that are too hazardous for man. By equipping the vehicle with temperature and energy sensors, the motor schema approach can be extended (as described in chapter 3) to include signal schemas. These signal schemas would modify the dynamic motor behavior of the robot based on continuous internal sensing. Instead of using worst-case analysis in the formulation of global plans, dynamic replanning would occur based on available information. This would result not only in safer robots, but also more efficient robotic systems.

§3. General contributions

The general contributions for this work include:

- **Extensions of techniques for representing world knowledge to include multi-terrain representations.**

  Previous approaches, typically using a regular grid representation, suffered from high memory requirements, large search times, and digitization bias, all of which are avoided with the meadow map approach (Chapter 4).

- **The ability to conduct path planning in diverse terrain types in an efficient manner.**

  By only resorting to path relaxation at terrain boundaries and by using the path improvement strategies mentioned below, quality paths can be obtained at low computational costs (Chapter 4).

- **The invention of path improvement techniques that refine initially coarse and unacceptable paths into "reasonable" paths.**

By adding tightening and straightening techniques to remove unnecessary twists and turns in the raw global path, significant improvement over previous path planning work using meadow maps is evidenced. These improvement strategies enhance the navigator's ability to produce safe or short paths depending on the mission planner's goals (Chapter 4).

- **The embedding of knowledge to guide uncertainty management and perceptual processing into the long-term memory representation.**

  Landmarks, terrain statistical data, spectral characteristics, etc., can be tied to the LTM representation through the use of the feature editor. The hybrid free-space vertex-graph representation (meadow map) saves information on both the obstacles and free space, unlike several other representation strategies (e.g. pure vertex graph and Voronoi diagrams), making the addition of further knowledge much easier (Chapter 4).

- **Decomposition of reactive/reflexive behaviors into a set of simple motor schemas which when combined react intelligently to sensory events.**

  By having individual motor schemas produce outputs that are relevant to the *overall* system goal, intelligent navigation can be achieved. The pilot selects these schemas based upon the current navigational subgoal, and parameterizes them to fit the task at hand (Chapter 5).

- **Making path execution amenable to distributed processing by allowing multiple concurrent schema instantiations operating asynchronously to produce the motor control for the robot.**

  The individual schema instantiations can run on separate processors each posting their output on a blackboard which is monitored by the move-robot SI. The combined results are then forwarded to the motor subsystem for execution (Chapter 5).

- **Applying potential field methodology to sensory-based navigation and incorporating the uncertainty in perception into the resultant velocity fields.**

  Representing a motor goal by a velocity field is an important means for accomplishing a variety of tasks. Fields have been specified for obstacle avoidance, goal attraction, path following, etc. The computation to produce the resultant vector is quite simple and can be performed rapidly (Chapter 5). Experimental results in Chapter 8 bear out these conclusions.

- **The embedding of action-oriented perceptual strategies specific to the task within the motor schema itself.**

  Action-oriented perception can only be accomplished by associating perceptual techniques with specific motor skills. A further advantage is the added capability to use multiprocessing for perception due to the natural partitioning of perceptual tasks based upon specific motor needs (Chapters 5 and 6).

- **The application of new visual perception techniques to the navigation problem (i.e. path finding, obstacle avoidance and localization) including fast line finding, fast region segmentation and a depth-from-motion algorithm.**

  Experimental results using these techniques are presented in Chapter 8. Some of the algorithms are quite robust (e.g. fast line finding) while others are more sensitive to environmental conditions such as camera roll (e.g. depth-from-motion). Much has been learned from both the successes and difficulties encountered by the application of these algorithms to navigation (Chapter 6).

- **A new technique for the management of spatial uncertainty.**

  By explicitly representing the degree of location and orientation uncertainty relative to the *a priori* world map information, the ability to establish expectations for perceptual processing is gained. The use of specialized uncertainty growth techniques based on empirical terrain data and uncertainty reduction techniques based on landmark recognition provide the feedback necessary for uncertainty management (Chapter 7).

- **The AuRA framework itself which ties together, in a cybernetic context, the individual components of the architecture.**

  Schema-based navigation and action-oriented perception are the principal tenets on which AuRA is founded (Chapters 3, 5, and 6).

§4. Directions

The field of mobile robotics foretells a bright and promising future. As inroads are made into this artificial intelligence problem, more and more applications will arise. Already there exist sentry robots, industrial cleaning robots, and mobile robots that operate in the manufacturing domain. The military is strongly supporting research into autonomous vehicles for battlefield, resupply, and reconnaissance operations.

There seems to be some innate fascination in people regarding autonomous machines. Even though Hollywood has come to treat autonomous robots as a solved problem (e.g. R2D2), the mystique continues. What perhaps is most important to the future of this field is that researchers do not resort to a series of contrived demonstrations to prove that their robots work. We, as artificial intelligence researchers, must learn from the mistakes that our predecessors in the 1950s and 60s made, especially in promising too much too soon.

The key, in my estimation, is to make steady progress through fundamental research, both in perception and motor control. Too much emphasis on applications will detract from the real issues involved. Fortunately, there exist excellent mobile systems (animals), provided by God, that can be studied to yield insights into the mobility problem. The biological successes that already are in place should not be ignored by the roboticist.

In conclusion, if our expectations are realistic, the support is consistent, and work in other domains is drawn upon, a new era in intelligent machines awaits.

APPENDIX A

GLOBAL PATH PLANNING COMPUTATIONAL COSTS

This appendix provides some typical timings for the cartographic and navigator components of the AuRA architecture. All CPU times are from a moderately loaded VAX-11 750 (8 MB memory and floating point hardware). It is estimated that the times cut be cut by 67% to 75% by using a 68000 based workstation (e.g. SUN).

