From the DARPA Image Understanding Workshop, February 1987, Los Angeles, CA. pp. 417-431.

AuRA: AN ARCHITECTURE FOR VISION-BASED ROBOT NAVIGATION

Ronald C. Arkin
Edward M. Riseman
Allan R. Hanson

Computer and Information Science Department, University of Massachusetts,
Graduate Research Center, Amherst, Massachusetts, 01003

Abstract

The VISIONS research environment at the University of Massachusetts provides an integrated system for the interpretation of visual data. A mobile robot has been acquired to provide a testbed for segmentation, static interpretation, and motion algorithms developed within this framework. The robot is to be operated in test domains that encompass both interior scenes and outdoor environments. The test vehicle is equipped with a monocular vision system and will communicate with the lab via a UHF transmitter-receiver link.

Several visual strategies are being explored. These include a fast line-finding algorithm for path following, a multiple frame depth-from-motion algorithm for obstacle avoidance, and the relationship of schema-based scene interpretation to mobile robotics, especially regarding vehicle localization and landmark recognition. These algorithms have been tailored to meet the needs of mobile robotics by utilizing top-down knowledge when available to reduce computational demands.

To facilitate the implementation of these algorithms, the UMASS Autonomous Robot Architecture (AuRA) has been developed. Employing both high-level semantic knowledge and control structures consisting of low-level motor schemas, action-oriented perception and schema-based navigation are being investigated. Initial path planning is conducted through a ground-plane meadow-map which contains semantic and visual data relevant to the actual navigational execution of the path. Information from the meadow-map along with the goals developed by the mid-level path planner are then passed to the pilot, which selects appropriate motor schemas for instantiation in the motor schema manager. Pertinent vision algorithms or perceptual schemas are associated with each instantiated motor schema to guide the vehicle along its way.

Results of both actual robot navigation in an outdoor campus environment and appropriate simulations are presented to show the viability of this AI architecture. These examples concentrate in the three areas of vision research mentioned above.

1. Introduction

The VISIONS group at the University of Massachusetts has an extensive and ongoing research project in the interpretation of real-world images [1,2]. More recently, efforts are being made to migrate many of the concepts developed within the VISIONS system to the application of mobile robot navigation. A mobile robot has been acquired to provide an experimental testbed for this effort. This vehicle (a Denning Mobile Robot - DRV) is equipped with a monocular video camera and a ring of 24 ultrasonic sensors.

This research was supported in part by the General Dynamics Corporation under grant DEY-601550 and the U.S. Army under ETL grant DACA76-85-C-008.

AuRA (Autonomous Robot Architecture) is a system architecture that provides necessary extensions to the VISIONS system that are primarily concerned with safe mobile robot navigation. 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.

The remainder of this paper is divided into the following sections. Section 2 will present an overview of the AuRA architecture. Section 3 will describe 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. Section 4 describes the relationship of VISIONS to AuRA, concentrating particularly on the VISIONS schema system. Modular vision algorithms, including techniques already embedded as well as those currently being investigated for potential incorporation, are described in section 5. Experimental results, including robot experiments as well as simulations, are presented. A summary of the paper and a brief description of future work complete the report.

2. Architecture Overview

A block diagram of AuRA is presented in Figure 1. 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 3. 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) stores 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 uncertainty in the vehicle's position.

A perception subsystem, (ultimately 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

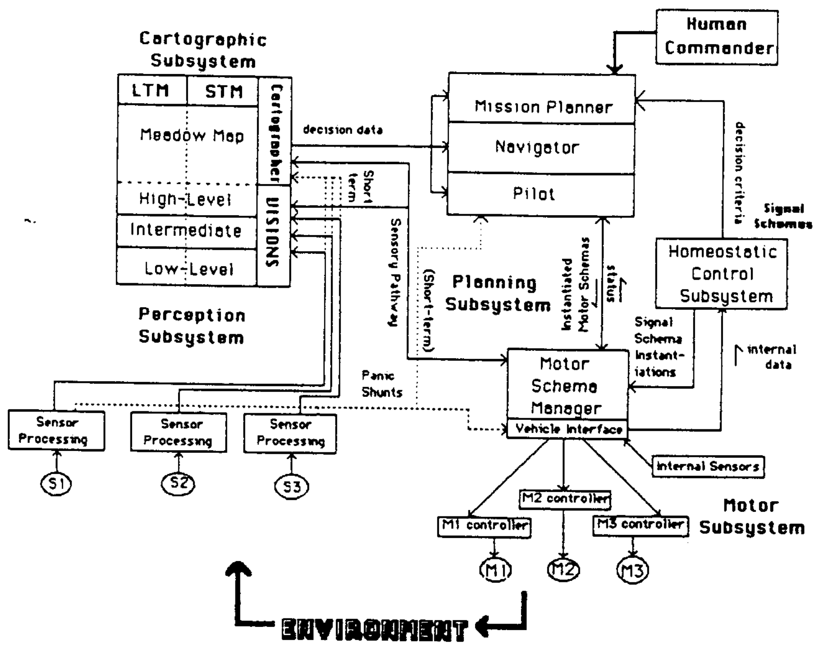


Figure 1. System Architecture for AuRA

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.

The first pass implementation of the perceptual system does not draw on the VISIONS system in its entirety. The VISIONS system is ultimately expected to be the location where all sensor fusion occurs, yielding ideally a rich 3-D model of the perceived world. At this stage in AuRA's development, however, relevant vision algorithms are extracted from the VISIONS environment and are used outside of its context. The real-time needs of mobile robotics can be handled by this strategy as the vision algorithms are not yet developed on parallel hardware. In so doing, the cartographer assumes greater responsibility than might be needed in future designs when many of the cartographer's responsibilities are subsumed by the VISIONS system. Figure 2 shows AuRA's initial implementation strategy. 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 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 3.)

