NAVIGATION FOR AN INTELLIGENT MOBILE ROBOT

JAMES L. CROWLEY

Reprinted from IEEE Journal of Robotics and Automation, Vol. RA-1, No. 1, March 1985

Navigation for an Intelligent Mobile Robot

JAMES L. CROWLEY, MEMBER, IEEE

Abstract—**A navigation system is described for a mobile robot equipped with a rotating ultrasonic range sensor. This navigation system is based on a dynamically maintained model of the local environment, called the *composite local model*. The composite local model integrates information from the rotating range sensor, the robot's touch sensor, and a pre-learned *global model* as the robot moves through its environment. Techniques are described for constructing a line segment description of the most recent sensor scan (the *sensor model*), and for integrating such descriptions to build up a model of the immediate environment (the composite local model). The estimated position of the robot is corrected by the difference in position between observed sensor signals and the corresponding symbols in the composite local model. A learning technique is described in which the robot develops a global model and a *network of places*. The network of places is used in global path planning, while the *segments* are recalled from the global model to assist in local path execution. This system is useful for navigation in a finite, pre-learned domain such as a house, office, or factory.**

I. INTRODUCTION

THIS work describes a system for autonomous navigation by an intelligent mobile robot in a known domain. This system is based on maintaining a description of the external environment of the robot using a focused rotating ultrasonic ranging device. The system is designed to provide autonomous navigation by an intelligent mobile robot in a previously learned floorplan.

The techniques described are part of an effort to develop a low-cost *intelligent mobile platform* (IMP). By the term "intelligent" we mean that the system is designed to plan and execute tasks based on a model of the current state of the external world. The IMP is designed to respond to commands of the form "go to ⟨place⟩" where ⟨place⟩ is a pre-learned location in a network of "learned places." The IMP is able to use its network of places to plan a path to ⟨place⟩. It is then able to use its sensing, modeling, and navigation abilities to execute this plan and to modify the plan dynamically in reaction to unexpected events. The IMP is to serve as a foundation for mobile household, business, and factory robots which require intelligent navigation.

This first section introduces the problems of world modeling, position estimation, and navigation and summarizes solutions for each of these problems. Techniques for dynamic world modeling, path planning, learning, position estimation, and navigation are then described.

Manuscript received September 4, 1984; revised February 1985. This work was supported by Commodore Business Machines, Inc., Denning Mobile Robotics, Inc., and The State of Pennsylvania.

The author is with The Robotics Institute, Carnegie-Mellon University, Pittsburgh, PA. He is currently on leave with IMAG, Laboratoire LIFIA, BP 68, 38402 St. Martin d'Heres Cedex, France.

A. Navigation and Dynamic World Modeling: The Problem

The task of a navigation system is to plan a path to a specified goal and to execute this plan, modifying it as necessary to avoid unexpected obstacles. The path planning problem can be divided into global path planning and local obstacle avoidance. Global path planning requires a pre-learned model of the domain which may be a somewhat simplified description of the real world and might not reflect recent changes in the environment. This global model must provide the planning algorithm with a network of landmark points which are connected by simple local movements.

A local navigation system carries out the steps in the global plan, maintaining an estimate of the robot's position with respect to the global model and planning local paths as needed to avoid unexpected obstacles. Whereas global navigation may operate on a pre-stored model, local navigation requires a model that reflects the state of the evironment, including changes, as the plan is being executed. This is provided by the *composite local model*.

The composite local model is built up by integrating recent information from different sensors, taken from different positions. When available, information from a pre-learned *global model* is also integrated into the composite local model. The construction and maintenance of a composite local model involves:

1) building an abstract description of the most recent sensor data (a *sensor model*);
2) matching to determine the correspondence between the most recent sensor models and the current contents of the composite local model;
3) modifying the components of the composite local model and reinforcing and decaying the confidences to reflect the results of matching.

Having the correspondence between the sensor model and the composite local model also makes it possible to measure and correct for errors in the estimated position and orientation of the robot due to wheel slippage.