§1. Mapbuilder

Recognizing that the mapbuilder code is far from optimized, still some basic interpretations of the performance of the underlying algorithms can be made. Each component of the mapbuilding algorithm will be discussed in turn. The results of this section are based on a total of 72 different decompositions on 8 maps.

§1.1 *Border Growing*

The border growing algorithm is dependent on the number of vertices to be grown. The time required to grow the border for figure 16 was approximately 5.2 CPU seconds (28 vertices). A larger border (57 vertices) took approximately 22 seconds.

§1.2 *Obstacle attachment*

The obstacle attachment component of the mapbuilding algorithm is currently the least efficient part of the code. Massive improvements in CPU speed can be made. This phase of the mapbuilding is the most time consuming and appears to be exponential in order. A significantly more efficient algorithm for this phase could and should be developed if this system was to be used for other than experimental testing. The total

time to grow and attach 1 typical obstacle is 0.05 sec. This increases to 28 seconds for 7 obstacles and is dependent on the number of vertices of the border, number of obstacles, and number of obstacle vertices. Efficiency is currently not a prime factor during the long term memory mapbuilding phase as it is only compiled once at the start of the run. Nonetheless, for more complex environments this part of the code should be cleaned up.

§1.3 *Decomposition Times*

Perhaps of more interest is the time required for the recursive decomposition of the single region output by the obstacle attachment algorithm. Surprisingly rapid results were obtained. For Figure 18, computational times from 3.2 to 12.2 CPU seconds were observed (including merge). The general trends were as expected; choosing the first concave vertex consistently outperformed the least or most concave modes (an example timing is provided in Table 6). Of greater significance was the choice of the victim vertex: the opposite vertex performed best, followed by leftmost then rightmost victim. In many cases choice of the most opposite vertex resulted in decomposition times about 1/4 of the time required for rightmost victims.

§2. Navigational path finder

Fifteen different paths using four different start and end point pairs and two different safety margins were computed using the decomposition shown in Figure 22a. Many of these paths are shown in Figures 26-32.

No claim is made for the efficiency of the implementation. Significant time savings could probably be achieved (especially in the path improvement section) if the code was rewritten. These should be considered relative figures, not those that indicate the lower bounds of the algorithm.

§2.1 *Search*

As would be expected the search time for the A*-3 algorithm is considerably higher than that for the A*-1 method. The total number of possible nodes to be expanded triples, while the path solution space is cubed. Although the A*-3 method is definitely

Table 6: Typical mapbuilding timings

Concave Mode	Victim Mode	Decompose time	Clean-up merge time	Total time	Number regions
Most	Opposite	2.06	1.10	39.3	29
Most	Leftmost	3.08	3.95	43.1	38
Most	Rightmost	6.15	6.06	48.3	35
Least	Opposite	2.56	0.95	39.6	37
Least	Leftmost	3.50	5.06	44.7	37
Least	Rightmost	4.00	5.35	45.5	38
First	Opposite	2.00	1.86	40.0	31
First	Leftmost	2.76	5.20	44.1	38
First	Rightmost	3.55	6.88	46.5	34

These timings are for the decomposition of the map shown in Figure 18.

The time required to grow the border region (configuration space) is 5.2 seconds.

The time required to grow and attach the obstacles to the border is 30.9 seconds.

more costly, it is not prohibitive as the data below confirms.

Range for A*-1 (search) : 0.46-1.38 sec (avg: 0.95 sec)

Range for A*-3 (search) : 1.51-6.15 sec (avg: 4.07 sec)

Percentage Increase for A*-3 over A*-1 (search):

- Range: 188% - 781%
- Average: 455%

§2.2 *Path Improvement*

A*-3 will perform better than A*-1 in the path improvement component of the algorithm as it can completely bypass the initial tautness part. This results in significant time savings which offset some of the additional computation as described above.

Using the same paths as in the search component, these results follow:

Range for A*-1 (path imp.): 0.76-6.91 sec (avg: 3.18 sec)

Range for A*-3 (path imp.): 0.43-4.20 sec (avg: 1.46 sec)

Percentage of A*-3 over A*-1 (path improvement):

- Range: 12% - 83%
- Average: 58%

§2.3 *Total Path time (simple terrain - no relaxation)*

In all but one case the A*-1 algorithm outperformed the A*-3 method using CPU time as the indicator. The results are not too widely separated due to the shorter path improvement times required for A*-3. This shows the feasibility of this method in certain instances. It should also be noted that in every case (except one) the A*-3 algorithm came up with a path equal to or lower in cost to the one produced by the A*-1 algorithm. Only when high safety factors were involved did A*-1 derive a lower cost path. This is misleading as the goal was to produce safe, not short paths. If the straightening was turned off the A*-3 algorithm would have outperformed A*-1 as expected.

Range for A*-1 (total) : 1.22-7.59 sec (avg: 4.13 sec)

Range for A*-3 (total) : 2.14-8.66 sec (avg: 5.54 sec)

Percentage Increase per run for A*-3 over A*-1 (total):

- Range: 75% - 271%
- Average: 162%

On the average it took 62% longer to find a path using A*-3 than A*-1 when all path improvement techniques are in use.

§2.4 *Path cost savings*

Of the three paths produced by the A*-3 method that were less costly than those produced using the A*-1 method, the first was 8% less costly, the second was 7% less costly, and the third was 2% less costly. These are rather paltry cost savings for the mobile robot domain in which dynamic obstacles and/or uncertainty can render any precomputed path useless.

§2.5 *Multi-terrain transition zone relaxation*

Path relaxation figures were harder to collect and are highly dependent on the relaxation step size. The increment was set to 3.0 feet, about the diameter of the robot. Obviously, the number of transition zones to be relaxed also has a profound effect on the relaxation time.

For a single relaxation zone, (the longest concrete-grass border: see Figure 36), relaxation times observed ranged from a minimum of 0.40 CPU sec to a maximum of 4.13 sec. For two zones (grass across gravel path to other grassy section), times from 3.06 to 5.23 seconds resulted.

