Path planning for a vision-based autonomous robot

Ronald C. Arkin

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. To provide a testbed for many of the algorithms developed within this framework, a mobile robot has been acquired.

A multi-level representation and the accompanying architecture used to support multi-sensor navigation (predominantly visual) are described. A hybrid vertex-graph free-space representation (meadow map) based upon the decomposition of free space into convex regions capable for use in both indoor and limited outdoor navigation is discussed. Of particular interest is the capability to handle multiple terrain types.

A hierarchical path planner that utilizes the data available in the above representational scheme is described. An overview of the UMASS mobile robot architecture (AuRA) is presented.

1. Introduction

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

HARV is a mobile robot (a Denning Research Vehicle) equipped with video and ultrasonic sensors (fig. 1). It is to be operated in two distinctly different domains:

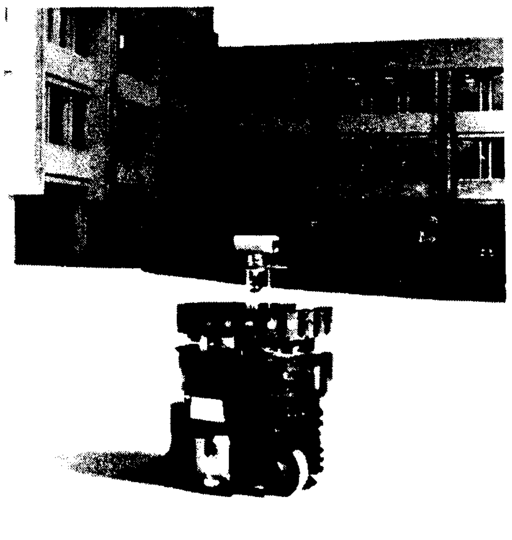


Figure 1.

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.

- The confines of a research building: including halls, foyers, and large rooms.
- The grounds of the UMASS campus; including sidewalks, building entrances, parking lots, and grassy areas.

Thus, any architecture and representation scheme to be used must be sufficiently general to handle both the indoor and outdoor case.

This paper is concerned primarily with path construction and navigation in a partially modeled environment. The UMASS autonomous robot architecture (AuRA) incorporates a hierarchical planner (fig. 2) consisting of a pilot, navigator, mission planner and motor schema manager (the execution arm of the pilot). This paper addresses specifically the role and operation of the navigator and its associated world model representations upon which the navigator bases its decisions.

In the remainder of this introduction, a brief review of relevant work will be presented followed by a description of the UMASS environment. Section 2 will describe the representation used by the navigator and the software that builds this map. The operation of the navigator will be described in Section 3. The extension of the representation to include diverse terrain types will be related in section 4. A brief overview of the entire AuRA system architecture will appear in section 5. A summary and conclusions will complete this report.

1.1 Previous work

Early in the days of artificial intelligence, Amarel showed that a good representation is essential for the efficient solution of a problem[1]. Many different tacks have been taken by various researchers regarding appropriate navigational representations. These include pure free-space representations[2,3,4], vertex graphs, potential fields[5], regular grids[6,7,8,9], quadtree[10] and automaton representations[11]. A hybrid representation involving both vertex graph and free-space representation techniques has also been used[12,13,14]. Each approach has its own strengths and weaknesses[15]. In several instances[16,17] multi-level representations are used to exploit the best of several different representations.

The UMASS VISIONS system[18,19,20], a system used for the interpretation of natural scenes, also encompasses a multi-level representation scheme. Knowledge sources provide hypotheses and instantiate perceptual schemas based on top-down knowledge and video sensor input. This system will ultimately be the gateway for visual and other sensor data entering into the robot. Previously, VISIONS has not maintained representations specifically addressing navigational path planning. While provision is made for 3D representations of objects and their 2D projections in the vertical plane, no express representation of horizontal projections to the ground plane has been present. The system proposed in section 5 extends and complements the VISIONS system by adding representation levels that specifically deal with the issues involved in navigational path planning.

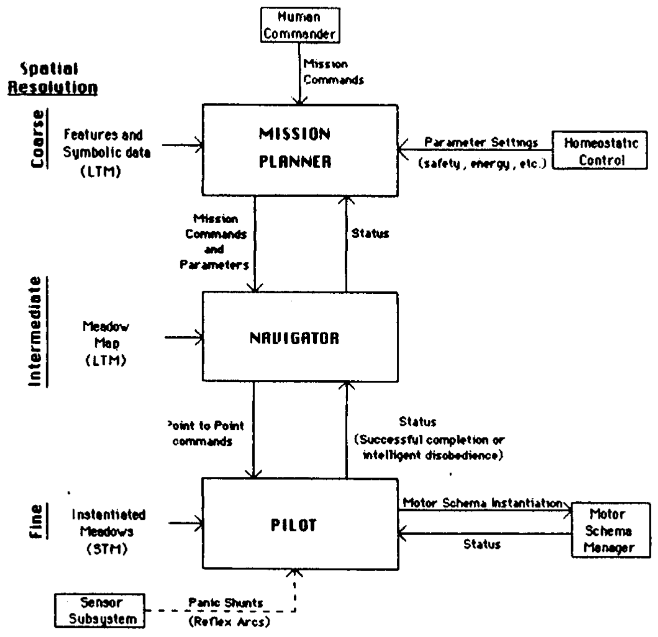

Human Commander
Mission Commands
Spatial Resolution
Coarse
Features and Symbolic data (LTM)
MISSION PLANNER
Parameter Settings (safety, energy, etc.)
Homeostatic Control
Mission Commands and Parameters
Status
Intermediate
Meadow Map (LTM)
NAVIGATOR
Point to Point commands
Status (Successful completion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

Figure 2. Hierarchical Planner for UMASS AuRA

1.2 UMASS environment

The UMASS Denning Research Vehicle (fig. 1) is a mobile robot manufactured by Denning Mobile robotics. It is equipped with 24 ultrasonic sensors and shaft encoders for both the steering and drive motors. A single video camera (the VISIONS system uses monocular images) is mounted on the vehicle and connected to a Gould digitizer. Although some of the sensor preprocessing (ultrasonic and encoder) is done on board the vehicle, the bulk of the software runs on the Computer and Information Science department's VAXen. It is anticipated that the processing load will be distributed over multiple processors by taking advantage of the VAXcluster architecture connecting the department's numerous computers. The development of the UMASS Image Understanding Architecture and the CAAPP (Content Addressable Array Parallel Processor[23]) is expected to enhance the vision processing performance when it becomes available. In addition, the robot is to be equipped with a UHF TV transmitter enabling remote operation.