B. Summary of Solution

In the system to be described, global path planning is based on a pre-learned *network of places*. The network of places is learned in a special "active learning mode" in which the robot explores its environment. Automatic learning greatly simplifies the practical problem of giving the system an accurate model of the world. Each place in the network is connected to a set of adjacent places by "legal highways." Global navigation is a process of choosing a set of legal highways that

0882-4967/85/0300-0031$01.00

will carry the robot from its current location to the specified goal. Traversing each legal highway and planning paths to avoid unexpected obstacles is the job of the local navigator. The network of places and its use in path planning is described in Section VII.

A legal highway consists of a straight line path which connects two landmark points. Local navigation is accomplished by a finite state process which turns the IMP toward the next landmark point and keeps the IMP on its path as it moves. Each path is tested for blocking obstacles using both the raw sensor data and the composite local model. If an obstacle is detected, a recursive obstacle avoidance procedure plans a new sequence of straight line paths to the next local goal. This recursive obstacle avoidance procedure is based on the current contents of the composite local model. The local navigation process is described in Section VIII.

The composite local model, the sensor model, and the global model are represented in terms of line segments in a two-dimensional (2-D) "floor-plan" world. All three models are expressed in a world-centered coordinate system so that they can be matched invariant to the robot's position. The line segments that compose the sensor model are constructed using a variation of the recursive line splitting algorithm which is often used to find edges in images [5]. This process is described in Section IV.

The confidence of line segments in the composite local model is represented by a finite set of states. A relatively simple state transition mechanism is used to reinforce and decay the confidence in line segments. Segments in the composite local model are "grown" by an update process that extends the segments whenever there is a partial overlap with sensor model segments. The process of incrementally matching and updating a composite local model is described in Section V.

As the IMP travels it uses the mismatch in position between the *sonar* model and the composite local model to detect and correct errors in its estimated position. These techniques are described in Section VI.

When the IMP navigates autonomously, it recalls expected segments from a prelearned global model into the composite local model. The global model, which provides the basis for the network of places, is learned during a special *Learn Mode*. In learn mode the IMP systematically learns the geometry of a finite domain by a wall-following technique. Learn mode is discussed in Section IX.

C. Problem Context

A photograph of the IMP is shown in Fig. 1. At the top of the IMP is a rotating depth sensor which senses the distance to external surfaces with a beam with a starting diameter of approximately 3 in and a beam spread of approximately 5°. The sensor is mounted at a height of 30 in, which is about the level of most tables. The sensor is turned by a stepper motor in steps of 3°. Approximately 10 s are required to obtain the 120 depth readings given by a complete revolution. With each reading, the sensor returns the distance to the nearest surface within 25.6 ft to a resolution of 0.10 ft. As the IMP travels, rotary position encoders mounted on its power wheels are used to maintain an instantaneous estimate of the IMP's position in a Cartesian coordinate system.

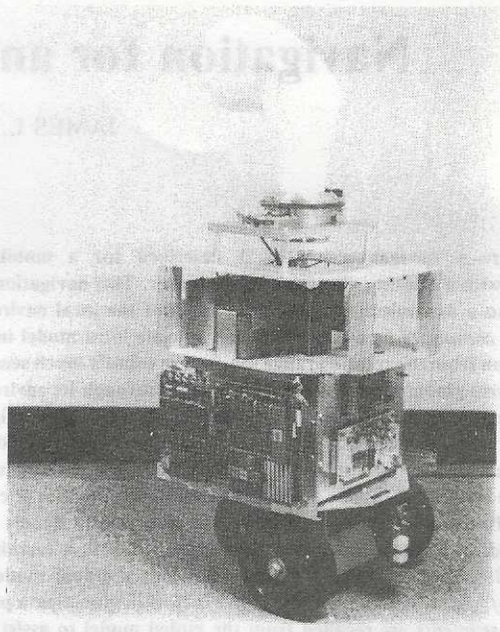

Fig. 1. Experimental prototype intelligent mobile platform.