The relaxation for Figure 37 (two zones) took 0.38 seconds (this was overhead cost only as the path produced by the improvement strategies was already at a minimum cost). Figure 38 (2 zones) took 3.73 seconds and Figure 39 (3.5 zones) took 10.7 seconds. The longest observed relaxation time (4 zones) was 22.81 seconds. It is anticipated that most paths for the mobile robot will not involve numerous transition zone crossings.

APPENDIX B

ROBOT VEHICLE INTERFACE C LIBRARY

The routines listed below are used to establish communications, execute translation and orientation changes of the vehicle, control sensor data collection processes, and other miscellaneous functions. Fortunately, many of the functions existed within the DRV terminal emulation software and the major function for several of the routines below was to communicate the command from the host VAX to the robot. In other cases, the DRV software had no parallel. The Denning documentation set [38] and the UMASS DRV's Software Development Guide [12] serve as further references.

If AuRA is to be ported to another vehicles, these routines would need to be recoded to accommodate the new vehicle. As long as the vehicle's locomotion system is not that different from the DRV's, most of the other subsystems of AuRA can be considered insulated from a vehicle change and thus support the concept of robot vehicle independence.

§1. Robot Primitives

- Translation commands:
  - move(distance,velocity,acceleration)
    move a given distance then stop.
  - drive(velocity,acceleration)
    initiate translational motion.
  - wait_for_drive_to_stop()
    signal when drive motor has stopped.
  - stopdrive()
    terminate translation normally (smooth deceleration).

- killdrive()

  emergency stop.

- stop()

  stop both translation and rotation.

• Rotation commands:

- turn(angle,angular_velocity,angular_acceleration)

  turn a given angle then stop.

- steer(angular_velocity, angular_acceleration)

  initiate rotation.

- home(angular_velocity,angular_acceleration)

  turn wheels to known orientation relative to body.

- wait for steer to stop()

  signal when robot has stopped turning.

- stopsteer()

  terminate rotation normally.

- killsteer()

  emergency rotation stop.

• Sensor

- Ultrasonics:

  * range_start()

    start ultrasonic sensors firing.

  * range_stop()

    stop firing of ultrasonic sensors.

  * range_read(sonar data)

    acquire ultrasonic data.

- Shaft encoders:

  * initxy()

    set xy position coordinates to x = 0, y = 0, theta = 0.

* getxy(x,y,theta)

  acquire positional information from DRV.

* setxy(x,y,theta)

  set positional information from DRV.

* drivepos()

  acquire shaft encoder data from drive motor.

* steerpos()

  acquire shaft encoder data from steering motor.

* drivevel()

  acquire velocity of drive motor.

* steervel()

  acquire velocity of steering motor.

* drivestat(status)

  read the status of the drive motor controller.

* steerstat(status)

  read the status of the steering motor controller.

- Communications

  - open_robot_channel()

    Establish communications with the robot.

  - send robot message(channel,message,length)

    send a command to the robot.

  - flush_to(channel,character)

    remove garbage output from robot.

  - read robot character(channel,character)

    read a character from robot.

  - find_first_non(channel,character,buffer)

    strips data up to character.

  - close tty line(channel)

    terminate communications with robot.

- Miscellaneous
  - set_terminal_raw()

    configure the DRV's communication port.
  - set_parameter(name,value)

    set a DRV internal parameter to a specified value.
  - monitor_move(sensor,spread,limit)

    safe translation using ultrasonic data.

APPENDIX C

VISION ALGORITHMS

This appendix contains outlines of two of the vision algorithms referred to in chapter 6. References are cited where appropriate to direct the reader to more information on their details.

§1. Fast line finder

The outline of this algorithm, developed by Kahn, Kitchen and Riseman, is reproduced from [64]. The interested reader is referred to that paper for additional details. It is based on a previous algorithm by Burns, Hanson, and Riseman [31].

1.0 PASS 1 over the image

- 1.1 for each image pixel do
  - 1.1.1 Compute the gradient direction and magnitude
  - 1.1.2 Threshold on gradient magnitude
  - 1.1.3 Use gradient direction to classify suprathreshold pixels into buckets. (Pixels below threshold and pixels whose gradient is not in a direction of interest are specially labeled and ignored in all subsequent processing.)
  - 1.1.4 Perform connected components analysis to group adjacent pixels that share identical bucket labels into region fragments, and build a *fragment equivalence list*
- 1.2 for every unprocessed region fragment do
  - 1.2.1 Use the *fragment equivalence list* to merge fragments into line support regions and set the region pixel count to the sum of the fragment pixel counts
  - 1.2.2 Eliminate regions whose pixel count is below a certain threshold by tagging their constituent fragments "insignificant" so they will be ignored in subsequent processing
  - 1.2.3 Assign region labels to fragments which are part of these "significant" regions

2.0 PASS 2 over the fragment labeled image and gradient magnitude image

2.1 **for** each image pixel in a "significant" region **do**

2.1.1 For the region within which the pixel is contained, accumulate statistics needed to compute the scatter matrix and endpoints (for line fitting)

2.1.2 If desired, build a region labeled image

2.2 **for** each "significant" region **do**

2.2.1 Compute the scatter matrix from the accumulated statistics

2.2.2 Compute the best-fit line orientation and centroid anchor position

2.2.3 Compute the line endpoints

2.2.4 Put the fitted line and associated statistics into a line list to be output by the FLF program

§2. Fast Region Segmenter

This algorithm closely resembles the fast line finder on which it is based. See [64] for additional information.