The two locales in which the robot is to be operated include both indoor and outdoor environs. The first is within the confines of our building, the Lederle Graduate Research Center (GRC). The navigator assumes significant but incomplete *a priori* knowledge of the world. Blueprints for the building constitute the basis on which the initial indoor representation is built. A digitizer is used to enter the relevant map features. It is conceivable that this map could be acquired dynamically by interaction with the environment in a manner akin to HILARE[24] or Neptune[25]. Environmental acquisition via learning will not be addressed in the near future in our work, although other UMASS researchers may be involved in this research.

The second locale is the grounds surrounding the GRC. The model is derived largely from a map made by aerial photography. Multiple terrain types are present including concrete sidewalks, grassy regions and a gravel path, all of which are available for navigation by the robot.

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

The navigator serves a role analogous to the navigator in a road rally: to provide a piecewise linear path to the vehicle pilot (driver) for execution. Instructions might be: proceed 120 m and turn right 85 degrees. The navigator operates from a relatively static map and is not concerned with unrepresented obstacles unless the pilot expressly requests an alternate route.

The pilot is considerably more short-sighted. It is concerned only with satisfying one subgoal from the navigator at a time (although future subgoals may affect its decisions). The pilot additionally accepts constraints from the navigator such as criteria for failure to attain a subgoal. If any of those criteria are met, the navigator is informed and navigational replanning initiated. Local alterations in the route specified by the navigator can be made without navigator reinvocation as long as the alterations fall within these limits. The pilot does not utilize the same representation the navigator does, as it assumes that the navigator has correctly produced a path that avoids any modeled obstacle. Consequently the pilot is concerned with avoiding unmodeled obstacles (subject to certain constraints). Other work[16,21,22] describes the use of similar hierarchical planning systems.

Section 3 mentions how the pilot will be schema based; choosing motor actions from a library of appropriate behaviors (schemas) that are relevant to satisfying the navigator's subgoals. The scope of this paper, however, is primarily limited to navigational planning issues and the schema based pilot and its representations will be described in a separate paper.

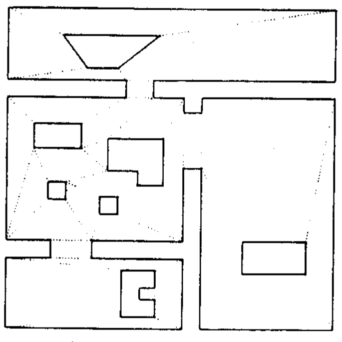
2. Representation

To address the issues of path planning, a static representation and a dynamic representation have been developed. The static form, or long term memory (LTM), is where all *a priori* knowledge is embedded. Although a variety of sensor interpretation strategies access data stored in LTM, the navigator is the prime consumer of this representation within the confines of the hierarchical planner.

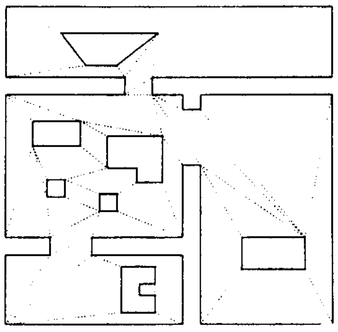
The dynamic representation or short term memory (STM), is a layered representation consisting of the robot's current perception of the world based on a long term memory context. Of the planner components, the pilot and motor schema manager are the principal users of this data. Portions of LTM are instantiated in STM based upon the robot's current position and the navigator's instructions. As the robot traverses this path, sensor data (visual and ultrasonic) are incorporated to build up a dynamic model of the perceived world. This is then used to direct the pilot to appropriate action when the path is blocked or other alternate actions are in order. Additionally, vehicle localization (increasing positional certainty) can be guided by available landmarks found in these regions of visibility. Further discussion of the details of short term memory for navigation will be deferred to a separate paper.

2.1 Long term memory - meadow map

The principal representational form used by the navigator is a "meadow map" and is based on previous work by Crowley[26] and Chatila and Laumond[12]. It models free space as a collection



(a)



(b)

Figure 3. Different Convex Decompositions

of convex polygons. Diagrams depicting an indoor scene and an outdoor scene appear in figures 3 and 4. The rationale for using convex regions is that a line between any point within one convex region to any other point within that same region is guaranteed to be free of collisions with all known obstacles. Thus the global path planning problem simplifies to finding an appropriate sequence of convex region traversals. (Actually finding a "good" path is more difficult - see sec. 3.2 and 4.2).

What distinguishes this representational form from the efforts that preceded it lies in its ability to embed both terrain and sensor data and its extension to include diverse terrain types. Convex regions were chosen over a regular grid approach due to their ability to avoid digitization bias, a smaller search space and a significant reduction in memory requirements. Voronoi diagrams were avoided due to their inability to relate landmark and terrain data as readily and their perceived limitations on flexibility of path construction when compared to this strategy.

2.1.1 Meadow map construction

The algorithm for the construction of the long term memory meadow map representation is described in figure 5 and consists of the following phases: initialization, main map building and clean-up.

Initialization

In the initial phase, a series of vertices in global coordinates describing the maximum reaches of robot navigation are accepted. In the case of the interior of a building, this would be the bounding walls. In more open terrain, it might be boundaries of limiting paths or an imaginary polygon bounding the traversable region. There are no restrictions on the shape of the bounding region and obstacles other than that they be represented by a series of straight line segments. Curves must be converted to piecewise linear segment approximations. The raw data is obtained from a map or blueprint of the region and the use of a bitpad digitizer.

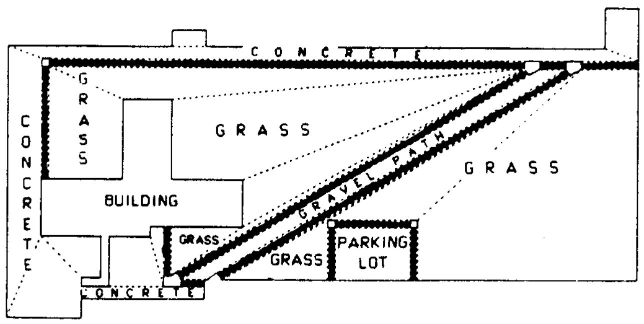


Figure 4. Multi-terrain Map

After the actual coordinates of the bounding region are accepted, the region is shrunk in the configuration space manner[27]. This enables the robot to be treated as a point thereafter for path planning purposes. A specialized shrinking technique[28] is available for highly cluttered environments.