The world modeling and navigation procedures for the IMP were originally implemented and refined using an interactive mobile robot simulation program. These techniques have been reimplemented on the IMP using the two on-board 16-bit microprocessors. Similar techniques have recently been implemented for a mobile security robot which uses 24 ultrasonic ranging devices, arrayed in a ring, in place of the rotating focussing horn.

II. REVIEW OF PRIOR TECHNIQUES

A number of interesting research results have been obtained on problems that are relevant to mobile robot navigation. A quick review of the salient systems provides a picture of the current state of the scientific art.

A. Find-Path

Planning a path based on a model is a problem that is fundamental to intelligent control of robot arms as well as mobile robots. Lozano-Pérez has developed a formal version of the general path planning problem. This formalization is referred to as the "find-path" problem [7]. In its most general form, the goal of find-path is to determine a continuous path for an object from an initial location to a goal location without colliding with an obstacle.

Lozano-Pérez provided a mathematical treatment of the find-path problem using the "configuration space" approach. The idea is to find those parts of free space which the object at particular orientations may occupy without colliding with an obstacle. Obstacles are "expanded" by the shape of an object at a set of orientations, while the object to be moved is shrunk to a point. The shortest path for the object, including rotations, is computed as the shortest connected path through the expanded obstacles.

The shortest path through obstacles generally leads through

a sequence of points that are adjacent to the expanded obstacles. If there is position error in the control of the path execution, such points can possibly result in a collision. Brooks has recently proposed a new approach to the find-path problem based on modeling free space [2]. Brooks' solution, developed in a two-dimensional plane, involved fitting two-dimensional "generalized cylinders" to the space between obstacles to obtain pathways in which the object may freely travel on a plane. The technique was extended to the third dimension by stacking planes.

B. The Stanford Cart and the C-MU Rover

Moravec [8] developed a navigation system based on sensory signals using the Stanford cart. This cart sensed its environment using a set of nine stereo images obtained from a sliding camera. A set of candidate points were obtained in each image with an "interest" operator. Small local correlations were then made at multiple resolutions to arrive at a depth estimate for the points. The matched points were plotted on a two-dimensional grid and then expanded to a circle. A best path from the current location to a goal was then chosen as the shortest sequence of line segments which were tangent to the circles. The cart would advance by 3 ft and then repeat the sensing and planning process. Stereo matching was also performed between the images taken at different steps to obtain confirming and additional depth information. A new vehicle, called the C-MU Rover [9], has recently been constructed by Moravec to support these techniques.

C. Hilare

A team under the direction of George Giralt at the LAAS laboratory in Toulouse has been investigating the design and control of mobile robots since 1977. They have developed a mobile robot named Hilare. Chatila developed a navigation system for Hilare that is based on dividing a pre-learned floor plan into convex regions [3]. Convex regions were formed by connecting nearest vertices to form areas called *C*-Cells. Laumond, at the LAAS in Toulouse, extended this idea by developing hierarchies of *C*-Cells to represent rooms and parts of a known domain [6].

D. Comment

A few other efforts towards developing autonomous mobile robots have also been reported. In many cases the efforts focus on engineering problems and pay little attention to the issues of world modeling or path planning [10]. Other groups have become bogged down on the vision problem, often spending their efforts on general solutions to the problems of low level vision. We believe that the most important problems to be addressed now are sensor interpretation, navigation, and system organization. Toward this end, we have developed a computational paradigm for intelligent robotic systems. This computational paradigm provides a framework for the processes involved in sensor interpretation, path planning, and path execution.