2.1 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(STM)
  - instantiate models from LTM into STM
  - provide sensor expectations and to guide schema instantiations
- Maintenance of uncertainty at all levels of representation
  - spatial error map maintenance for robot localization
  - STM environmental uncertainty handling (object location)
- Initial LTM Map building (i.e. knowledge acquisition)

2.2 Perception Subsystem

Environmental sensor processing occurs within the confines of the perception subsystem. This component of AuRA consists of three submodule types: sensors, sensor processors, and the VISIONS system. In the early stages of the robot system development, we utilize a small subset of available vision algorithms, including simplified versions of some low-level algorithms that are tuned for real-time performance, until a full real-time scene interpretation VISIONS environment becomes available.

Sensor processors serve to preprocess the sensor data into a form that is acceptable to the receiving modules. The principal goal for these sensor-specific filters (e.g. from vision, ultrasonic or dead-reckoning sensors) is to simplify the job facing the VISIONS system by removing noisy, extraneous, or errorful data and by converting the relevant data from diverse sensors into a form that is integrable into world representations.

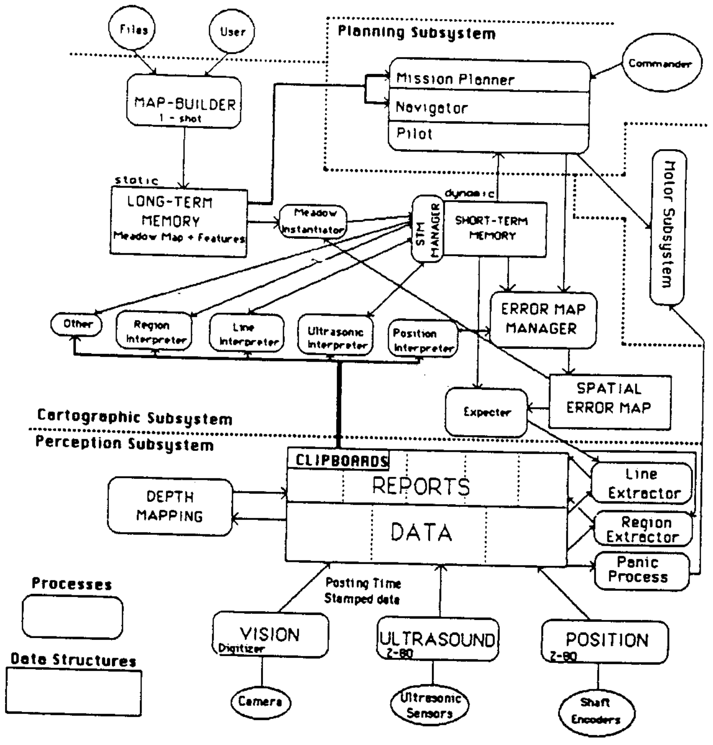


Figure 2. First Pass Implementation of AuRA Architecture

The VISIONS system is the heart of the perception subsystem. Multiple levels of 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 pilot or 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.

2.3 Motor Subsystem

The motor subsystem is delegated the responsibility of effecting the commands of the motor schema manager. The motor subsystem consists of three major components: motors, motor controllers, and vehicle interface. The steering motors, drive motors and motor controllers, in the case of the UMASS DRV, are provided by the manufacturer.

The vehicle interface, in its most general version, will use the output of motor schemas as a means for effecting action in the environment. This module is the one component of the overall architecture which is most profoundly influenced by the specific robot vehicle chosen. Vehicle independence is a design goal for all other AuRA modules. The vehicle interface translates the commands from the motor schema manager into the specific form required for the vehicle.

2.4 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. [3]) operating in hazardous environments simply determine if it is safe or not to enter a particular location, still others (e.g. [32]) make plans based on fuel reserves and other factors, but the robot's dynamic behavior is not considered.

In the proposed system, 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. A detailed description of the issues for such a homeostatic control system can be found in [5]. This extended functionality will provide the robot with enhanced survivability through a greater capability to respond to a changing environment.

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

3. Navigation

Several papers document the navigational and path planning techniques used in AuRA. The roles of the navigator and long-term memory are described in [6] and the motor schema manager's function is presented in [7]. A more complete description of the planning subsystem and navigational representations appears in [8]. The intent of this section is to provide 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.

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. 3), 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, the central characteristic of hierarchical control.

3.1 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 any 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.).

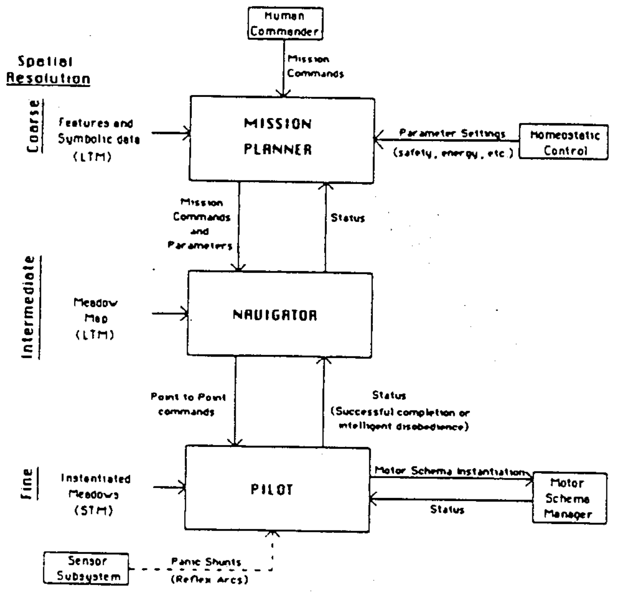

Human Commander
Spatial Resolution
Mission Commands
Coarse
Features and Symbolic data (LTM)
MISSION PLANNER
Parameter Settings (safety, energy, etc.)
Homeostatic Control
Mission Commands and Parameters
Status
Intermediate
Meadow Map (LTM)
NAVIGATOR
Point to Point commands
Status (Successful completion or intelligent disobedience)
Fine
Instantiated Meadows (STM)
PILOT
Motor Schema Instantiation
Status
Motor Schema Manager
Sensor Subsystem
Panic Shunts (Reflex Arcs)

Figure 3. Hierarchical Planner for AuRA

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 will consist of a list of parameters and modes of operation that determine the overall behavior of the robot. Additionally, a list of mission specifications and commands (subgoals) for the current task will be provided.