Known obstacles that are present in the environment (pillars, telephone poles, ... any static impediment to motion) are then added. These are grown in the C-space style for the same reason that the bounding region was shrunk. Any obstacles whose growth results in a collision with the bounding region, are merged into the border as they can no longer be completely circumnavigated.

Finally, the obstacles are attached to the bounding region. The obstacle vertex that is closest to a border vertex is attached to the border by two passable links; one going out to the obstacle and the other returning. This is repeated until all the obstacles are connected. In essence, a single region is produced which includes all grown obstacles (fig. 6).

Main map building algorithm

This portion of the algorithm decomposes the region produced in the initialization step by recursively splitting the area until all resulting regions are convex. Upon receipt of the initial region it is checked for convexity. If the region is convex, this portion of the procedure terminates. If it isn't, a concave angle is selected from those available in the region. There is guaranteed to be at least one concave angle or the region would be convex. Three options are available for selection: the least concave, the

FREE SPACE MAP BUILDING ALGORITHM

Initialization

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

Main map building algorithm

```
IF region is convex
    done
ELSE
    Find (most,least,first) concave angle
    Connect it to (most opposite,leftmost,rightmost) clear vertex
    Apply main map-building algorithm recursively
       to the two resulting regions
ENDIF
```

Clean-up

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

Figure 5.

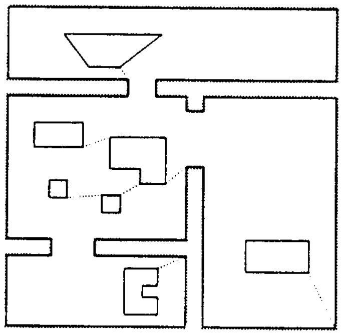


Figure 6. Attached Obstacles
The grown obstacles have been attached to the surrounding shrunken border. The resulting single region is now ready to undergo convex decomposition.

most concave, or the first concave angle found can be chosen. After an appropriate concave vertex is selected, the second (victim) vertex for splitting the region in two must be chosen. Three possibilities again exist: the leftmost clear vertex, the rightmost clear vertex, or the most nearly opposite vertex (right of center). A connecting edge, labeled as passable, is completed between the concave angle and its victim and the initial region is split in two. The algorithm is then applied recursively to each of the resulting two newly formed regions. Thus a graph of convex regions and their traversability (fig. 3) is produced, facilitating search during the path finding process. This decomposition continues until all of the regions produced are convex.

Considerable experimentation was carried out, trying to determine which of the concave selection modes and victim selection modes (of the 9 possibilities) yields the "best" results. Figures 3a-b show 2 different decompositions on the same region. The decomposition results were evaluated on the basis of the quality of the paths that they produced.

How to define what constitutes the best results is nebulous. Shortest Euclidean distance as a path length metric, (which might appear to be the most obvious choice), may result in significant problems with the clipping of modeled obstacles during travel due to the inherent positional uncertainty found in the mobile robotics domain. Fewest overall legs in a particular path is another possible choice.

When the path search was restricted to the midpoints of the bounding regions, (A^*-1, see sec. 3.1), the experimentation indicated that the results obtained were more strongly influenced by the shape of the initial bounding region and the choice of start and goal points of a particular path than by any predetermined choice of vertex selection modes for decomposition. That is not to say the choice of decomposition method did not produce significantly different paths in certain circumstances for the midpoint search; rather information that is dependent on a particular initial region and the most likely paths to be taken within that region should appropriately influence the vertex selection process. An expanded search (A^*-3) through 3 points on each passable meadow boundary (the midpoint and one point near each end) largely decouples the dependency of the path cost on the decomposition method. Consequently, it becomes less significant which mapbuilding strategy is chosen if this more costly search methodology is used. Map building times ranging from 3 to 12 CPU seconds were observed in the non-optimized version of this algorithm for the indoor scenario.

Clean-up

The resulting number of convex regions produced by the main decomposition algorithm is not always minimal. There may be some regions which can be merged together that still result in a convex region. In some instances there are relatively few, in others it is a bit more common. During this phase, a pass is made on the convex region list that merges together any regions that would result in a single convex region. The mapbuilder process then invokes the feature editor.

It should be recalled that the algorithm thus far described is for the simplest case, involving only one terrain type. Multiple terrain types require additional processing and will be described in section 4.

2.1.2 Feature editor

A major advantage this representation affords is the ease in which additional representations of objects, landmarks, terrain features, etc. can be embedded. In an experimental system this is very important. The level of granularity can vary; for obstacles or walls, full 3D models of the entire object, 2D planar representations of surfaces (pop-up views) associated with sides (edges), or even simplistic line models for individual corners, projected up into the image plane, can be readily embedded via pointers. For terrain, entire region characteristics (traversability data, statistical error data, data for guiding visual region segmentation algorithms, etc.) can be tied to the free space regions. Individual meadows can have topographical models (for non-planar surfaces) which represent contours in any way the designer of such a representation chooses. This flexibility for adding world representations in an object-oriented manner is one of the prime factors in the choice of the meadow map scheme over other alternatives.

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

This data can be acquired via sensing as well. For example, in the case of visual data for region segmentation, by pointing the robot's camera in the direction of a known region type (e.g. grass), then acquiring the appropriate statistics through the video digitizer and a histogram process, the robot can store the statistical features for a particular terrain type on a per run basis. This avoids the inflexibility that would be present if the statistics had to computed once for all weather and seasonal conditions. The result should be more robust visual segmentation. The robot can be trained quickly and efficiently to recognize certain terrain features.

The initial direction for features in our system include (but are not limited to) :

- **Terrain data**
  1. Traversability factor (ease of passage)
  2. Translational and rotational error data - (to guide the spatial error map for managing positional uncertainty)
  3. Data to guide visual region labeling
  4. Unmodeled obstacle density
- **Obstacle data**
  1. 2D pop-up views of sides
  2. Vertical edge data for particular vertices (corners of buildings, doorways, etc.)

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

3. Navigation (global path planning)

After the mapbuilder process terminates, the planner process is initiated. The high-level planner is hierarchical in design; consisting of a mission planner, navigator and pilot. The mission planner is delegated the responsibility of interpreting high level commands, determining the nature of the mission, setting criteria for mission, navigator and pilot failure, and setting appropriate navigator and pilot parameters. The mission planner, although part of the overall design, is not yet implemented, and has a relatively low priority.