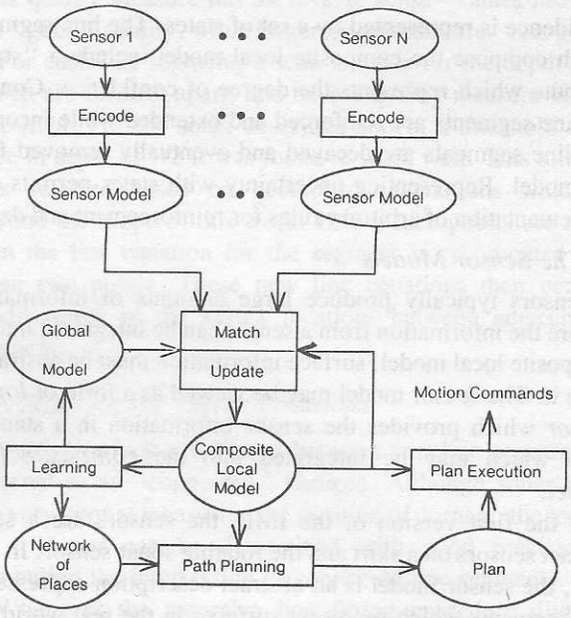

Fig. 2. Framework for intelligent navigation system.

III. THE COMPUTATIONAL FRAMEWORK

A. The Composite Local Model

The navigation system of the IMP is based on the computational framework shown in Fig. 2. At the core of this framework is a dynamic model of the surface and obstacles in the immediate environment of the IMP called the composite local model. "Local" refers to the fact that only information in the local environment of the IMP is represented. "Composite" refers to the fact that this model is composed of information obtained over time from multiple sensors and from many views.

The composite local model plays two fundamental roles in this computational framework.

1) It is the structure in which potentially conflicting information from diverse sensors is integrated with recently observed information and information recalled from long term storage (the Global Model).
2) It is the structure on which processes for local path planning, path execution, learning, object tracking, object recognition, and other "higher level" processes are based.

Because of the nature of the navigation task and the sensors that are employed, the composite local model in the IMP is implemented with a relatively simple 2-D representation. The IMP models the world and plans paths in a 2-D "flat-land" universe. Because the rotating range sensor is mounted at a height of 30 in, the robot is able to detect and represent most of the furniture that it encounters. Surfaces and obstacles are represented as connected sequences of line segments. Thus a table and a wall have the same structure; both appear as a barrier with an infinite (or unknown) extent in vertical dimension.

The composite local model must include the ability to represent the uncertainty of information. In this system,

confidence is represented by a set of states. The line segments which compose the composite local model include a "state" attribute which represents the degree of confidence. Consistent line segments are reinforced and extended while inconsistent line segments are decayed and eventually removed from the model. Representing uncertainty with states permits easy implementation of arbitrary rules for reinforcement and decay.

B. The Sensor Models

Sensors typically produce large amounts of information. Before the information from a sensor can be integrated into the composite local model, surface information must be abstracted from it. The sensor model may be viewed as a form of *logical sensor* which provides the sensor information in a standard form which may be integrated into the *composite local model*.

In the first version of the IMP, the sensors are a set of contact sensors on a skirt and the rotating sonar sensor. In each case, the sensor model is an abstract description expressed as line segments which represent surfaces in the real world.

C. Match and Update

The module labeled "match" determines the correspondence between the line segments which compose the sensor model and the line segments which compose the composite local model. The correspondence is then used to determine errors in the estimated position and to update the position, length, and confidence of the segments in the composite local model. Special procedures also exist for detecting and tracking moving objects.

The module labeled "update" integrates the information from the sensor models with the current composite local model. This module adjusts the position, size, connectivity, and confidence of the segments in the composite local model to reflect the results of correspondence matching. This update process also removes segments for which the confidence is low or for which the distance is too far. The process does not remove nearby surfaces which are not currently visible.

IV. CONSTRUCTING AN ABSTRACT DESCRIPTION OF RANGE DATA

Depth readings from the rotating sonar are converted into line segments by a sequence of four steps.

1) Project the reading to a Cartesian world coordinate system.
2) Segment measured points into line segments by detecting "discontinuities" and then applying a recursive line fitting process.
3) Compute the line equations of the points from the most reliable interior points.
4) Recompute the segment endpoints as the intersection points with neighboring line segments.

These processes are described below.