1.0 PASS 1 over the image

1.1 **for** each image pixel **do**

Using a previously loaded look-up table and an arbitrary image, produce the bucket-labeled image based on gradient magnitude

1.1.2 Perform connected components analysis to group adjacent pixels that share identical bucket labels into regions and build a *fragment equivalence list*

1.2 **for** every unprocessed region fragment **do**

1.2.1 Use the *fragment equivalence list* to merge fragments into regions of similar intensity and set the region pixel count to the sum of the fragment pixel counts

1.2.2 Eliminate regions whose pixel count is below a certain threshold by tagging their constituent fragments "insignificant" so they will be ignored in subsequent processing

1.2.3 Assign region labels to fragments which are part of these "significant" regions

2.0 PASS 2 over the region labeled image and gradient magnitude image

2.1 **for** each image pixel in a "significant" region **do**

2.1.1 For the region within which the pixel is contained, accumulate region statistics.

•

2.1.2 Build a region labeled image

2.2 for each "significant" region do

2.2.1 Compute the scatter matrix from the accumulated statistics

2.2.4 Put the realm statistics into a list to be output by the FRS program

BIBLIOGRAPHY

[1] Amarel, S., "On Representations of Problems of Reasoning about Actions", *Machine Intelligence* 3, 1968, reprinted in *Readings in Artificial Intelligence*, ed. Webber & Nilsson, pp. 2 -22, Tioga, 1981.

[2] Anandan, P., "Measuring Visual Motion from Image Sequences", *Ph.D. Dissertation*, University of Massachusetts at Amherst, Jan. 1987.

[3] Andresen, F.P., Davis, L.S., Eastman, R., and Kambhampati, S., "Visual Algorithms for Autonomous Navigation", *Proc. IEEE Int. Conference on Robotics and Automation*, St. Louis, Mo., pp. 856-861, 1985.

[4] Andresen, F. and Davis, L., "Visual Position Determination for Autonomous Vehicle Navigation", *Center for Automation Research TR-100*, Univ. of Maryland, 1984.

[5] Arbib, M.,*The Metaphorical Brain*, Wiley, 1972.

[6] Arbib, M., "Perceptual Structures and Distributed Motor Control", *Handbook of Physiology - The Nervous System II*, ed. Brooks, pp. 1449-1465, 1981.

[7] Arbib, M., Iberall, T. and Lyons, D., "Coordinated Control Programs for Movements of the Hand",*COINS Technical Report 83-25* Dept. of Comp. and Info. Science, University of Massachusetts, Amherst, Ma., 1983, (also *Exp. Brain Res. Suppl.*, 10, pp. 111-129, 1985.

[8] Arbib, M. and House, D., "Depth and Detours: An Essay on Visually Guided Behavior", *COINS Technical Report 85-20* Dept. of Comp. and Info. Science, University of Massachusetts, Amherst, Ma., 1985; also in *Vision, Brain, and Cooperative Computation*, eds. Arbib and Hanson, MIT Press, 1987.

[9] Arkin, R., "Path Planning for a Vision-based Mobile Robot", *Mobile Robots, SPIE Proc. Vol. 727*, 1986. Also *COINS Technical Report 86-48*, Dept. of Computer and Information Science, Univ. of Mass., 1986.

[10] Arkin R., "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior", *Proc. IEEE International Conference on Robotics and Automation*, Raleigh, N.C., 1987.

[11] Arkin, R., "Path Planning and Execution for a Mobile Robot: A Review of Representation and Control Strategies", *COINS Technical Report 86-47*, Computer and Information Science Department, University of Massachusetts, Spring 1986.

[12] Arkin, R.,"User's Manual and Software Development Guide for the UMASS DRV Mobile Robot", *VISIONS Working paper*, Computer and Information Science Department, University of Massachusetts, 1986.

[13] Arkin, R., Riseman, E. and Hanson, A., "AuRA: An Architecture for Vision-based robot Navigation", *Proceedings of the DARPA Image Understanding Workshop*, pp. 417-431, Los Angeles, Feb. 1987.

[14] Arkin, R., "Internal Control of a Robot: An Endocrine Analogy", unpublished paper, Dec. 1984.

[15] Arkin, R., "Reactive/Reflexive Navigation for an Autonomous Vehicle", to appear in *Proceedings of the American Institute for Aeronautics and Astronautics Computers in Aerospace VI Conference*", Wakefield, MA, October 1987.

[16] Avrunin, G. and Wileden, J., "Describing and Analyzing Distributed Software System Designs", *Report*, University of Massachusetts, Amherst, 1985.

[17] Ayache, N. and Faugeras, O., "HYPER: A New Approach for the Recognition and Positioning of Two-Dimensional Objects", IEEE Trans. on Pattern Anal. and Machine Intel., Vol. 8, No. 1, pp. 44-54, 1986.

[18] Ballard, D. and Brown, C., *Computer Vision*, Prentice-Hall, 1982, pp. 69-70.

[19] Bartlett, F.C., *Remembering: A Study in Experimental and Social Psychology*, London, Cambridge at the Univ. Press, 1932.

[20] Bazakos, M., Panda, D., and Duncan, D., "Passive Ranging in Outdoor Environment", *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, Mo., pp. 836-842, 1985.

[21] Bharwani, S., Riseman, E. and Hanson, A., "Refinement of Environmental Depth Maps over Multiple Frames", *Proc. IEEE Workshop on Motion Representation and Analysis*, pp. 73-80, May 1986.

[22] Bharwani, S., Riseman, E. and Hanson, A.,"Multiframe Computation of Accurate Depth Maps using Uncertainty Analysis", *COINS Tech. Report* (in preparation), Dept. of Computer and Information Science, Univ. of Mass., 1986.

[23] Brady, M., "Artificial Intelligence and Robotics", *Artificial Intelligence*, vol. 26, no. 1, pp. 79-121, April 1985.

[24] Brooks, R., "A Robust Layered Control System for a Mobile Robot", *IEEE Jour. of Robotics and Auto.*, Vol. RA-2, No. 1, 1986 pp. 14-23.