The navigator accepts a start and goal point from the mission planner and using the global map built by the mapbuilder determines the "best" path to attain that goal. Just what constitutes "best" is determined by the mission planner. It might be the shortest, or the safest, or the fastest, or the least energy consuming path. In essence the mission planner determines which cost functions and heuristics the navigator will use in carrying out its role in the hierarchical planner. The remainder of this section will deal with how the navigator fulfills its responsibilities. But first, a brief discussion of the operation of the pilot.

The pilot is charged with implementing leg-by-leg the piecewise linear path that is output by the navigator. To do so the pilot chooses from a repertoire of available sensing strategies and motor behaviors (schemas) and passes them to the motor schema manager for instantiation. Distributed control and low-level planning occur within the confines of the motor schema manager during its attempt to satisfy the navigatior's requirements. As the robot proceeds, the sensors build up a model of the perceived world in short term memory that is used by the motor schemas and when the pilot recognizes that it cannot satisfy the navigator's goals without making changes in the specified path. If the changes required are too severe, (as specified by the higher levels of the planner), the navigator will be reinvoked and a new global path computed. If the deviations are within acceptable limits, the pilot and motor schema manager will, in a coordinated effort, attempt to bypass the obstacle or other problems. Additionally, the problem of robot localization is constantly addressed through the monitoring of short term memory and appropriate find landmark schemas. Multiple concurrent behaviors may be present during any leg, for example:

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

The control of the priorities of the behaviors, (e.g. when is it more important to follow the sidewalk than to avoid uncertain but possible obstacles) is dependent on the certainty associated with the STM representation and is controlled by the motor schema manager. A sensor independent short term memory representation will be the fusion point for all the different sensor modalities and strategies.

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

3.1 Search

The navigator's task is to search through the meadow map produced by the map builder and derive a good path for a specified start and goal. A reasonable but non-optimal solution is satisfactory for this domain. Even if an optimal path (by whatever definition) was attainable by the navigator, it would only be based on partial information (i.e. the modeled world). Since the robot's environment is subject to unmodeled and even moving obstacles, there is no *a priori* guarantee that any path produced by any navigator is truly optimal. Consequently, the ultimate goal of the navigator is to arrive at a "reasonable" path. Optimality is perhaps a misplaced notion in a dynamically changing world (without constant replanning), thus the value of spending high computational effort in ensuring absolute minimal costs in this mobile robot's domain is unjustified.

After stating that, let me contradict myself a bit by saying that the A^* algorithm is used with heuristics (at this point) that guarantee optimality. Two different search spaces are available for the search algorithm. The simplest and most efficient, A^*-1, is built from the midpoints of the bordering passable regions (as in [26]). The larger space, A^*-3, is derived from a triad of points on the bordering regions; the midpoint and two points near each end of the passable edge (separated from the end by a specified safety margin). Although computationally more expensive, the advantage of A^*-3 over A^*-1 lies in a significant decoupling of the path planning from the mapbuilding decomposition method. Finding the initial coarse path is a fairly rapid operation (less than 2 seconds for A^*-1; less than 7 seconds for A^*-3 for typical paths). It is guaranteed to be the best path available (subject to the cost function chosen) within the specified search space.

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

The simplest heuristic function is the shortest straight-line Euclidean distance on the plane assuming optimal terrain. Since the search space is relatively small, no effort has been placed into finding better heuristics. The cost function used takes into account the traversability factor of a given terrain type, the actual distance traversed, and can readily incorporate other factors such as threat measurement, topographical grades, unmodeled obstacle density (perhaps a function of time of day - (e.g. high between classes on a sidewalk, low otherwise)), and ease of localization (based on numbers of readily discernible landmarks within a given region).

In summary, the navigator algorithm (shown in fig. 7) accepts first two points from the mission planner. It then searches, using the A^* algorithm with a cost function based on terrain factors and traversability, the space of midpoints (A^*-1) or triads (A^*-3) of connecting adjacent passable meadows, outputting a coarse path consisting of a series of piecewise linear segments connecting the start, the edges of bordering meadows and the goal (fig. 9a). This approach expands fewer nodes than would a comparable pure vertex graph of the obstacle edges (and obviously much fewer than a regular grid). The pure vertex graph (although guaranteed to produce the shortest path) also suffers from an inability to readily produce safer paths from shorter paths, hence the strategy used here is considerably more versatile. A Voronoi diagram can readily produce safe paths, but also

```
PATH FINDING ALGORITHM
Accept start and goal from mission planner.
                    Search
Apply A* search algorithm through convex region connectors.
    (A* - 1: midpoint only)
    (A* - 3: midpoint + two points near endpoints of connectors)
Output Raw Path
            Path Improvement Techniques
If specified
    A. Tighten path by sliding towards side vertex
        by some given amount.
    B. Straighten path by removing any turns that are not essential
        for a clear traversal.
            (details for both parts A and B appear in fig. 8)
Return "reasonable" piecewise linear path through world model.
```

Figure 7.

```
PATH IMPROVEMENT ALGORITHM
(single terrain type)
Accept the coarse path from the search component of the navigator
                Tautness part
Accept safety margin (clearance from side)
Get first border midpoint of coarse path
DO WHILE Not at end of path
    compute length of path for three cases
        a. Midpoint unchanged
        b. Midpoint slid to right (maintaining safety margin)
        c. Midpoint slid to left (maintaining safety margin)
    choose lowest cost path from a, b or c.
    modify path if necessary and mark path node as moved
    Get next path node
ENDDO
                Straightness Part
Get start of path
DO WHILE not at end
    IF clear path is available to any path node ahead of current node
        delete all intervening nodes
    ENDIF
    Get next path node
ENDDO
                Clean up
Slide towards edges again as in tightening part above if path was
    straightened (only for path nodes still at midpoint)
Output refined path
```

Figure 8.

lacks the flexibility afforded by this representation to change its strategies (safe to short to fast) when deemed appropriate by the mission planner.

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

3.2 Path improvement strategy

Path improvement techniques are relatively common in the use of regular grids. Although the representation used in the UMASS AuRA long term memory is a meadow map and not a regular grid, the precedent of refining a coarse path into a better one exists. Thorpe[9] uses a relaxation based approach on a coarse grid while Mitchell and Keirsey[6] use a compensation technique to minimize inefficiency due to digitization bias.