A. Projection to Cartesian World Coordinates

Depth readings are obtained from the rotating sonar in cylindrical coordinates, i.e., as depth at a particular angle. As each depth reading is made, the current estimated position and orientation of the IMP is affixed to it. This permits the system to project the reading into a world coordinate system at a later time, even if the data were taken while the IMP was moving.

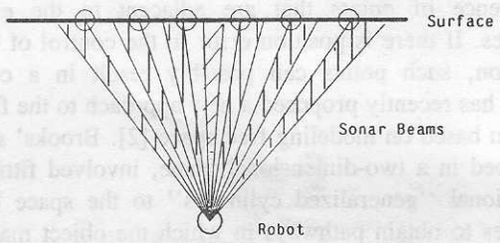

Fig. 3. Edges of sonar beam are projected to world coordinates.

As a consequence of the detection mechanism in the sonar, the depth reading refers to the depth to the nearest reflecting surface anywhere in the sonar beam's circular footprint. When the beam reflects from a flat surface at a non-perpendicular angle, the sonar returns the distance along the short edge of the beam. Knowledge of this physical process is used in interpreting the sonar depth readings.

As each sonar reading is obtained, it is converted from robot-centered polar coordinates to a world-centered Cartesian coordinate system. This is done by projecting a line by the specified depth at the specified angle, as illustrated in Fig. 3. If the readings are decreasing as the sonar rotates in a counterclockwise turn, the angle is adjusted to be the left edge of the beam by adding the estimated half angle of the sonar beam. If the depth is increasing, the angle is adjusted to the right by subtracting the estimated half angle. The difference in depth between the adjacent readings to the right and to the left is computed and affixed to the projected beam as a quality measure.

B. Segmenting the Points Into Line Segments

The points are first grouped into a sequence of roughly colinear readings such that the distance between each adjacent pair of points is less than a tolerance. This tolerance is selected as a compromise between the maximum distance at which the depth readings can be taken and the smallest gap between objects that the system can detect. For a difference in orientation of α degrees per reading, the minimum distance gap size $G_{\min}$ is determined by the desired maximum range R by considering the difference of beam edges for a perpendicular surface. Such a geometry gives the relationship

$$G_{\min} > R\,Tan(\alpha)$$

In our system, $\alpha = 3°$ and $R = 25.6$ ft, giving $G_{\min}$ of 1.34 ft. We have found a value of $G_{\min}$ of 1.5 ft to be satisfactory. Thus, the points are scanned to detect any points where the distance between adjacent readings is greater than $G_{\min}$. Such points are called break points or discontinuity points. Break points mark the boundaries of collections of points that are passed to a recursive line fitting procedure.

Recursive line fitting has been used for years to fit lines to edge points in images [5]. The algorithm is illustrated in Fig. 4. A line equation of the form

$$Ax + By + C = 0$$

is computed between the two endpoints in the collection of

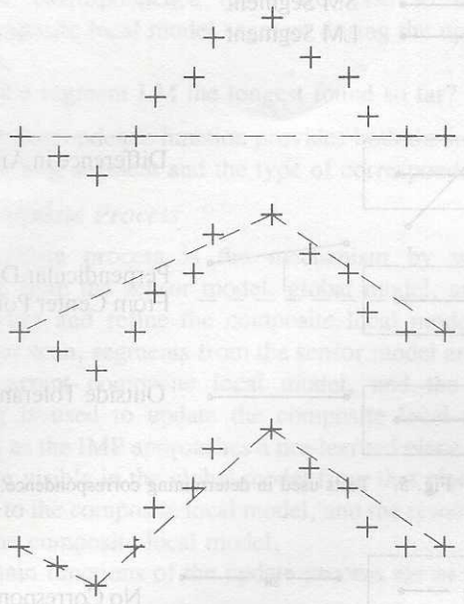

Fig. 4. Recursive line fitting.

points. If the coefficients A and B are normalized so that the sum of their squares is one, then evaluating the line equation at the location of a point (x, y) gives the perpendicular distance from the computed line equation. The points in the group between the endpoints are tested to detemine the point where the perpendicular distance is largest. If this largest perpendicular distance is below a tolerance, then the line is accepted as representing the points. Otherwise, the collection of points is divided into two groups at the point where the perpendicular distance was largest. The line fitting procedure is then evaluated recursively for each of these two groups. The result is a collection of line segments which represents the collection of points.

