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

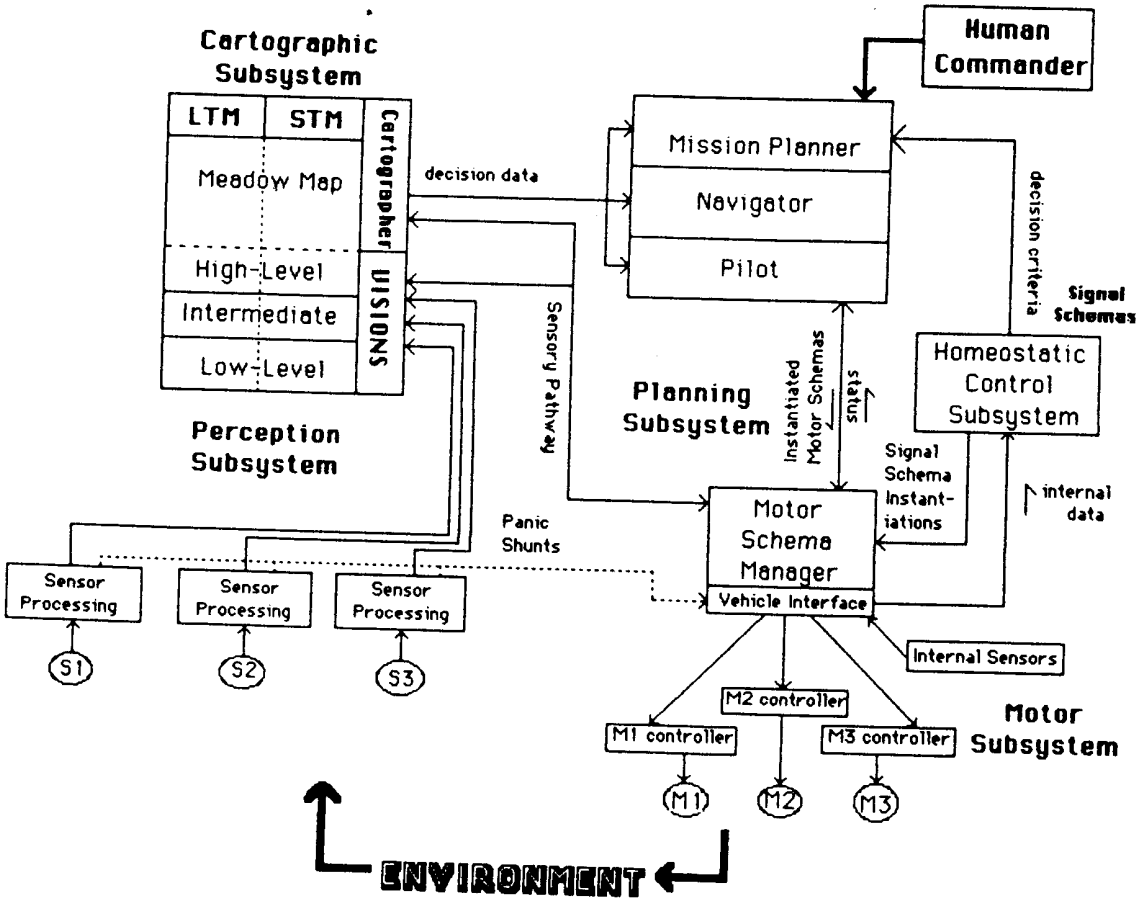


Figure 2: System architecture fo
Block diagram
design for the Autonomous Robot
Architecture (A

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.