The algorithm for path improvement for the simple single terrain type case is presented in figure 8. The "raw" path is first received from the planner. Beginning at the start path node and proceeding to the end node, each node on a passable meadow border is tested at three locations; slid all the way to the left (leaving a specified safety margin clearance), slid all the way to the right (minus safety margin), and unchanged at the middle. The lowest cost solution is chosen and the path modified accordingly. This can be visualized as pulling on the ends of the path thus tightening the path around the obstacles and walls. This is considerably less costly than a relaxation algorithm requiring multiple iterations over the entire path. (A limited relaxation algorithm involving only the transition zones is required for the multiple terrain case - see sec. 4). The A*-3 search method can bypass this initial tautness processing as its search strategy has already effectively accomplished it.

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

If the path is straightened, a better path may now be obtained by sliding some of the previously unmoved path nodes. Before exiting, the algorithm checks all these unmoved nodes to see if a lower cost path can be obtained by sliding them along their meadow boundaries The resulting refined path is output from the navigator and stored in short term memory for use by the pilot.

3.3 Results

See figures 9-11 for the results. The straightness part can be

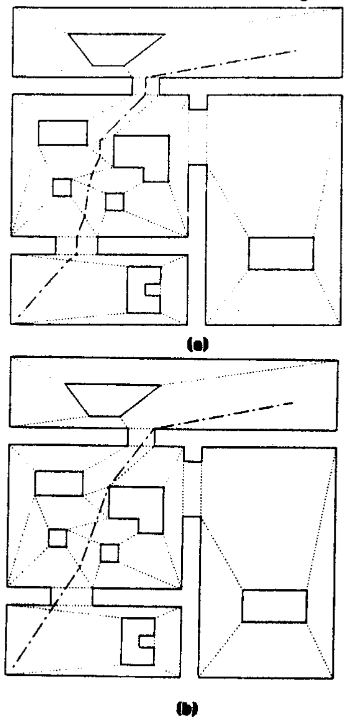
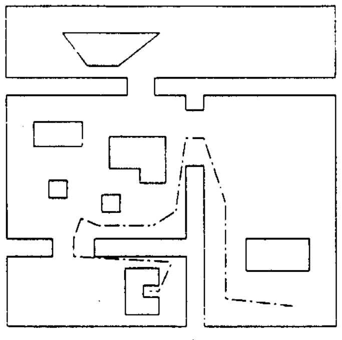


Figure 9. Single Terrain Path Planning Example (A*-1)

(a) The initial path produced by the A* search algorithm through the midpoints of the passable bordering meadows.

(b) The path after undergoing path improvement strategies


a)

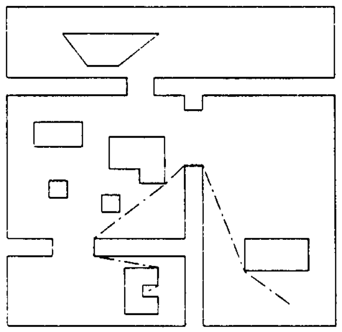

b)

Figure 10. Another Single Terrain Planning Example
(a) Initial coarse path
(b) Improved path (safety margin 1 foot)

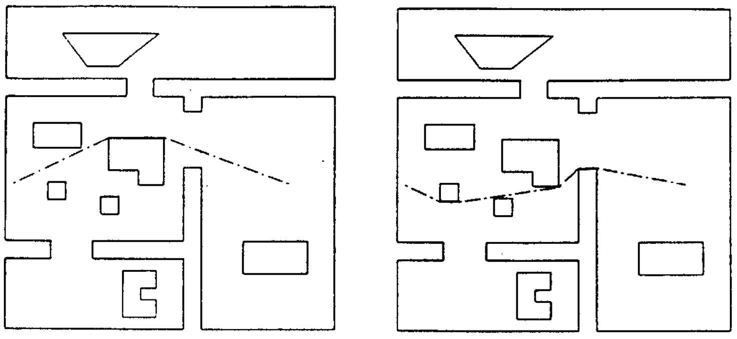

(a) (b)

Figure 11. A*-1 versus A*-3
In tight quarters, A*-3 can make a slight difference. The gain however is small, on the order of 2%.
(a) A*-1 final improved path.
(b) A*-3 final improved path.

observed to remove unnecessary detours around obstacles while the tightening component reduces the overall path cost.

For A*-1, (figures 9-10), the actual paths produced from the navigator are a function not only of the start-end points and improvement strategies used, but are also dependent on the modes used during the mapbuilding. Considerable experimentation was conducted trying to determine which if any of the nine modes available to the mapbuilder resulted in consistently better paths. No clear connection could be made between the cost of the path, the start and end points of the path, and the nature of the convex region decomposition. In some decompositions, for a given start and end point a better path (A*-1) could be obtained using one decomposition approach over another. For another set of start-goal points however the same approach that performed poorly in the first case did better than the one that previously performed well. For an arbitrary start and end point, no single map building strategy was clearly superior.

For A*-3 noticeable improvement occurred. Figure 11 shows the ability to produce a better path than was the case with the A*-1 method. The computational penalty however can be significant as the search space is considerably larger[28].

4. Multi-terrain extensions

One of the principal advances of this works lies in its extension to handle diverse terrain types. Previously the regular grid has been the predominant approach used to deal with diverse ground covers[6]. Certainly, for the planner to produce realistic paths in outdoor scenarios, a reflection of different terrain types must be taken into account by the navigator. Some terrain types will be more costly to traverse than others (e.g. grass as opposed to concrete). We do not want to exclude these different terrains as navigable areas, but yet we don't want to lump them into one uniform terrain type. The traction of the vehicle will depend on the specific surface encountered. More slippage is expected to occur on gravel than on pavement. The cost in terms of positional uncertainty can be high on loose ground. On the other hand, if a significant reduction in the total distance to be traversed from start to goal can be obtained (and associated reduction in time cost), the tradeoff of increased positional uncertainty for greater time savings may be warranted. In some cases the total amount of positional uncertainty gained by traveling over poor surfaces may be substantially less than that garnered by traveling over a superior cover due to the much shorter distance the robot may travel in taking a rougher terrain short-cut.

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

This section will first describe how the mapbuilder accommodates multiple terrain types through the construction of transition zones. The next section will describe how the navigator has been modified to accommodate path-planning through this extended representation.