The mission planner, although a significant component of the overall architecture, has a relatively low priority for implementation at this time.

3.2 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 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 [9] and Giralt and Chatila [10,11,12]. Our work is distinguished from that which preceded it by the incorporation of extensions to include multiple terrain types, the use of specialized map production algorithms, the availability of several search strategies and the ability to easily embed perceptual 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 navigational computation. The necessary visual representations (feature map) for path execution and uncertainty management are tied to these polygonal ground plane projection models. This 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 2-D feature map can be viewed as a facet or appendage of the associated meadow map. Data pertaining to the distinctive features of terrain, obstacles, landmarks and other significa that are embedded within the meadow map constitute the 2-D feature map. The information stored here contains the attributes of the meadow map's vertices (1-D representations), lines (2-D image representations), and polygons and their associated obstacles or free space (3-D models).

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 visible from the robot's current position and orientation as well as those expected to be visible during the path traversal will be made current in STM.

Output of the navigator is directed to the pilot. This output consists of a point-to-point path and necessary parameters and modes 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 [6] for a complete description of the navigator and meadow-map representation.

3.3 Pilot

The pilot's function is to accept a point-to-point path specified by the navigator and provide the robot with suitable motor behaviors that will lead to its successful traversal. The pilot accomplishes this by selecting appropriate motor schemas from a repertoire of available behaviors (based on the current long-term memory context) and 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 and an alternate path is computed by the pilot, 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 will be provided by the mission planner, navigator and LTM. The new commands issued by the pilot will 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.

3.4 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 [13,14] 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. Figure 5 shows a simulation of a field produced by the instantiation of several independent motor schemas. 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 fields representations it uses, and the underlying motivation for its use are presented in [7].

3.5 Navigation Scenario

Perhaps the best way to convey the navigational process within AuRA is by example. Fig. 4a represents an LTM meadow-map model of the area outside the Graduate Research Center at UMASS. Embedded within this map, (although not visible in the figure), is additional data regarding landmarks, building

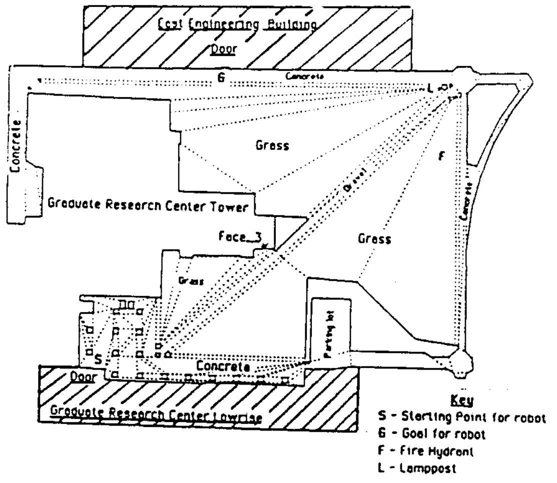


Figure 4a. Outdoor Meadow Map

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

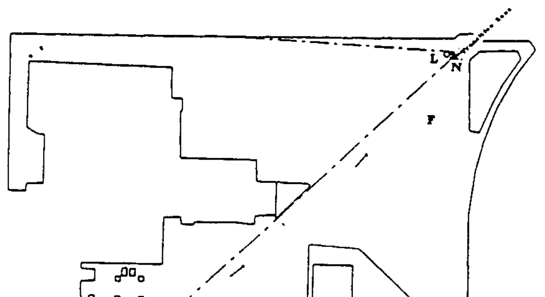


Figure 4b. Global path constructed by navigator

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

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, sets the traversability factors for the grassy regions to IMPASSABLE, locates the fact that Prof. X's office is in the East Engineering (EE) building, determines 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 invokes 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 for the purposes of this paper.

The navigator, based on the instructions from the mission planner, determines a global path that satisfies these goals using an A* search algorithm through the meadow boundaries (fig. 4b – see [6] for 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 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 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.

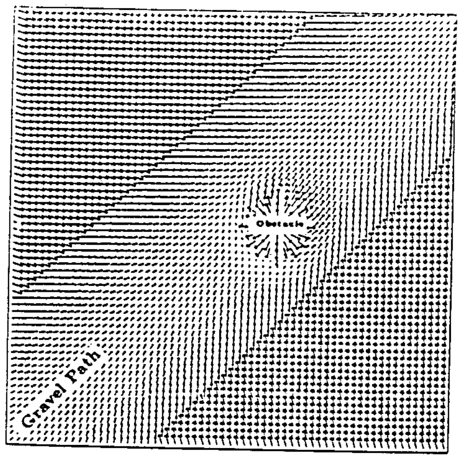
From this information, the pilot determines that appropriate behaviors for this particular leg (travel across the gravel path) include:

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**

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

(a)



(b)

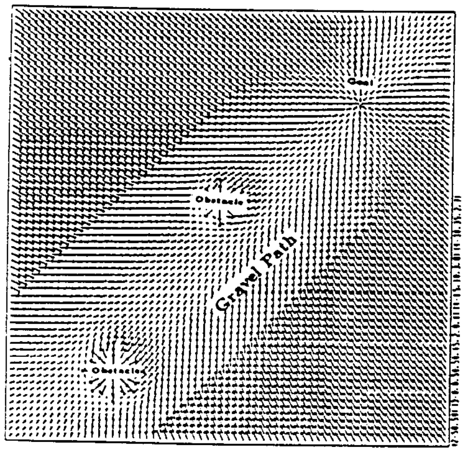


Figure 5. Potential fields produced during leg traversal
The arrows represent the desired velocity vectors that constrain the robot's motion.
a) Before the goal is identified, the **move-ahead** and **stay-on-path** SIs conduct the robot on its way. A single **obstacle** SI is present.
b) After the goal is identified, the **move-to-goal** SI replaces the **move-ahead** SI. Two **obstacle** SIs are shown as the goal is approached.