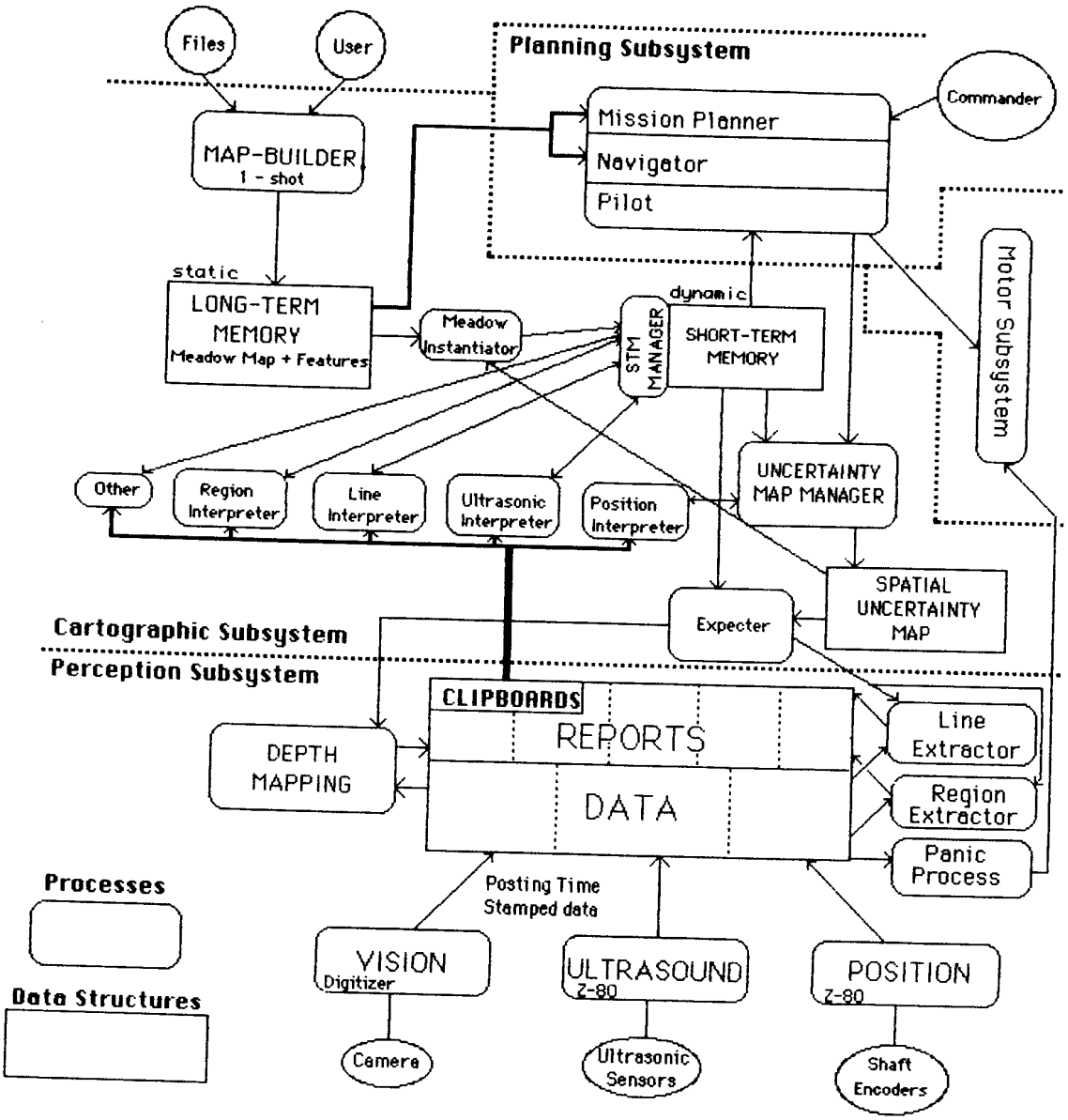


Figure 3: First pass implementation of AuRA architecture

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*

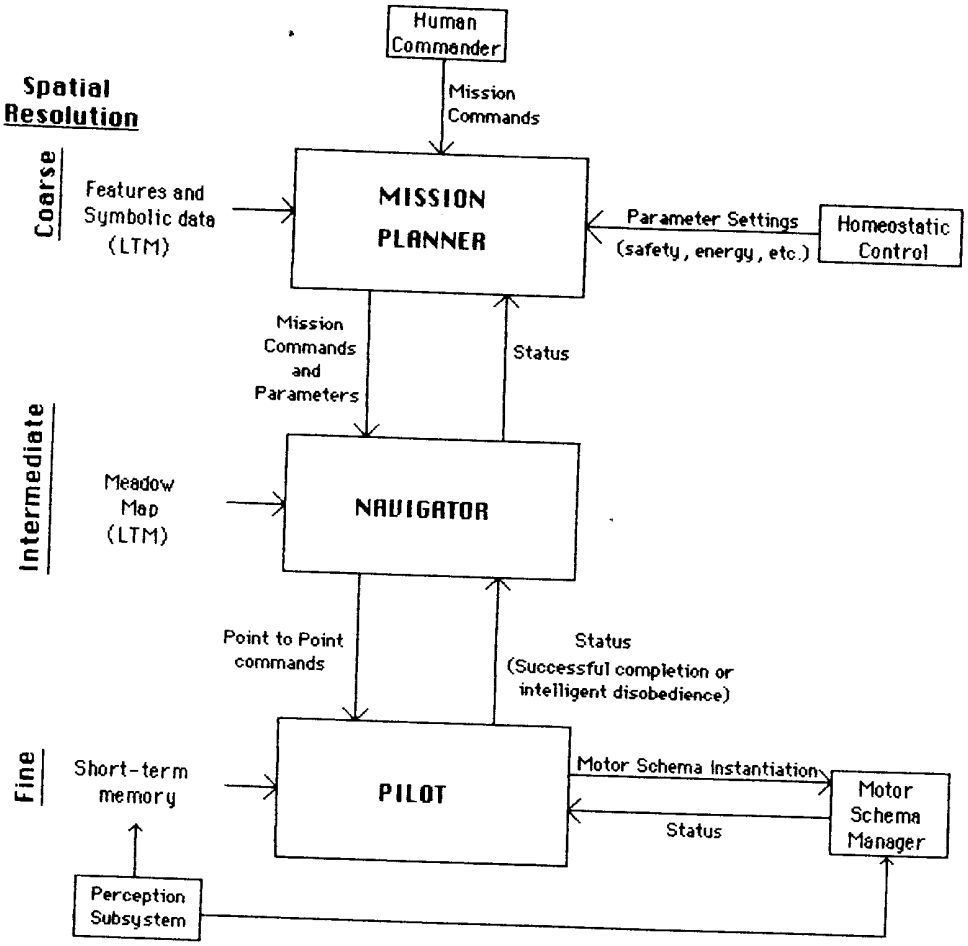


Figure 4: Hierarchical planner for AuRA

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

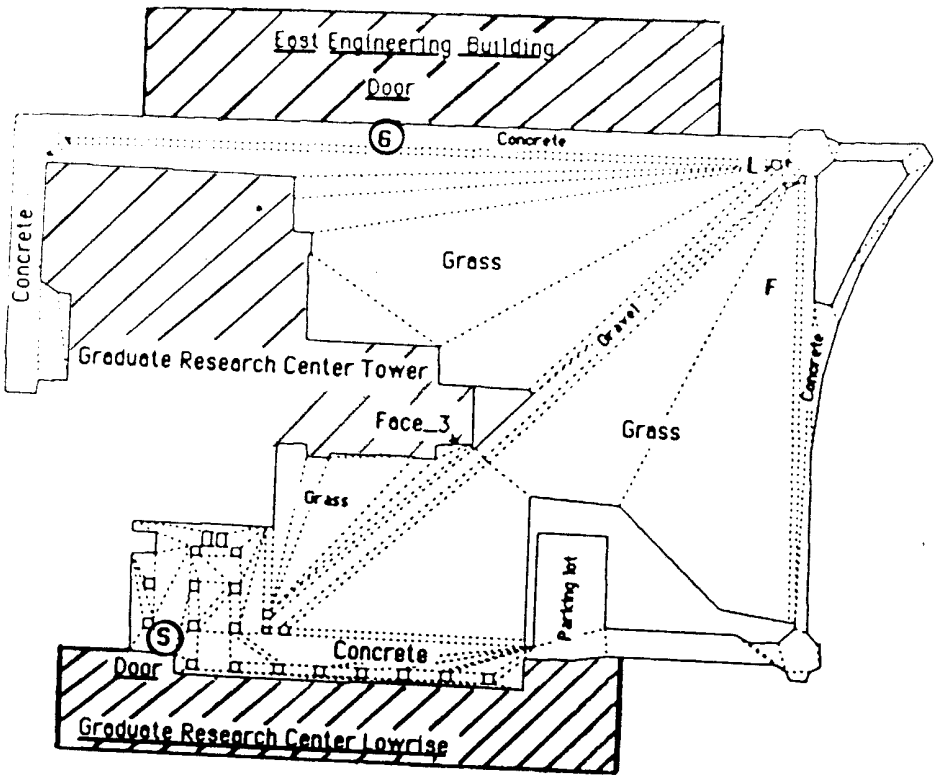
visual algorithm or perceptual schema is associated with it. Additionally, various perceptual schemas are instantiated to identify available landmarks (as predicted by long-term memory and the current uncertainty in the robot's position). These are used to localize the vehicle without necessarily evoking 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
6 - 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**