4.1 Multi-terrain mapbuilder

The extended mapbuilder is built from the uni-terrain mapbuilder described in section 2.1. The algorithm appears in figure 12. The input structure of a terrain region is identical to that for the previous mapbuilding algorithm: a list of border and obstacle vertices. This region is decomposed in exactly the same manner as before. All borders of the terrain region and enclosed obstacles (which may later turn into other terrain regions) are initially labeled as impassable.

After the initial terrain area is decomposed, the mapbuilder algorithm keeps accepting new areas until none remain. After each terrain area is decomposed in isolation, a matching algorithm is run on each of the new terrain convex regions to see if it shares any common impassable edges with any of the previously decomposed regions. If a match is identified, evidenced by at least partial overlap of any impassable edges, a transition zone is built.

MULTI-TERRAIN MAPBUILDER ALGORITHM

```
DO WHILE no more terrain to add
    Run the uni-terrain map builder (fig. 5) on a terrain region
    Tag all resulting free space regions with a new terrain identifier
    Match borders of new free space regions against
        terrain free space regions already produced
    IF matches exist
        Build transition zones connecting terrain types
        Add these transition zones to free space regions
    ENDIF
ENDDO
```

Figure 12.

The transition zone is a special region connecting two differing terrain types. Basically, the two grown edges representing the two matched regions are used for two of the edges (fig. 13a). This gives a distance across the zone equal to the robot's diameter plus twice the safety margin that was used in the shrinking of the initial terrain regions (see fig. 13b). The initial zone consists of the four vertices of the two matched edges.

It is highly desirable to minimize the time it takes for the robot to cross a transition zone, implying a normal straightline path. Consequently the initial polygonal representation is converted into a rectangle (fig. 13c). The new edges produced are labeled as impassable, producing small forbidden zones which the planner construes as unnavigable. Any path that is produced by the path planner is guaranteed to be normal to the original matched edges, thus ensuring the smoothest and fastest transition possible. Finally, appropriate passable links are made to connect the new transition zone and the two bordering free space regions of the different terrain types.

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

Summarizing: wherever two different terrain types are found to touch, rectangular transition zones are built allowing a limited type of traversability between them. As a side effect, forbidden zones, (corners of intersecting bounding regions), are marked as off-limits for later path planning purposes. This restriction ensures that any path taken across a transition zone will result in a minimal distance path across said zone.

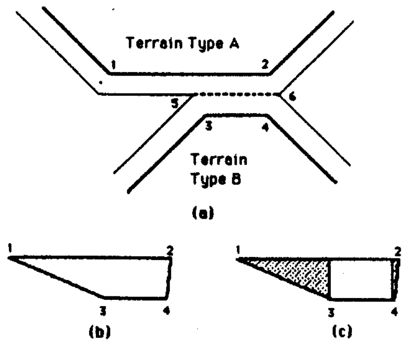


Figure 13. Transition Zone Construction
(a) Initial bordering terrain types. Terrains A and B share a common edge from vertex 5 to 6.
(b) The initial transition zone is built by connecting the four vertices of the bordering C-space lines.
(c) The initial region is converted into a rectangle yielding the final transition zone. The resulting forbidden zones are shown as shaded areas.

4.2 Multi-terrain navigator

The navigator must be modified to ensure that the path produced is reasonable in the multi-terrain case. The only components of the navigator that must be changed are the path improvement strategies. This includes both the straightening and tautness components. No modifications whatsoever are necessary for the search component. As the transition zone is rectangular, any path produced by the A*-1 method through the midpoints of passable regions is guaranteed to result in a straightline across the transition zone. The A*-3 case occasionally requires slight path preprocessing to ensure a perpendicular crossing of the transition zones prior to improvement. Unfortunately, modifying the path improvement strategies (for both A*-1 and A*-3) was non-trivial and required the implementation of a relaxation algorithm, limited to relaxing the terrain crossings only.

The algorithm for multi-terrain path improvement is shown in figure 14. Actually, the complexity is somewhat greater due to special case treatment. Reviewing the algorithm: the previous path improvement strategy is first run within the framework of each terrain type in isolation. This is the identical algorithm as described in section 3.2 but restricted to individual terrain types. To reduce the cost of the relaxation later, the transition zone crossings are then slid in the same manner as was done for the individual meadow border passages, with one exception. Both crossing points on the transition zones are slid in tandem, insuring a perpendicular passage across the transition region. Any previously unmoved vertices within the regions themselves are then retested to see if sliding will lower the overall cost. If necessary, additional path straightening is then performed.

Although avoiding a relaxation method for path improvement was an initial design goal due to perceived high computational costs, it eventually became necessary to resort to one. The cost associated with this relaxation is not particularly high however, due to the preprocessing on the path and, more importantly, only the transition zone crossings are relaxed. The algorithm used is fairly standard: displace the transition zones an increment in both directions and measure the lowest cost. Use the new lower cost point as the starting point for the next displacement. Keep repeating until any displacement results in a higher or equal cost path. Convergence is guaranteed using this standard hill-climbing methodology. The time for convergence is determined to a large extent by the displacement size and on the number of terrain crossings. The results for the worst test cases in the lab have yielded times for the relaxation component that are not disproportionate with the other components of the planning algorithm[28]. Finally the path is attempted to be restraightened within the context of each terrain type before final release to the pilot.

4.3 Results

A schematic model of the environment outside the Graduate Research Center was used for the outdoor terrain examples. Five different terrain regions are present: concrete, two disjoint grassy regions, a gravel path and a parking lot. For the purposes of path planning: the traversability factor of the concrete and the parking lot was set to 1.0, grass 1.5, and gravel a factor of 1.2 (these are relative values). The terrain types and their associated transition zones can be seen in figure 4.

In figures 15 and 16, the results of the path planning algorithm are illustrated. The A*-1 search method was used for all these cases. Sub-figures (a) show the initial path through the search space. Note in this and all other cases the perpendicu-

MULTI-TERRAIN PATH IMPROVEMENT ALGORITHM

```
Accept a coarse path from search component of navigator
Run tautness and straightness component of uni-terrain path planner
    on each part of path within a given terrain type (fig. 8)
Slide only the transition zones as in previous tautness algorithm
Run tautness and straightness component again on each part of path
    within a given terrain type (only on previously unmoved vertices)
Relax path by settling transition zone crossings into
    a minimal cost point
Restraighten if necessary
```

Figure 14.