C. Refining the Line Equation

When a sonar beam measures depth near a corner, the depth measurement is often corrupted by reflections. Yet these points give the breakpoints which are used for recursive line fitting. Thus it is desirable to recalculate the equation of each line segment from interior points.

In our early experiments, we observed that the most reliable sonar points are those to either side of the measurement which is perpendicular to the surface. The perpendicular measurement may be detected as a smooth minimum in the depth readings. However, if a line equation is fit to points that are too close to each other, the equation is very sensitive to small errors in position. To compromise between these conflicting constraints, the difference in depth between sonar readings is used as a quality measure. For each point, a first difference operator (a discrete derivative) is computed from the difference in depth of the point to the left and to the right. That is, if the depth readings are denoted by a sequence, $D(k)$, then the quality measure, $Q(k)$ is given by

$$Q(k) = |D(k-1) - D(k+1)|.$$

This quality measure has an inverse sense—values near zero are "good" quality, while larger magnitudes are "less good."

For each line segment, a scan is made for the depth points which are furthest apart, and have a quality measure below a threshold. For the sonar described above, a threshold difference in depth of 1/2 ft was found to work well. This tolerance was selected based on geometric calculations which are beyond the scope of this work. If two such points are found, then the line equation for the segment is recomputed using these two points. These new line equations then permit a readjustment of the vertex locations between adjacent line segments.

D. Adjusting the Vertex Locations

Vertices which are shared between two line segments are referred to as "connected" vertices. Although sonar beams are very poor at measuring the position of corners, the location of a corner can be determined with good precision by computing the intersection of connected line segments.

Whenever the recursive line fitting procedure divides a group of points, the resulting pair of line segments will share a common endpoint. In such a case, the position of the shared vertex is computed from the intersection of the line equations which express the two lines. This has been found experimentally to yield corner locations whose position accuracy is close to the depth resolution of the sonar ranging device. It is important to accurately detect the locations of corners because these points are used to correct for errors in the estimated position of the IMP that arise due to wheel slippage.

The result of this sequence of operations is a list of line segments. These line segments comprise the sensor model used to verify that the IMP is not about to collide with an obstacle, to correct errors in the IMP's estimated position, and to update the composite local model.

V. THE COMPOSITE LOCAL MODEL

As previously noted, the composite local model is at the core of the world modeling and navigation system. Three functions based on the composite local model have been found to have very wide utility throughout the navigation system. These functions—VISIBLE, FREEPATH, and CORRESPOND—are described as follows.

VISIBLE: Is the point P visible from the location L? If not, what is the index of the nearest composite local model line segment which blocks it, and what is the location of the intersection point between this line segment point and the line segment from L to P?

FREEPATH: Is it possible for the IMP to pass from location L to location P? If not, what is the index of the line segment that gives the nearest collision? FREEPATH is implemented as a sequence of calls to VISIBLE along parallel lines.

CORRESPOND: What is the index of the line segment in the composite local model which corresponds to a given line segment?

A. Representing the Composite Local Model

The composite local model is represented as a list of directed line segments. Each line segment contains two vertices which are ordered in a counterclockwise direction. Each vertex is labeled as concave, convex, or disconnected. If the vertex is shared with another line segment, a pointer is given to that line segment. To save time in calculations, the line equation and the angle of the vertex are also stored in the structure.

In addition to the line segment information, each segment also has a *state* and a *type*. The state represents the confidence that the system has in the existence of that segment. At the current time, the state is represented by integers ranging from 1 (transient) to 5 (stable and connected). The type number represents the source of the segment. There is a precedence between sources of segments to resolve the type when a segment is given by more than one source.