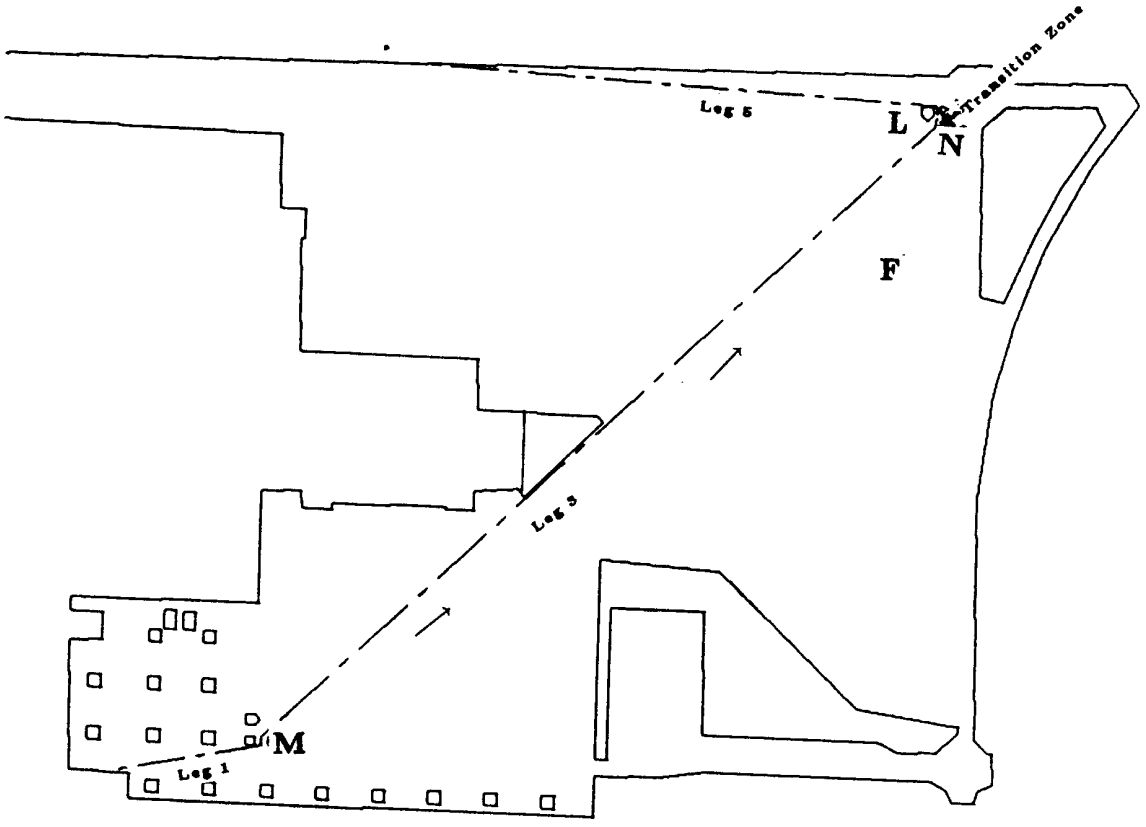


Figure 6: Global path constructed by navigator

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.

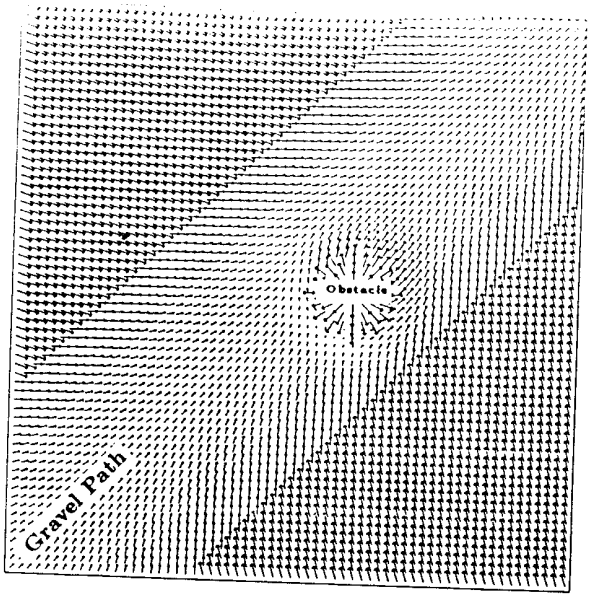
We define a schema instantiation (SI) to be the activity of applying a general class of schemas to a specific case [6,7,140]. The fact that a lamppost is present at the end of the path results in the creation of a **find-landmark** SI dedicated to finding LAMP-POST_107, whose