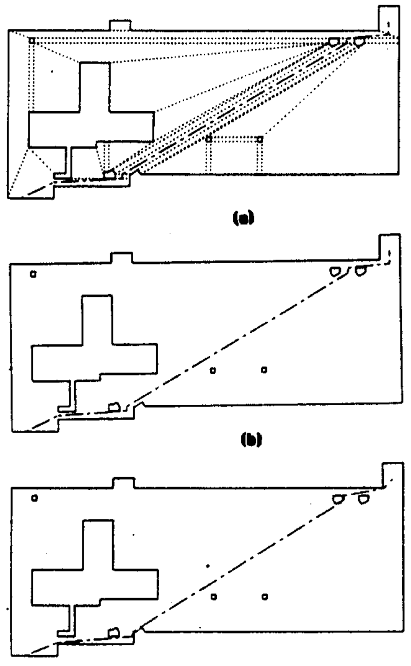


Figure 15. Multi-terrain Path Planning Example
A start point in the lower left corner on concrete with goal in upper right corner on concrete. The path planner decides it is more efficient to take the gravel path to acheive its goal, requiring the traversal of two transition zones.
(a) Initial raw path from A* search through midpoints of passable regions. The cost function included a traversability factor dependent on terrain.
(b) The same path without the passable borders. Note the forbidden zones present at the edges of transition zones.
(c) The final improved path. Note how the total length over concrete was lengthened while the distance over gravel shortened. (safety margin 1 foot).

lar passage through the transition zone is evident. Sub-figures (b) show the improved path before transition zone relaxation. Sub-figures (c) display the final path after relaxation and post-relaxation straightening.

5. UMASS AuRA architecture overview

A block diagram of the proposed system is presented in figure 17. Central to the system is the hierarchical planner consisting of a pilot, navigator and mission planner and charged with the responsibility for decision making (sec. 3). The motor schema manager, the execution arm of the pilot, implements pilot-specified behaviors at the lowest level of planning (sec. 3). A cartographer, whose task is to maintain and provide information on demand to requesting planning and sensory modules, is adjacent. The mapbuilder (sec. 2.1) resides under the cartographer's jurisdiction. The cartographer also includes modules to maintain the spatial error map. This map provides an explicit representation of the robot's probable location and orientation and is used to restrict perceptual processing. Appropriate transforms for each "turn-and-run" move of the robot increase the area of probable positional limit while landmark identification and other localization techniques are used for error area reduction. Statistical translational and rotational error data gathered experimentally from diverse terrain types are used in the error transforms.

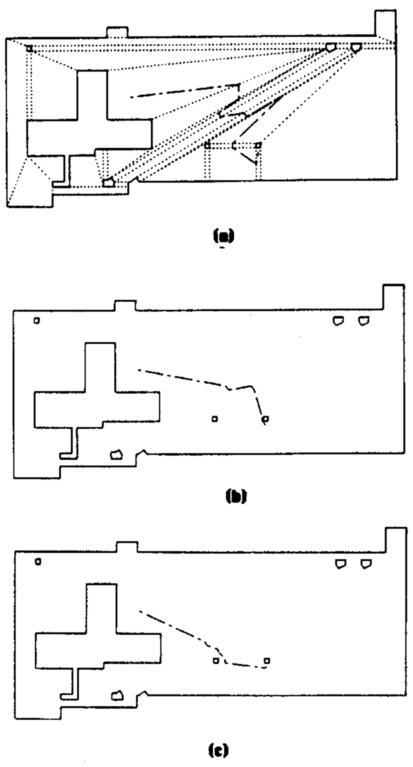


Figure 16. Yet Another Multi-terrain Path Planning Example
Start in lower right- goal in upper left.
(a) Initial raw path.
(b) Improved but unrelaxed path.
(c) Final relaxed path.

A perception subsystem is delegated the task of fielding all sensory information from the environment, performing some preliminary filtering on that data to remove noise and then structuring it in a form that is acceptable to the cartographer and VISIONS. It is also the location where expectations are maintained to assist sensory processing.

The motor subsystem is the means by which the vehicle interacts with its environment in response to stimuli. Motors and motor controllers serve to effect the necessary changes. A vehicle interface is provided that guides the motors to carry out the requested response from the higher level processing.

One of the distinguishing characteristics of this architecture is the utilization of cybernetic theory as the basis for much of its design. Concern with the action-perception cycle, the implementation of reflex arcs through panic shunts from the sensor interface to the motor subsystem bypassing higher level planning ,n times of eminent danger, consideration of the global data structures in the context of long-term and short-term memory, and the use of perceptual and motor schemas, are important contributions to the overall design of a mobile robotic system. A

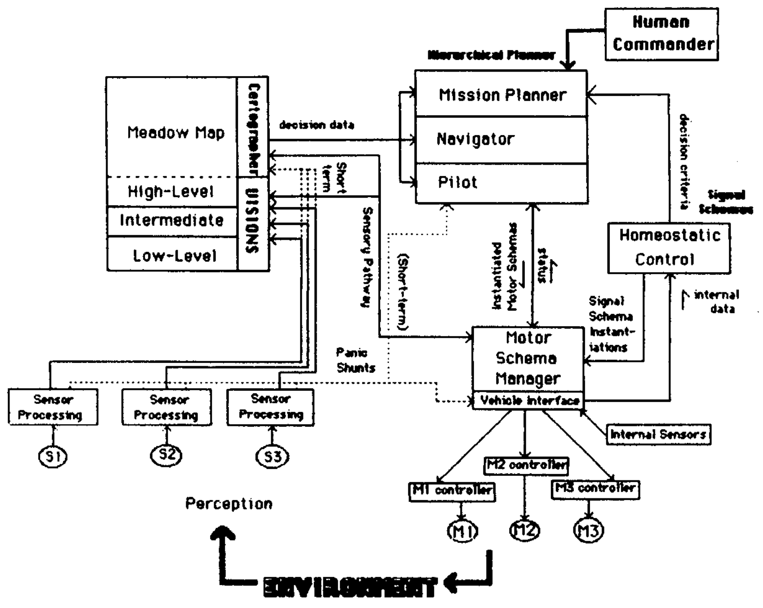


Figure 17. AuRA

homeostatic control subsystem is present which is concerned with maintaining a safe internal environment for the robot. Utilizing schema theory[29], dynamic behavioral modifications are made in response to the changing internal conditions of the robot (energy level, temperature, etc.) For further information regarding the system architecture see [30].