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 moving the vehicle towards the center of the path (fig. 5a).

We will define a schema instantiation (SI) to be the activity of applying a general class of schemas to a specific case [7,33]. The lamppost at the end of the path results in the creation of a **find-landmark** schema(s) 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 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 error 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 **find-landmark**(GRC_TOWER(face_3)) 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. Simply put, 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 5 shows a potential field simulation representative of the robot traversing an obstacle studded path as above. More details regarding the interaction and operation of the motor schemas in AuRA can be found in [7].

4. VISIONS and AuRA

Scene interpretation has long been a primary research effort within the VISIONS group at the University of Massachusetts. A considerable literature exists describing the progress to date [1,2,23,24,29,31]. The remainder of this section will first describe briefly 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 and computationally expensive process. Work is underway, however, to provide parallel hardware (the UMASS Image Understanding Architecture [29,30]) 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.

4.1 The Schema System

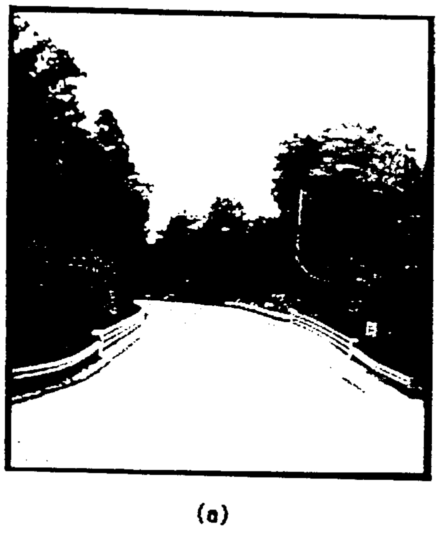
The VISIONS schema system accepts an image as input and produces a labeled interpretation of the observed environmental objects (fig. 6) and, to the degree possible, a 3-D representation of the environment. There are 3 levels of processing available utilizing both bottom-up and top-down processing (fig. 7). Taking a bottom-up view first, the low-level processes operate on pixel level data producing an intermediate symbolic representation of lines, surface and volume tokens. At the highest level, schema processes exist which interpret and collect the intermediate representations into labeled objects.

If no top-down guidance was 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. 29,31] for a better understanding of its operation.

4.2 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. 6). These efforts have been predominantly concerned with full scene labelings with few specific expectations established for the particular image or environment in question other than it being a house or road scene (i.e. there is no world map of the domain).



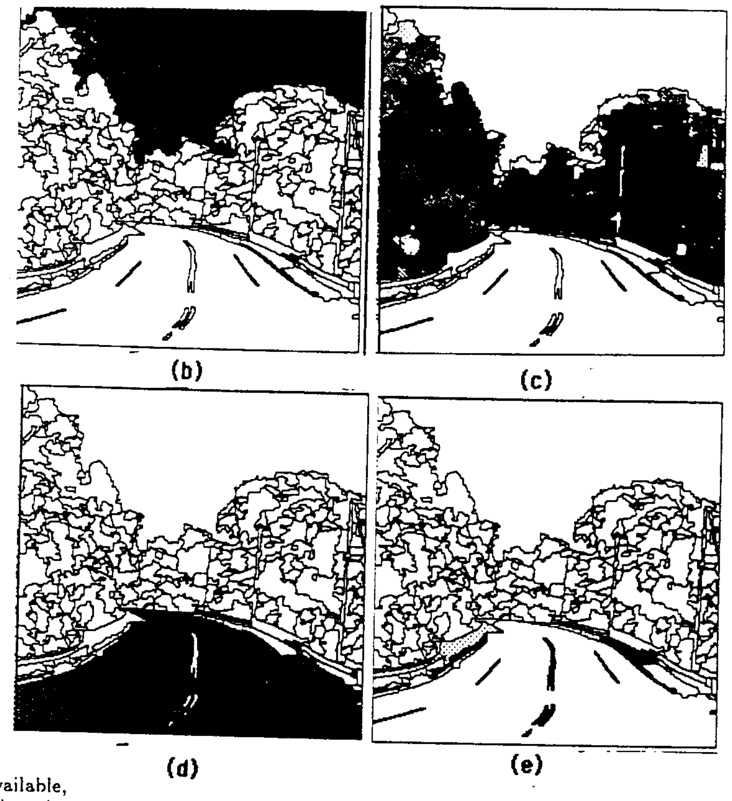


Figure 6. Schema-based Scene Interpretation
Initial Hypotheses for road scene
a) Original Image
b) Sky region
c) Foliage
d) Road
e) Road lines

If *a priori* knowledge of a specific environment is available, it can guide the posting of schema hypotheses, 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 the other algorithms described in this paper (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 be provided as well, 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 reinvoked to enable the robot to regain its bearings relative to the modeled world.

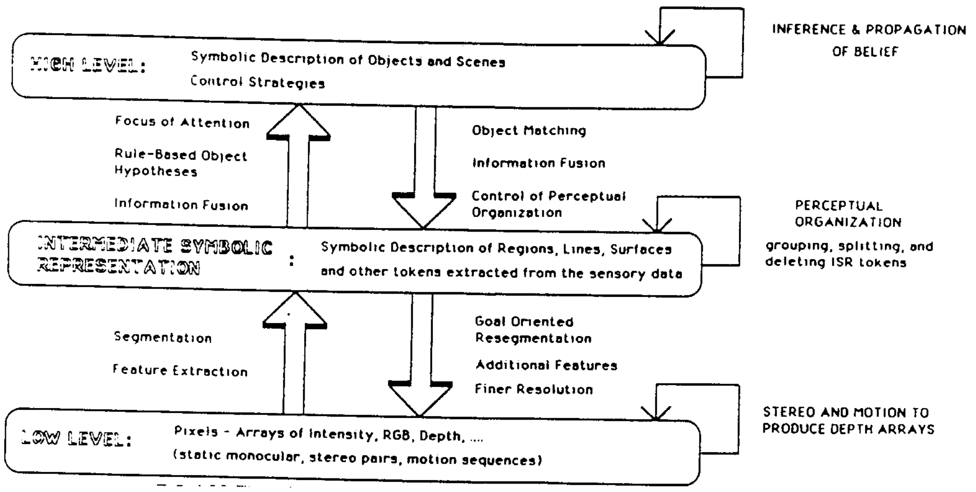
5. Modular Vision Algorithms

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. AuRA should be able 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. The next stage will be to dedicate specific processors for each algorithm to improve performance and then to eventually distribute the load 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