model is extracted from 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

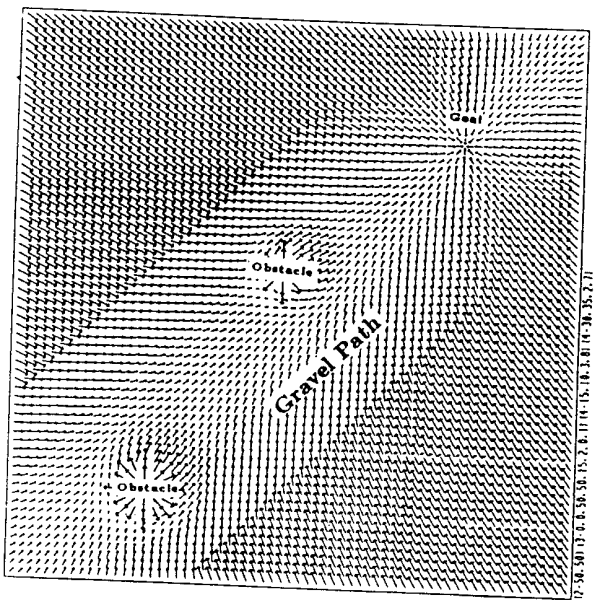
The arrows represent the desired velocity vectors that constrain the robot's motion, indicating the velocity the robot should undertake based on its position within the field. The primitive schema potential fields that are summed to yield this composite behavior appear in Fig. 47.

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.