6. Summary and conclusions

In order to produce a reasonable path through a partially modeled environment, a hybrid vertex-graph free space representation was chosen for a long term memory model of the world. This meadow map decomposes free space into a group of connected convex regions. Data for landmark recognition, localization, error modeling, and the like can be associated with these regions or their obstacles through the use of a feature editor.

The output of the mapbuilder is consumed in part by the navigator component of the planner process whose duty it is to build a collision-free path through the partially modeled world represented in LTM. An A* search is conducted through the midpoints (A*-1) or triads (A*-3) of the bordering passable convex regions to arrive at a coarse path. This is then subjected to path improvement strategies which tighten and straighten the path subject to parameters specified by the mission planner. This allows for the production of short, safe, or other types of paths more freely than other representations might allow.

Multiple terrain situations are accommodated by extending the basic algorithms to include the construction of transition zones. These zones assure minimum distance traversal when the robot changes terrain types, minimizing the increase in positional uncertainty inherent in this maneuver.

The navigation and long term memory component of the overall UMASS architecture described within this paper is to be the basis for path planning for vision-based mobile robot experiments. The flexibility to accommodate diverse new representations, without any changes to the underlying path planning representation, is a major advantage of this approach.

Acknowledgments

The author would like to thank Prof. E. Riseman, Prof. A. Hanson, Prof. M. Arbib and the members of the LPR and VISIONS groups for their assistance. This research is being conducted as part of the requirements for completion of the author's Ph.D. degree at the University of Massachusetts, Amherst.

References

1. Amarel, S., "On Representations of Problems of Reasoning about Actions", Machine Intelligence 3, 1968, reprinted in READINGS IN ARTIFICIAL INTELLIGENCE, ed. Webber and Nilsson, pp. 2-22, Tioga, 1981.
2. Iyengar, S., Jorgensen, C., Rao, S., and Weisbin, C., "Learned Navigation Paths for a Robot in Unexplored Terrain", IEEE Second Conf. On Artif. Intel. App., pp. 148-155, Dec. 1985.
3. Kuan, D.T., Zamiska, J., and Brooks, R., "Natural Decomposition of Free Space for Path Planning", Proc. IEEE Int. Conf. Robotics and Automation, St. Louis, Mo., pp. 168-173, 1985.
4. Miller, D., "A Spatial Representation System for Mobile Robots", CH2152-7/85, IEEE, pp. 122-127, 1985.
5. Krogh, B., "A Generalized Potential Field Approach to Obstacle Avoidance Control", RI Technical Paper, MS84-484, Soc. of Mech. Engr., 1984.
6. Mitchell, J. and Keirsey, D., "Planning Strategic paths through Variable terrain data", SPIE Vol. 485, Appli. of Artif. Intell., pp. 172-179, 1984.
7. Mobile Robot Laboratory Staff, "Towards Autonomous Vehicles", Annual Research Review, The Robotics Institute, Carnegie-Mellon University, 1984.
8. 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.
9. Thorpe, C., "Path Relaxation: Path Planning for a Mobile Robot", CMU Robotics Inst. Tech. Rep. CMU-RI-TR-84-5, 1984.

10. Andresen, F.P., Davis, L.S., Eastman, R., and Kambhampati, S., "Visual Algorithms for Autonomous Navigation", Proc. IEEE Int. Conf. Rob. and Auto., St. Louis, Mo., pp. 856-861, 1985.
11. Tachi, S. and Komoriya, K., "Guide Dog Robot", ROBOTICS RESEARCH, Second Int. Symposium, ed. Hanafusa and Inoue, MIT Press, pp. 333-340, 1985.
12. 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.
13. Monaghan, G., "World Modeling and Path Planning for Autonomous Mobile Robots", manuscript 579-51.
14. Thorpe, C., Kanade, T., and Shafer, S., "ALV System Integration Plan", The Robotics Inst., CMU, May 1985.
15. Arkin, R.C., "Path Planning and Execution for a Mobile Robot: A Review of Representation and Control Strategies", COINS Tech. Rept., Comp. and Info. Sci. Dept., Univ. of Massachusetts, 1986.
16. Keirsey, D., Mitchell, J., Payton, D., and Preyss, E., "Multilevel Path Planning for Autonomous Vehicles", SPIE Vol. 485, Applications of Artificial Intelligence, pp. 133-137, 1984.
17. Stentz, A. and Shafer, S., "Module Programmer's Guide to Local Map Builder for ALVan", CMU Comp. Sci. Dept., July 1985.
18. 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.
19. 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.
20. Weymouth, T.E., "Using Object Descriptions in a Schema Network for Machine Vision", PhD dissertation, Comp. and Info. Sci. Dept., Univ. of Massachusetts, May 1986.
21. Koch, E., Yeh, C., Hillel, G., Meystel, A., and Isik, C., "Simulation of Path Planning for a System with Vision and Map Updating", Proc. IEEE Int. Conf. Rob. and Auto., St. Louis, Mo., pp.146-160, 1985.
22. Nitao, J. and Parodi, A., "An Intelligent Pilot for an Autonomous Vehicle System", IEEE Second Conf. On Artif. Intel. App., pp. 176-183, Dec. 1985.
23. Weems, Charles, "Image Processing on a Content Addressable Array Parallel Processor", COINS Technical Report 84-10, Comp. and Info. Sci. Dept., Univ. of Massachusetts, 1984.
24. 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.
25. Moravec, H. and Elfes, A., "High Resolution Maps from Wide Angle Sonar", CH2152-7/85 1985 IEEE, pp. 116-121, 1985.
26. Crowley, J., "Navigation for an Intelligent Mobile Robot", CMU Robotics Institute Tech. Rep., CMU-RI-TR-84-18, 1984.
27. Lozano-Perez, T., "Automatic Planning of Manipulator Transfer Movements", in ROBOT MOTION: Planning and Control, ed. Brady et al, pp. 499-535, 1982.
28. Arkin, R.C., "Path Planning for a Vision-based Autonomous Robot", forthcoming COINS Technical Report, Comp. and Info. Sci. Dept., Univ. of Massachusetts, 1986.
29. Arkin, R.C., "Internal Control of a Robot: An Endocrine Analogy", unpublished paper, Dec. 1984.
30. Arkin, R.C., Ph.D. Proposal, Computer and Information Science Department, University of Massachusetts, Spring 1986.