[25] Brooks, R.A., "Visual Map Making for a Mobile Robot", *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, Mo., pp. 824-829, 1985.

[26] Brooks, R., "Planning Collision-free Motions for Pick-and-Place Operations", *The Int. Jou. of Robotics Res.*, Vol. 2, No. 4 , pp. 19-37, Winter 1983.

[27] Brooks, R., "Solving the Find-Path Problem by Good Representation of Free Space", *IEEE Systems, Man, and Cyber.*, SMC-13, No. 3, pp. 190-196, 1983.

[28] Brooks, R., "Symbolic Error Analysis and Robot Planning", *The Int. Jou. of Robotics Res.*, Vol. 1-4, pp. 29-68, Winter 1982.

[29] Brooks, R., "Aspects of Visual Map Making", *Robotics Research, 2nd Int. Symp.*, MIT Press, pp. 369-376, 1985.

[30] Brown, M., "Feature Extraction Techniques for Recognizing Solid Objects with an Ultrasonic Range Sensor", *IEEE Journal of Robotics and Automation*, RA-1, No. 4, pp. 191-205, December 1985.

[31] Burns, J., Hanson, A., Riseman, E., "Extracting Straight Lines", *Proc. IEEE 7th Int. Conf. on Pattern Recognition*, Montreal, Can, pp. 482-485. 1984.

[32] Burns, J. B. and Kitchen, L., "Recognition in 2D Images of 3D Objects from Large Model Bases using Prediction Hierarchies", *Proc. Int. Joint conf. on Artif. Int.*, Milan, 1987.

[33] Chatila, R. and Laumond, J.P., "Position referencing and Consistent World Modeling for Mobile Robots", *Proc. IEEE Int. Conf. Rob. and Auto.*, St. Louis, Mo., pp. 138-145, 1985.

[34] Chattergy, "Some Heuristics for the Navigation of a Robot", *Int. Jour. of Robotics Research*, Vol. 4, No. 1, pp. 59-66, Spring 1985.

[35] Chong, K. and Hsia, P., "Software Diagnostics for Distributed Data Processing Systems", *IEEE CH1941-4/83*, pp. 154-159, 1983.

[36] Crowley, J., "Navigation for an Intelligent Mobile Robot", *CMU Robotics Institute Tech. Rep.*, CMU-RI-TR-84-18, 1984.

[37] Davis, E., "Organizing Spatial Knowledge", *Research Report no. 193*, Dept. of Computer Science, Yale University, 1981.

[38] "Denning Research Vehicle (DRV-1) Software Documentation", Denning Mobile Robotics, Inc., Woburn, Ma., 1985.

[39] Dickmanns, E. and Zapp, A., "Guiding Land Vehicles along Roadways by Computer Vision", *AFCET Conference*, Toulouse, France, 1985.

[40] Draper, B., "Introduction to the Schema Shell", and "Programmer's Reference for using the Schema Shell", *VISIONS working papers*, Univ. of Massachusetts, Amherst, 1986.

[41] Draper, B., Hanson, A. and Riseman, E., "The Schema System" *COINS Tech. Report* (in preparation), Dept. of Computer and Information Science, Univ. of Mass., 1987.

[42] Draper, B., Collins, R., Brolio, J., Griffith, J., Hanson, A. and Riseman, E., "Tools and Experiments in the Knowledge-Directed Interpretation of Road Scenes", COINS Technical Report , Dept. of Computer and Info. Science, Univ. of Mass., 1987.

[43] Drumheller, M., "Mobile Robot Localization Using Sonar", *A.I. Memo 826*, MIT Artif. Intel. Lab, January 1985.

[44] Eddy, W., "Algorithm 523 - Convex, A new Convex Hull Algorithm for Planar Sets", *Collected Algorithms from ACM*, no. 523, 1977.

[45] Elfes, A., "A Sonar-based Mapping and Navigation System", *Proc. IEEE Int. Conf. on Robotics and Auto.*, San. Fran., Calif., pp. 1151-1156, 1986.

[46] Feigenbaum, Barr and Cohen, *The Handbook of Artificial Intelligence*, William Kaufman, 1981.

[47] Flynn, A., "Redundant Sensors for Mobile Robot Navigation", Master's thesis, Mass. Inst. of Tech., 1985.

[48] Frederies, J.A.M., "Disorders of the Body Schema", *Handbook of Clinical Neurology, Disorders of Speech Perception and Symbolic behavior*, Ed. Vinken and Bruyn, North Holland, 1969, Vol. 4, pp. 207-240.

[49] Fukui, I., "TV Image Processing to Determine the Position of a Robot Vehicle", *Pattern Recognition*, Vol. 14, Nos. 1-6, pp. 101-109, 1981.

[50] Geschke, C., "A System for Programming and Controlling Sensor-based Robot Manipulators", *IEEE Trans. on PAMI*, PAMI-5, No. 1, 1983, pp. 1-7.

[51] Giralt, G., "Mobile Robots", *Artificial Intelligence and Robotics*, ed. Brady, Gerhardt, and Davidson, Springer-Verlag, NATO ASI series, pp. 365-394, 1984.

[52] Giralt, G., Chatila, R., and Vaisset, M., "An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robots", *Robotics Research, The First International Symposium*, MIT Press, pp. 191-214, 1984.

[53] Hanson, A. and Riseman, E., "VISIONS, A Computer System for Interpreting Scenes", *Computer Vision Systems*, (Hanson and Riseman eds.), Academic Press, pp. 303-333, 1978.

[54] Hanson, A., and Riseman, E., "A Summary of Image Understanding Research at the University of Massachusetts", *COINS Tech. Report 83-35*, Dept. of Computer and Information Science, Univ. of Mass., 1983.