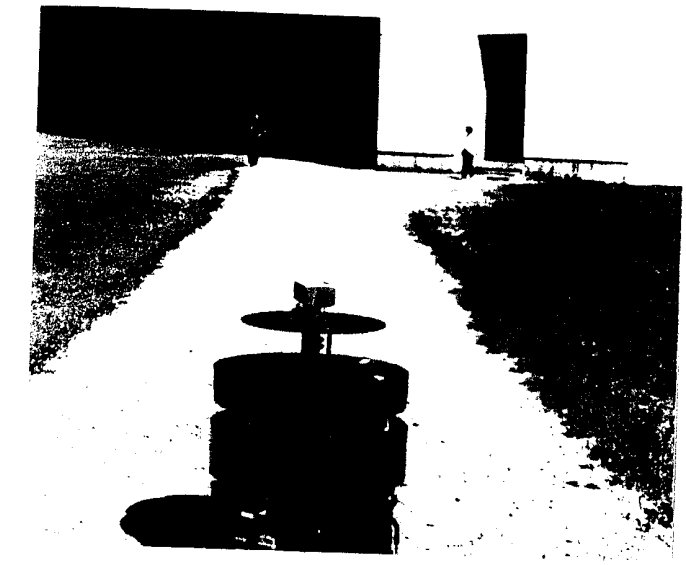
§5.1 *Hardware configuration*

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.

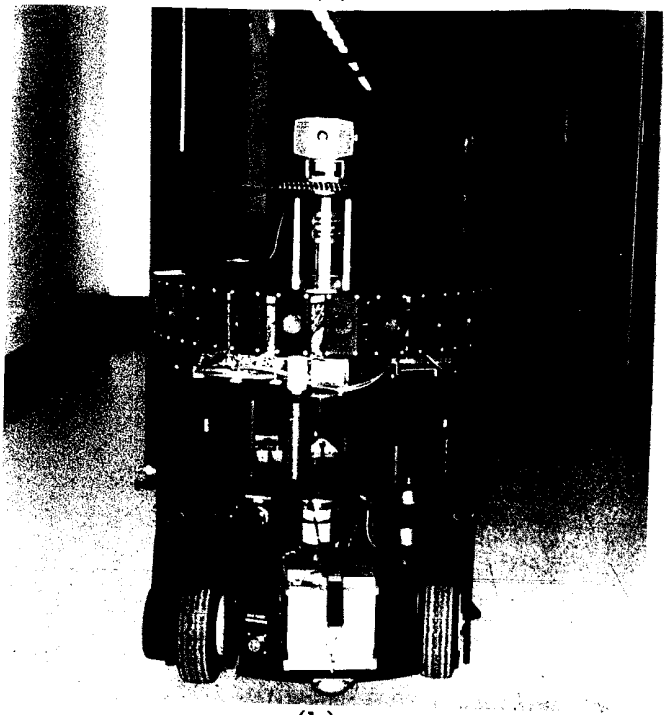
The basic hardware support is depicted in Figure 9. Most of the software runs on the VISION's group VAXen. The LTM mapbuilding and STM maintenance components of the cartographer, higher level components of the planning system, motor subsystem and the perception algorithms are coded in C. The pilot and 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