B. Matching the Sensor Model to the Composite Local Model

The correspondence between line segments in the sensor model and line segments in the composite local model is needed to correct errors in the estimated position and to update the composite local model. Correspondence matching is also used in introducing line segments into the composite local model from the global model and from the contact sensor, for keeping track of the "current" line segment during active learning, and for a variety of other spatial reasoning functions.

The sensor model is matched to the composite local model in two stages. In the first stage, the best correspondence is found for each line segment in the sensor model by making a call to the function CORRESPOND. This list is then scanned to determine the sensor model line that has the best correspondence to each segment in the composite local model. This second correspondence list, from the composite local model to the sensor model, is then used for updating the composite local model.

C. The CORRESPOND Function

The function CORRESPOND is a general purpose function for determining which line segment in the composite local model has the best correspondence with a given line segment. A call to CORRESPOND is made for each segment in the sensor model. The CORRESPOND function is organized as a sequence of tests of increasing cost based on the attributes of orientation, position, and length. The correspondence problem is made very simple by assuming that the position and orientation of a segment are known within some tolerance. In the case of the sensor model, this assumption is justified because the IMP has kept track of its estimated position using wheel encoders as it moves. The required error tolerance in the estimated position can be reliably estimated.

The sequence of tests used by the correspondence function are illustrated in Fig. 5. In this figure, LM denotes composite local model, while SM denotes the line segment for which correspondence is sought. For a given segment SM the following tests are computed for each segment LM in the composite local model. If a segment LM fails any test, then the process advances to the next segment in the composite local model.

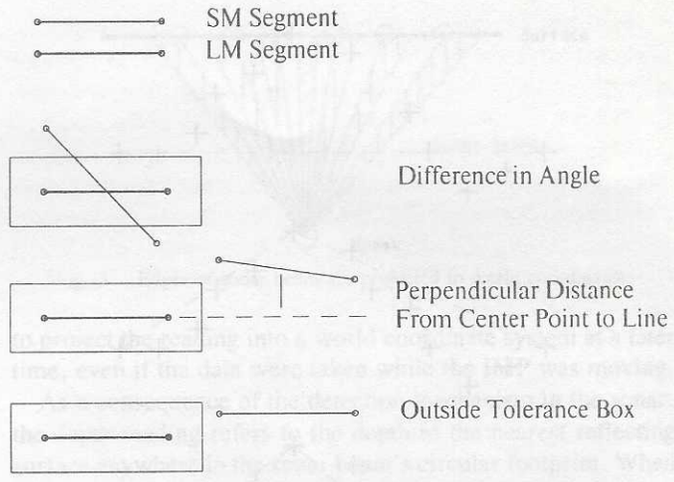

Fig. 5. Tests used in determining correspondence.

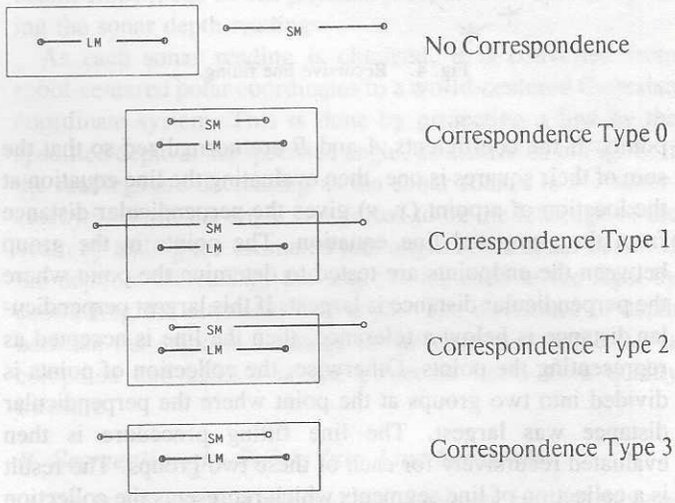

Fig. 6. Correspondence types.