[55] Hanson A. and E. Riseman, E., "The VISIONS Image Understanding System - 1986", in *Advances in Computer Vision*, (Chris Brown ed.), to be published by Erlbaum Press.

[56] Hart, P.E., Nilsson, N.J., and Raphael, B. "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", *IEEE Trans. Syst. Science and Cybernetics*, SSC-4(2), pp. 100-107, 1968.

[57] Head, H. and Holmes, G., "Sensory disturbances from cerebral lesions", *Brain* 34:102-254, 1911.

[58] Iberall, T. and Lyons, D., "Towards Perceptual Robotics", *Report from Laboratory for Perceptual Robotics*, University of Massachusetts.

[59] Iyengar, S. Jorgensen, C., Rao, S., and Weisbin, C., "Learned Navigation Paths for a Robot in Unexplored Terrain", *IEEE Second Conf. on Artif. Intel. App.*, pp. 148-155, December 1985.

[60] Kadonoff, M., Benayad-Cherif, F., Franklin, A., Maddox, J., Muller, L., Sert, B. and Moravec, H., "Arbitration of Multiple Control Strategies for Mobile Robots", Mobile Robots - *SPIE proc. Vol. 727*, 1986.

[61] Kahn, P., Kitchen, L. and Riseman, E., "Real-time Feature Extraction: A Fast Line Finder for Vision-Guided Robot Navigation", *COINS Technical Report TR-87-57*, Dept. of Comp. and Info. Science, University of Massachusetts, Amherst, Ma., 1987.

[62] Kak, A.C., Depth Perception for Robots", *Technical Report TR-EE-83-44*, Purdue University, 1983, (also *Handbook of Industrial Robotics*, Ed. S. Nof, Wiley).

[63] Keirsey, D., Mitchell, J., Payton, D., and Preyss, E., "Multilevel Path Planning for Autonomous Vehicles", *SPIE Vol. 485, Applications of Artificial Intelligence*, pp. 133-137, 1984.

[64] Khatib, O., "Real-time Obstacle Avoidance for Manipulators and Mobile Robots". *Proc. IEEE Int. Conf. Robotics and Automation*, pp. 500-505, St. Louis, 1985.

[65] Kitchen, L. and Rosenfeld, I., "Grey-level Corner Detection", *Technical Report 887*, Comp. Sci. Ctr., U. Maryland, College Park, Md, 1980.

[66] Koch, E., Yeh, C., Hillel, G., Meystel, A., and Isik, C., "Simulation of Path Planning for a System with Vision and Map Updating", *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, Mo., pp.146-160, 1985.

[67] Kohler, R., "Integrating Non-semantic Knowledge into Image Segmentation Processes", *Ph.D. Thesis, COINS TR-84-04*, Dept. of Computer and Information Science, Univ. of Mass., 1984.

[68] Krogh, B., "A Generalized Potential Field Approach to Obstacle Avoidance Control", *SME - RI Technical Paper* MS84-484, 1984.

[69] Krogh, B. and Thorpe, C., "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles", *IEEE Conf. on Robotics and Auto.*, pp. 1664-1669, 1986.

[70] Krotkov, E., Kories, R., and Henriksen, K., "Stereo Ranging with verging camera: A practical calibration procedure and error analysis", *MS-CIS-86-86*, Department of Computer and Information Science, University of Pennsylvania, 1986.

[71] Kuan, D.T., Zamiska, J., and Brooks, R., "Natural Decomposition of Free Space for Path Planning", *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, Mo., pp.168-173, 1985.

[72] Kweon, I., "Trinocular Camera Model Calibration", Dept. of Mechanical Engineering Eng., The Robotics Institute, Carnegie-Mellon University, March 1986.

[73] Lawton, D., "Processing Dynamic Image Sequences from a Moving Sensor", *Ph.D. dissertation, COINS Technical Report 84-05*, Dept. of Computer and Information Science, Univ. of Mass., 1984.

[74] LeMoigne, J., Waxman,A., Srinivasan, B., and Pietikainen, M., "Image Processing for Visual Navigation of Roadways", *IEEE Jour. of Rob. and Auto.*, 1987.

[75] Lewis, T., Spitz, K., and McKenney, P., "An Interleave Principle for Demonstrating Concurrent Programs", *IEEE Software*, pp. 54-63, Oct. 1984.

[76] Lozano-Perez, T., "Automatic Planning of Manipulator Transfer Movements", in *Robot Motion: Planning and Control*, ed. Brady et al, pp. 499-535,1982.

[77] Lyons, D. and Arbib, M., "A Task-Level Model of Distributed Computation for Sensory-Based Control of Complex Robot Systems", *COINS Tech. Rep.* 85-30 (also *IFAC Symposium on Robot Control*, Barcelona), 1985

[78] Lyons, D., "RS: A Formal Model of Distributed Computation for Sensory-Based Robot Control", *Ph.D. Dissertation, COINS Technical Report 86-43* Dept. of Comp. and Info. Science, University of Massachusetts, Amherst, Ma., 1986.

[79] Lyons D., "Tagged Potential Fields: An Approach to Specification of Complex Manipulator Configurations", *IEEE Conf. on Robotics and Auto.*, pp. 1749-1754, 1986.

[80] Matthies, L. and Shafer, S., "Error Modelling in Stereo Navigation", *Technical Report CMU-CS-86-140*, Carnegie-Mellon University, 1986.

[81] Matthies, L., "Modelling Uncertainty in 3D Stereo Navigation", from *Autonomous Mobile Robots Annual Report 1985*, CMU Robotics Institute TR-86-4, 1985.

[82] McDermott, D., and Davis, E., "Planning Routes through Uncertain Territory", *Artificial Intelligence* 22, 107-156, 1984.