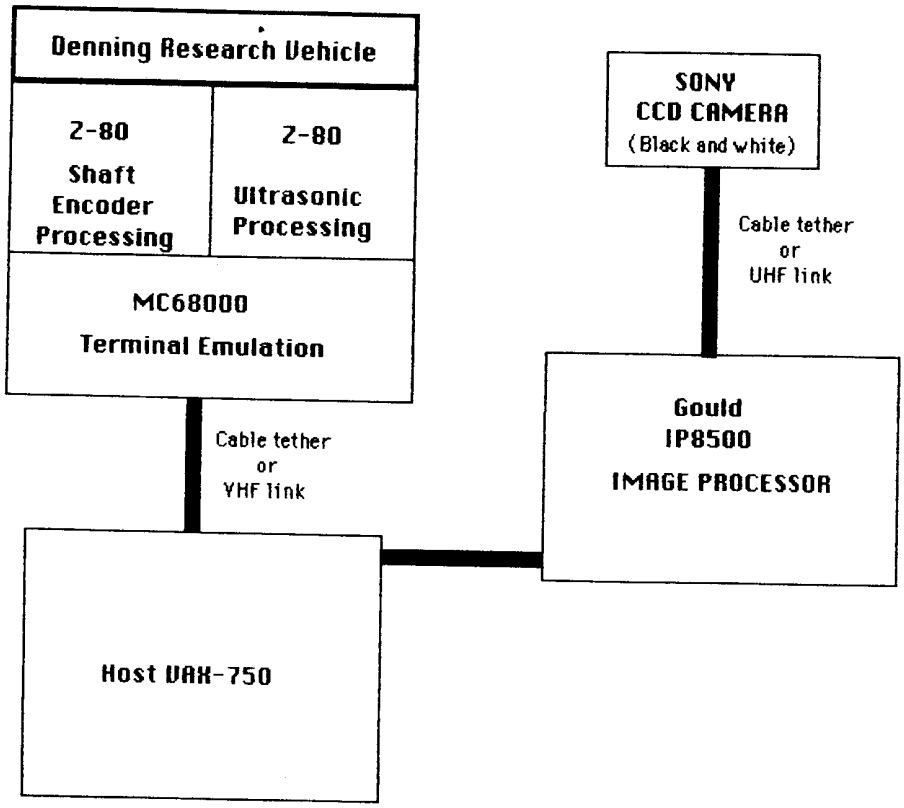


Figure 9: First pass AuRA Hardware

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.

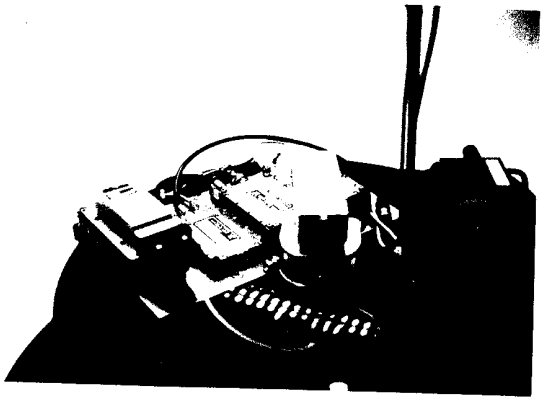
The invisible tether consists of a UHF/VHF TV-radio link broadcasting on channel 50 using satellite TV technology. A mobile station (Fig. 10) powered by separate batteries and not the robot's own power supply (to reduce noise) transmits video images while receiving motor and status commands on a separate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

§5.3 *Clipboards*

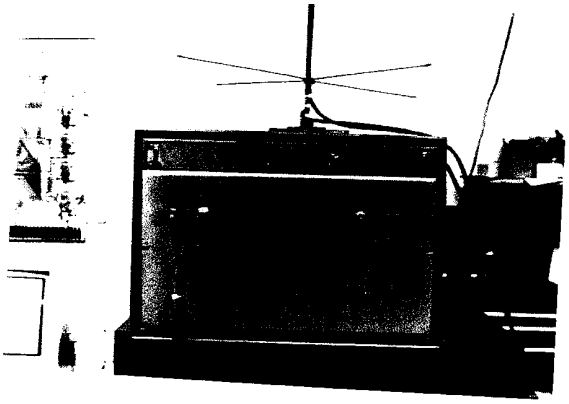
In order to effectively communicate the results from differing concurrent perceptual and interpretive processes, a global data structure termed clipboards has been developed. Clipboards are related to a heterarchical blackboard data structure, similar to the whiteboards used in the NAVLAB [115] and the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