HIGH LEVEL:
Symbolic Description of Objects and Scenes
Control Strategies
INFERENCE & PROPAGATION OF BELIEF
Focus of Attention
Rule-Based Object Hypotheses
Information Fusion
Object Matching
Information Fusion
Control of Perceptual Organization
PERCEPTUAL ORGANIZATION
grouping, splitting, and deleting ISR tokens
INTERMEDIATE SYMBOLIC REPRESENTATION:
Symbolic Description of Regions, Lines, Surfaces and other tokens extracted from the sensory data
Segmentation
Feature Extraction
Goal Oriented Resegmentation
Additional Features
Finer Resolution
STEREO AND MOTION TO PRODUCE DEPTH ARRAYS
LOW LEVEL:
Pixels - Arrays of Intensity, RGB, Depth, ...
(static monocular, stereo pairs, motion sequences)

Figure 7. Multiple Levels of Representation and Processing in VISIONS

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 extensions that may be needed by new algorithms.

A common thread running through 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., which can be 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 or are being developed for use within AuRA. The emphasis will be on visual processing, but some work already has been accomplished using ultrasonic data as well. The imaginative reader will undoubtedly think of other approaches and other sensors that can be used within a system such as AuRA. The strategies described below are not exhaustive, but rather they represent the current initial elements being introduced by VISIONS researchers for use within this framework.

5.1 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.

5.1.1 Fast Line Finder (FLF)

A fast line finder based on Burns' algorithm [15] has been developed by Kahn, Kitchen and Riseman. 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 simplifying 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 sped up further through the use of a conventional pipeline processor which supports a look-up table and convolution processing.

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.

A 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 are described in the remainder of this paragraph. These features include the ability to scope the image (i.e. perform line extraction on a subwindow of the image). If the robot has an approximate knowledge of the position of the line feature being sought, (derived from LTM, the spatial error map and/or previous images), substantial processing reductions can be 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 can be 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 with 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 8a is an image of a sidewalk scene. Figure 8b shows the results of the FLF using the default bucket orientation for the entire image, and fig. 8c shows the results with the buckets tuned and scoped to the anticipated road edge based upon the internal model of the vehicle position and orientation, while fig. 8d 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.

5.1.2 Path Following

Using line following to extract path boundaries requires the grouping of resultant FLF line fragments into a single line representing each path edge. No effort is being 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 [4]): 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 error 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 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 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. 9a). 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. 9b-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 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. 9d). 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

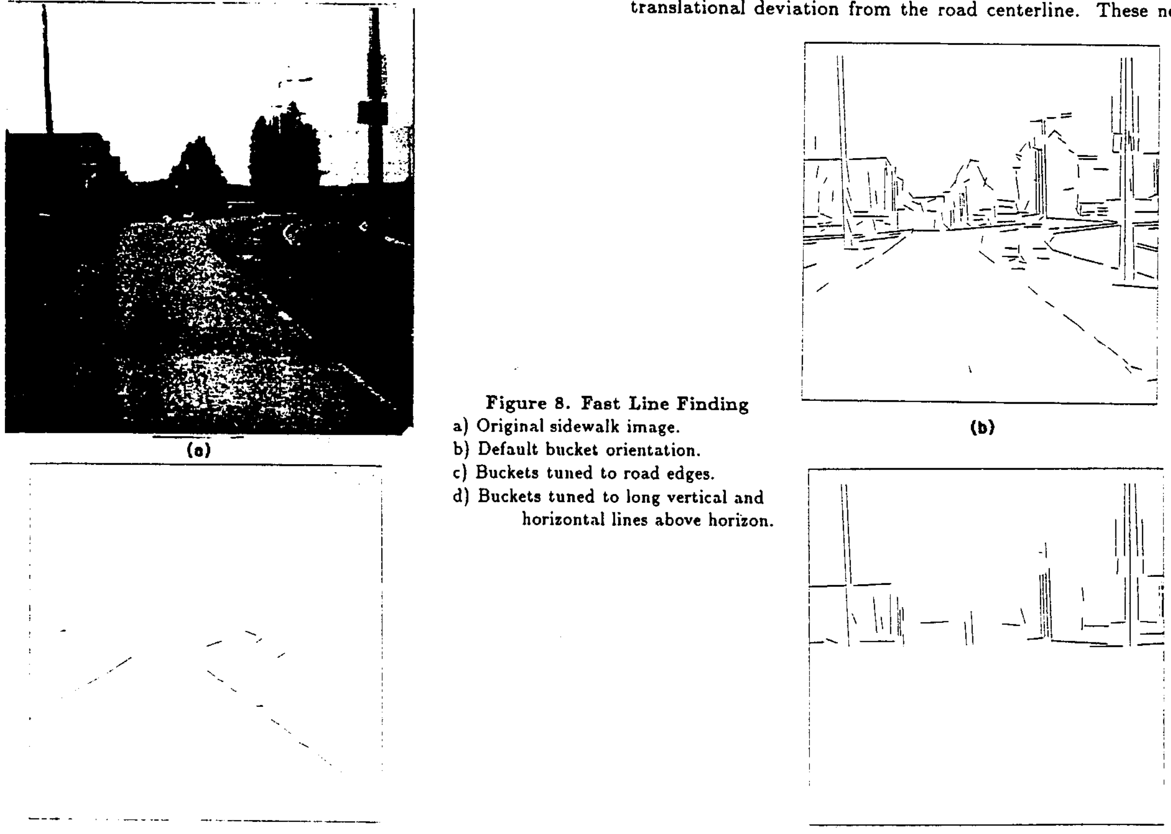


Figure 8. Fast Line Finding
a) Original sidewalk image.
b) Default bucket orientation.
c) Buckets tuned to road edges.
d) Buckets tuned to long vertical and horizontal lines above horizon.