[83] Meystel, A., "Planning in a Hierarchical Nested Autonomous Control System", *Mobile Robots - SPIE Proc. Vol. 727*, pp. 42-76, 1986.

[84] Miller, D., "A Spatial Representation System for Mobile Robots", *CH2152-7/85, IEEE* , pp. 122-127, 1985.

[85] Mitchell, J. and Papdimitriou, C., "Planning Shortest Paths", *(extended abstract)* - personal communication, September 1985.

[86] Mitchell, J., "An Autonomous Vehicle Navigation Algorithm", *SPIE Vol. 485, Applications of Artificial Intelligence*, pp. 153-158, 1984.

[87] Mitchell, J., Mount, D., Papadimitriou, C., "The Discrete Geodesic Problem", personal communication.

[88] Mitchell, J. and Keirsey, D., "Planning Strategic paths through Variable terrain data", *SPIE Vol. 485, Applications of Artificial Intelligence*, pp. 172-179, 1984.

[89] Mobile Robot Laboratory Staff, "Towards Autonomous Vehicles", *Annual Research Review*, The Robotics Institute, Carnegie-Mellon University, 1984.

[90] Monaghan, G., "World Modeling and Path Planning for Autonomous Mobile Robots", *manuscript 579-51.*

[91] Moravec, H., "Locomotion, Vision, and Intelligence", *Robotics Research, The First International Symposium*, MIT Press, pp. 215-224, 1984.

[92] Moravec, H. and Elfes, A., "High Resolution Maps from Wide Angle Sonar", *CH2152-7/85 1985 IEEE*, pp. 116-121, 1985.

[93] Moravec, H., *Robot Rover Visual Navigation*, UMI Press, 1981.

[94] Morgenthaler, D., Gremban, K.D., Nathan, M., and Bradstreet, J., "The ITA Range Image Processing System", *Proceedings: DARPA Image Understanding Workshop*, Los Angeles, pp. 127-137, 1987.

[95] Neisser, U., *Cognition and Reality: Principles and Implications of Cognitive Psychology*, Freeman, 1976.

[96] Nilsson, N., *Principles of Artificial Intelligence*, Tioga Publishing Co., 1980.

[97] Nilsson, N., ed.,"Shakey the Robot", SRI International *Technical Note 323*, 1984. Also, Rosen, C. and Nilsson, N., "Application of Intelligent Automata to Reconnaissance, First Interim Report, SRI, 1966.

[98] Nitao, J. and Parodi, A., "An Intelligent Pilot for an Autonomous Vehicle System", *IEEE Second Conf. On Artif. Intel. App.*, pp. 176-183, December 1985.

[99] Oldfield, R.C. and O.L. Zangwill, "Head's Concept of the Schema and its Application in Contemporary British Psychology - Parts I and II", *Br.J. Psychol.* 32:267-286 1942, 33:58-64,113-129,143-149, 1943.

[100] Ooka, A., Ogi, K., Wada, Y., Kida, Y., Takemoto, A., Okamoto, K., and Yoshida, K., "Intelligent Robot System II", *Robotics Research, Second Int. Symposium*, ed. Hanafusa and Inoue, MIT Press, pp. 341-347, 1985.

[101] Overton, K., "The Acquisition, Processing, and Use of Tactile Sensor Data in Robot Control", *Ph.D. Dissertation, COINS Technical Report 84-08*, Dept. of Comp. and Info. Science, University of Massachusetts, Amherst, Ma., 1984.

[102] Parma, C., Hanson, A. and Riseman E., "Experiments in Schema-driven Interpretation of a Natural Scene", *COINS Tech. Rep. 80-10*, Comp. and Info. Sci. Dept., Univ. of Massachusetts, 1980.

[103] Parodi, A., Nitao, J. and McTamaney, L.S., "An Intelligent System for an Autonomous Vehicle", *Proc. IEEE Int. Conf. on Robotics and Auto.*, San. Fran., Calif., pp. 1657-1663,1986.

[104] Parodi, A.M., "Multi-Goal Real-time Global Path Planning for an Autonomous Land Vehicle using a High-speed Graph Search Processor", *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, Mo., pp. 161-167, 1985.

[105] Pavlin, I., Riseman, E. and Hanson, A., "Translational Motion Algorithm with Global Feature Constraints", *COINS Technical Report 86-58*, Dept. of Computer and Information Science, Univ. of Mass., 1986.

[106] Pavlin, I. and Snyder, M.,"The Problem of Registration of Dynamic Image Sequences", *COINS Technical Report* in preparation, Dept. of Computer and Information Science, Univ. of Mass., July 1987.

[107] Payton, D., "An Architecture for Reflexive Autonomous Vehicle Control", *IEEE Conf. on Robotics and Auto.*, pp. 1838-1845, 1986.

[108] Pearson, G. and Kuan, D., "Mission Planning System for an Autonomous Vehicle", *IEEE Second Conf. On Artif. Intel. App.*, pp. 162-167, December 1985.

[109] Piaget, J., *Biology and Knowledge*, Univ. of Chicago Press, 1971.

[110] Pocock, G., "A Distributed Robot Control System", *Ph.D. proposal*, Univ. of Massachusetts, Amherst, Ma., 1986.

[111] Rigoutsos, I., "Camera Calibration", Department of Computer Science, University of Rochester, 1986.

[112] Rigoutsos, I., "Determining the coordinates of the pixel on the DataCube display through which the optical axis of the camera in use passes - (0,0) point of the camera", Department of Computer Science, University of Rochester University of Rochester, 1986.

[113] Rigoutsos, I., "Camera Calibration: Useful Hints", Department of Computer Science, University of Rochester University of Rochester, 1986.

[114] Sedgewick, R., *Algorithms*, Addison-Wesley, p. 313, 1983.