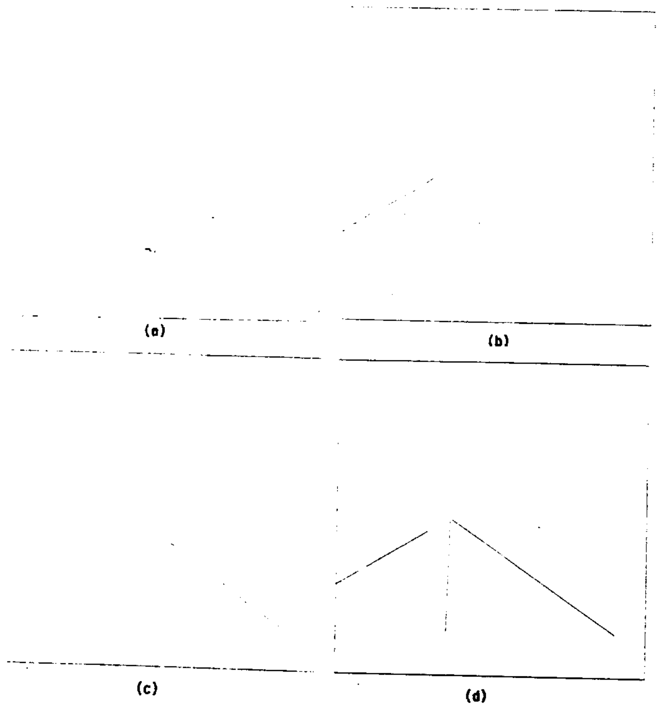


Figure 9. Path finding using FLF

a) Output of FLF when run on image 11a.
(tuned buckets - same as 11c.)
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.

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.

The robot has already been 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. The 512 by 512 image digitized on a Gould IP8500 is averaged to 256 by 256 before line extraction. This time will be reduced by using pipelined hardware available on the digitizer. The vehicle servos on the computed center line position, correcting both orientation and translational drift as it proceeds.

5.1.3 Landmark Identification through Line Finding

Vehicle localization can be 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. We do not propose that lines are the only mechanism for localization, but should serve in conjunction with other relevant algorithms. In the role of a confirmation mechanism, or

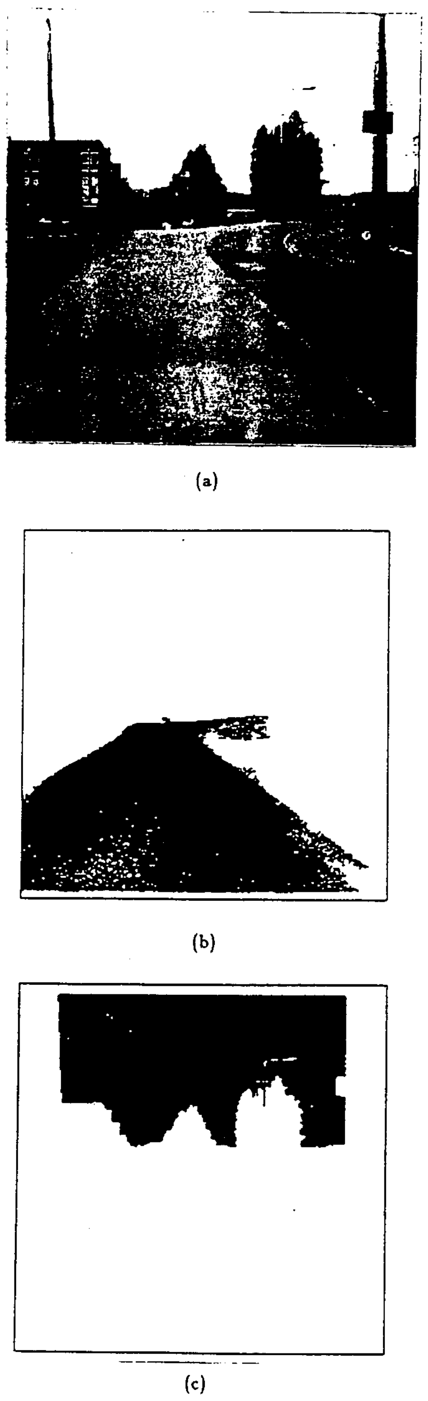


Figure 10. Road extraction via region segmentation
a) Sidewalk image.
b) Resultant extracted image representing road.
c) Resultant extracted image representing central portion of sky.

for tracking 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 as well, 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 that is suitable for this application. Edges of buildings, telephone poles, lampposts or doorframes can be tracked using the line finder. Figure 8d shows the result of running the FLF on image fig. 8a 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 error map and *a priori* knowledge of the global coordinates and dimensions of the feature in question (from LTM), 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 can be used to constrain the positional error of the vehicle by backprojecting the 2-D data to 3-D world coordinates when combined with the knowledge of the height of the feature.

5.2 Fast Region Segmenter (FRS)

FRS is a region extraction algorithm developed 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. The input image used can be an intensity image, a gradient image, a color plane, etc. This input image can be scoped (windowed) as is the case with the FLF. The look-up table maps ranges of pixel values to specific region labels and, as before, can be loaded in a top-down manner based upon stored knowledge or sampled data from previous images. Available knowledge can be used to define expected ranges of spectral attributes of interesting objects (fig. 7,10,11). 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. This algorithm has been motivated by histogram-based segmentation algorithms [34], but achieves great simplification via constraints from stored object knowledge in LTM or the result of processing previous frames, to define the look-up table ranges for objects in the next frame.

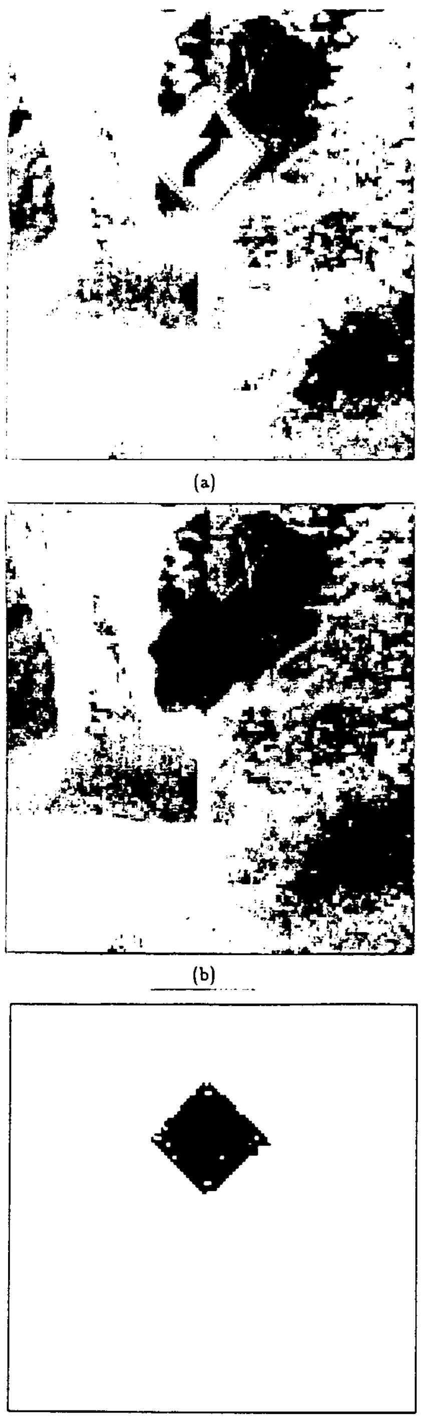
The speed of this algorithm arises from the use of the look-up table to provide a quick mapping to the image. The connected components routine is then run on a restricted portion of the image selected through the use of top-down map constraints.

This segmentation can be used for road extraction as in [16,17]. Preliminary experimentation using intensity images can be seen in figure 10. Fig. 10a shows the original image and fig. 10b the region extracted representing the road. The statistics collected for the road region are then used for providing the expectations (feedforward) for the next image in the sequence [as in 16].

Landmark extraction can be 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 11.

5.3 Depth from Motion

Passive navigation by the determination of the position of environmental objects through vision is an important sensor strat-


(a)

(b)

(c)

Figure 11. Landmark identification via region segmentation
a) Original sign image (combined RGB intensity).
b) Blue plane of (a) chosen for analysis due to the spectral data of anticipated landmark (available from LTM).
c) Extracted region representing sign.

egy for AuRA. The motion research group within VISIONS has long explored the extraction of depth from motion [18,19,20]. A more recent algorithm, developed by Bharwani, Riseman and Hanson, uses a sequence of frames 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.

5.3.1 Algorithm

A brief sketch of the multiple frame depth-from-motion algorithm developed by Bharwani, Riseman, and Hanson follows. The reader is referred to [21,28] 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. (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 in sec. 5.3.2). 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 points of high curvature and contrast in the image that are unlikely to correlate well with false match points in future frames. 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 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 computation and providing refinements of the original depth estimates. Work by Snyder [22], addressing the limits of uncertainty in this type of motion, is fundamental to efficient use of previous correct correspondences in constraining 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) can be 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, can be used to extract objects from the environment. 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. The specific application of this technique to both of these cases appears below.

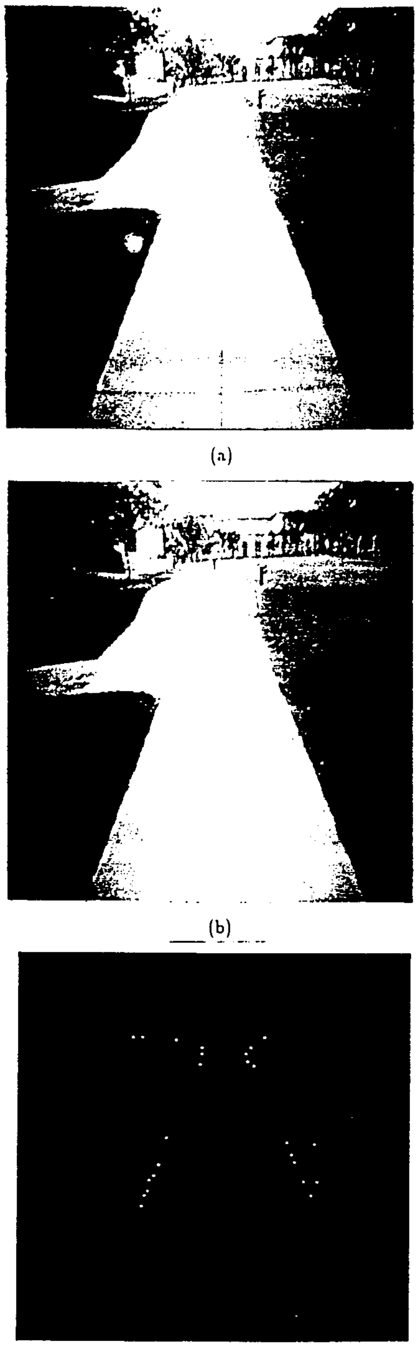


Figure 12. Depth from Motion Results
a-b) Two image sequence (distance traveled is 0.5 m)
c) Interest points tracked (from image 1)

The steps are at a distance of 13.5 meters in frame (a). The results for the interest points associated with the steps are within 10 percent of the ground truth.
(from [28]) (image sequence from CMU terregator).