[115] Shafer, S., Stentz, A., and Thorpe, C., "An Architecture for Sensor Fusion in a Mobile Robot", *Proc. IEEE Int. Conf. on Robotics and Auto.*, San. Fran., Calif., pp. 2002-2011, 1986.

[116] Shamos, I. and Hoey, D., "Closest Point Problems", *16th Annual Symposium on the Foundations of Computer Science*, pp. 151-162, October, 1975.

[117] Shen, H. and Signarowski, G., "A Knowledge Representation for Roving Robots", *IEEE Second Conf. on Artif. Intel. App.*, pp. 621-628, December 1985.

[118] Smith, R., and Cheeseman, P., "On the Representation and Estimation of Spatial Uncertainty", *Robotics Lab Technical Paper, SRI Projects 4760 and 7239*, SRI International, Sept. 1985.

[119] Snyder, M., "The Accuracy of 3D Parameters in Correspondence-based Techniques", *COINS Technical Report 86-28*, Dept. of Computer and Information Science, Univ. of Mass., July 1986.

[120] Stentz, A. and Shafer, S., "Module Programmer's Guide to Local Map Builder for ALVan", CMU Computer Science Department, July 1985.

[121] Tachi, S. and Komoriya, K., "Guide Dog Robot", *Robotics Research, Second Int. Symposium*, ed. Hanafusa and Inoue, MIT Press, pp. 333-340, 1985.

[122] Taylor, R., "A General Purpose Algorithm for Analyzing Concurrent Programs", *Comm. of Assoc. Comp. Mach.*, Vol. 26, No. 5, pp. 362-376, May 1983.

[123] Thompson, A. M., "Camera Geometry for Robot Vision", *Robotics Age*, Mar/Apr. 1981, pp. 20-27.

[124] Thorpe, C. and Kanade, T., "Vision and Navigation for the CMU Navlab", *Mobile Robots - SPIE Proc. Vol. 727*, 1986.

[125] Thorpe, C., Matthies, L., and Moravec, H., "Experiments and Thoughts on Visual Navigation", *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, Mo., pp. 830-835, 1985.

[126] Thorpe, C., "Path Relaxation: Path Planning for a Mobile Robot", *CMU Robotics Institute Technical Report CMU-RI-TR-84-5*, April 1984.

[127] Thorpe, C., Kanade, T., and Shafer, S., "ALV System Integration Plan", The Robotics Institute, CMU, 1985.

[128] Tou, J., " Software Architecture of Machine Vision for Roving Robots", *Optical Engineering*, Vol. 25-3, pp. 428-435, March 1986.

[129] Tsuji, S., Yagi, Y., and Asada, M., "Dynamic Scene Analysis for a Mobile Robot in a Man-made Environment", *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, Mo., pp. 850-855, 1985.

[130] Tsuji, S., "Monitoring of a Building Environment by a Mobile Robot", *Robotics Research, Second Int. Symposium*, ed. Hanafusa and Inoue, MIT Press, pp. 349-356, 1985.

[131] Turchan, M. and Wong, A., "Low Level Learning for a Mobile Robot: Environment Model Acquisition", *IEEE Second Conf. On Artif. Intel. App.*, pp. 156-161, December 1985.

[132] Wallace, R., Matsuzaki, K., Crisman, J., Goto, Y., Webb, J. and Kanade, T., "Progress in Robot Road-Following", *Proc. 1986 IEEE Int. Conf. on Robotics and Auto.*, San Fran., CA., pp. 1615-1621, 1985.

[133] Wallace, R., Stentz, A., Thorpe, C., Moravec, H., Whittaker, W. and Kanade, T., "First Results in Robot Road-Following", *Int. Joint Conf. Art. Int.*, Vol. 2, pp. 1089-1095, 1985.

[134] Wallace, R., "Robot Road Following by Adaptive Color Classification and Shape Tracking", *Proc. IEEE Int. Conf. on Rob. and Auto.*, pp. 258-263, 1987.

[135] Waxman, A., LeMoigne, J., Davis, L., Liang, E., and Siddalingaiah,T., "A Visual Navigation System for Autonomous Land Vehicles", *IEEE Jour. of Rob. and Auto.*, RA-3, pp. 124-141, 1987.

[136] Waxman, A., LeMoigne, J. Davis, L., Liang, E., and T. Siddalingaiah, "A Visual Navigation System", *Proc. IEEE Int. Conf. on Robotics and Auto.*, San. Fran., Calif., pp. 1600-1606,1986.

[137] Waxman, A.M., Le Moigne, J., and Srinivasan, B., "Visual Navigation of Roadways", *Proc. IEEE Int. Conf. Robotics and Automation*, St. Louis, Mo., pp. 862-867, 1985.

[138] Webster's *Ninth New Collegiate Dictionary*, Merriam-Webster, 1984.

[139] Weems, C., "Image Processing on a Content Addressable Array Parallel Processor", *Ph.D. Dissertation and COINS Tech. Report 84-14*, Dept. of Computer and Information Science, Univ. of Mass., Sept. 1984.

[140] Weymouth, T., "Using Object Descriptions in a Schema Network for Machine Vision", *Ph.D. Dissertation, COINS Technical Report 86-24* Dept. of Comp. and Info. Science, University of Massachusetts, Amherst, Ma., May, 1986.

[141] Whittaker, W., Turkiyyah, G. and Hebert, M., "An Architecture and Two Cases in Range-based modeling and Planning",*Proc. IEEE Int. Conf. on Rob. and Auto.*, pp. 1991-1997, 1987.

[142] Williams, T., "Computer Interpretation of a Dynamic Image from a Moving Vehicle", *COINS Technical Report 81-22*, Dept. of Computer and Information Science, Univ. of Mass., May 1981.

[143] Boldt, M. and Weiss, R., "Token-based Extraction of Straight Lines", *COINS Technical Report* in preparation, Dept. of Computer and Information Science, Univ. of Mass., 1987.