5.3.2 Applications

A primary goal of the depth-from-motion algorithm is to be able to provide information about the distance of an object lying in the path of the robot. In obstacle avoidance applications, computational requirements can be 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 12 illustrates some preliminary results of using the depth-from-motion algorithm for obstacle avoidance. It can be seen that the results, (accuracy within 10% in the recovery of obstacle depth), indeed look promising although continual research effort will be made to validate the initial results. 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, 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 can be achieved by applying the correlation matching process to points near and above the horizon, i.e. distant (hence relatively unmoving) features that can be registered from frame to frame. The algorithm is quite sensitive to rotation so every effort should be made to minimize or eliminate any pitch, roll, or yaw movements of the camera relative to the scene.

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. The approximate distance of a known landmark to the vehicle in a restricted portion of the overall image restricts the computation required substantially. 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 of 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. There is also a bottom-up strategy - obtain the depth of a subset of points that are on an anticipated object and the expected surface orientation from STM and search the depth points returned for a match. By overlapping region and line information, signs, buildings, telephone poles, etc., could be located within the robot's frame of reference.

5.4 Interest Operators

Interest operators are used in computer vision to pick out pixels associated with regions of high curvature and contrast. The Moravec operator [25] and the Kitchen-Rosenfeld gray-level corner detection interest operator [26] are two well-known examples. The depth from motion algorithm, described in section 5.3, uses an interest operator, (currently Moravec's), to determine the points on which to run the correspondence algorithms from frame to frame.

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. When combined with knowledge from the robot's positional error map and available data from LTM, this method can be used in restricted circumstances to confirm the position of a real corner as predicted by the line-finding intersection method (sec. 5.1.3). A succinct description of the Moravec operator appears in [27] 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 would indicate a potential obstacle. This low-cost focus of attention mechanism permits the concentration of higher-cost computational effort in such likely sitations.

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 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.

The vision algorithms to be 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 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 will 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.

Ongoing and future work includes the refinement of the individual perception algorithms used, increased real-time mobile robot experiments as parallel hardware is integrated into the system, and addressing the considerable system integration problems involved with the interfacing of the individual components being developed.

Acknowledgments

The authors would like to thank the members of the Laboratory for Perceptual Robotics and VISIONS groups for their assistance in the development of software and preparation of this paper. Special thanks are due to Raj Bharwani, Phil Kahn and Les Kitchen.

References

1. 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.
2. Weymouth, T., "Using Object Descriptions in a Schema Network for Machine Vision", *Ph.D. Dissertation, COINS Tech. Rep.* 86-24, Univ. of Massachusetts, Amherst, May 1986.

3. Shen, H. and Signarowski, G., "A Knowledge Representation for Roving Robots", IEEE Second Conf. on Artif. Intel. App., pp. 621-628, December 1985.
4. 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.
5. Arkin, R., "Internal Control of a Robot: An Endocrine Analogy", unpublished paper, Dec. 1984.
6. Arkin, R., "Path Planning for a Vision-based Mobile Robot", to appear in MOBILE ROBOTS - *SPIE Proc. Vol. 727*, 1986. Also COINS Technical Report 86-48, Dept. of Computer and Information Science, Univ. of Mass., 1986.
7. Arkin R., "Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior", COINS Technical Report, in preparation, Dept. of Computer and Information Science, Univ. of Mass., 1986.
8. Arkin, R.C., Ph.D. Proposal, Computer and Information Science Department, University of Massachusetts, Spring 1986.
9. Crowley, J., "Navigation for an Intelligent Mobile Robot", CMU Robotics Institute Tech. Rep., CMU-RI-TR-84-18, 1984.
10. 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.
11. Giralt, G., "Mobile Robots", Artificial Intelligence and Robotics, ed. Brady, Gerhardt, and Davidson, Springer-Verlag, NATO ASI series, pp. 365-394, 1984.
12. 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.
13. Krogh, B., "A Generalized Potential Field Approach to Obstacle Avoidance Control", *SME - RI Technical Paper* MS84-484, 1984.
14. Krogh, B. and Thorpe, C., "Integrated Path Planning and Dynamic Steering Control for Autonomous Vehicles", *IEEE Conf. on Robotics and Auto.*, 1986 pp. 1664-1669.
15. Burns, J., Hanson, A., Riseman, E., "Extracting Straight Lines", Proc. IEEE 7th Int. Conf. on Pattern Recognition, Montreal, Can, pp. 482-485. 1984.
16. Thorpe, C. and Kanade, T., "Vision and Navigation for the CMU Navlab", to appear in MOBILE ROBOTS - *SPIE Proc. Vol. 727*, 1986.
17. Tou, J., " Software Architecture of Machine Vision for Roving Robots", Optical Engineering, Vol. 25-3, pp. 428-435, March 1986.
18. 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.
19. 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.
20. Anandan, P., "Measuring Visual Motion from Image Sequences, Ph.D. Dissertation, University of Massachusetts at Amherst, Jan. 1987.
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. 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.
23. 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.
24. 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.
25. Moravec, H., *Robot Rover Visual Navigation*, UMI Press, 1981.
26. Kitchen, L. and Rosenfeld, I., "Grey-level Corner Detection", Technical Report 887, Comp. Sci. Ctr., U. Maryland, College Park, Md, 1980.
27. Ballard, D. and Brown, C., COMPUTER VISION, Prentice-Hall, 1982, pp. 69-70.
28. 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.
29. 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.
30. 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.
31. Draper, B., Hanson, A. and Riseman ,E., "A Software Environment for High Level Vision", COINS Tech. Report (in preparation), Dept. of Computer and Information Science, Univ. of Mass., 1986.
32. 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.
33. Arbib, M., "Perceptual Structures and Distributed Motor Control", HANDBOOK OF PHYSIOLOGY - The Nervous System II (V. Brooks ed.), pp. 1449-1465,1981.
34. 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.