1) Is the difference in angle between SM and LM less than a tolerance (currently 15°)?
2) Is the perpendicular distance from the center of SM to the line equation of LM less than a distance tolerance (currently 2.0 ft)?
3) Does SM pass through a box formed around LM? This is a fattened box, formed by adding a tolerance (0.5 ft) to the largest x and y coordinates of the segment LM. There are five possible outcomes, illustrated in Fig. 6. These are
   a) no overlap (segment rejected),
   b) both endpoints of SM inside box (correspondence type 0),
   c) both endpoints of SM extend outside box (correspondence type 1),
   d) first endpoint of SM extends outside of box (correspondence type 2), or
   e) second endpoint of SM extends outside of box (correspondence type 3).

The correspondence types are used to extend the composite local model segment during the update process.

4) Is the segment LM the longest found so far?

The correspondence function provides both the index of the corresponding segment and the type of correspondence.

D. The Update Process

The update process is the mechanism by which line segments from the sensor model, global model, and contact sensor enter and refine the composite local model. During each sonar scan, segments from the sensor model are matched to the current composite local model, and the result of matching is used to update the composite local model. In addition, as the IMP approaches a pre-learned place, segments which are visible in the global model from that place are also matched to the composite local model, and the result is used to update the composite local model.

The main functions of the update process are as follows.

1) Increase the confidence state of transient segments for which there is a corresponding segment in the sensor model.
2) Decay the confidence of segments that should be visible, but for which there is not a corresponding segment in the most recent sensor model.
3) Add newly observed sensor model segments and segments recalled from the global model to the composite local model.
4) Refine the vertex position of segments which are "reinforced" by the sensor model.

E. Marking the Visible Segments in the Composite Local Model

Segments in the composite local model for which there is no correspondence are only modified under two conditions.

- The segment was marked as visible during construction of the sonar model.
- The nearest point on the line segment is more than a given distance from the IMP's current position.

The second condition is a simple mechanism by which segments "fall off the end of the world" as the IMP moves away from them. The actual distance is relatively unimportant as long as it is beyond the sonar range and the current area of local navigation. Of course, the larger this distance, the more "extra" segments the system has to consider on each calculation.

The first condition establishes a "visible horizon" for the IMP. As each point is added to the Sensor Model, the function VISIBLE is called, for a point at the direction of the beam and the range of the sonar, to determine which segment in the composite local model should be visible. If a segment is found, the difference in angle between the beam and that segment is computed. If this difference in angle is not small ($<15°$) then that composite local model segment is marked as visible. Segments for which the angle of incidence of the sonar beam is very small are not detected reliably by the sonar and are thus not marked as visible.

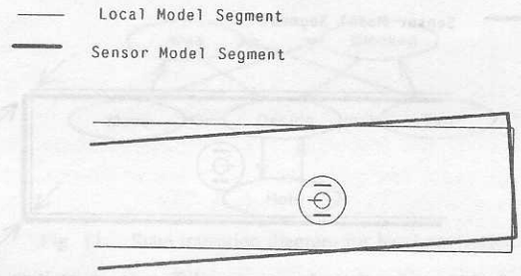

Fig. 7. Orientation error given by average difference in angle between sensor model segments and corresponding local model segments.

F. Updating the Vertex Positions and Segment States

The rules for updating the confidence states of segments in the composite local model have evolved by trial and error during experiments with the sonar data. Some principles that have emerged during these experiments are the following.

- Unconnected vertices should be extended when there is a correspondence of types 1, 2, or 3.
- The position of connected vertices has precedence over the position of an unconnected vertex.
- A segment is more stable when both its vertices are connected.

The actual rules for state updates are implemented as case statements based on the current state and then on the correspondence type.

After the vertex positions and states of the segments in the composite local model have been updated, segments from the sensor model for which there was no correspondence in the composite local model are added to the composite local model in the lowest confidence state (state 1). A relabeling process is then used to connect adjacent segments for which the vertices are very close.

VI. CORRECTING THE ESTIMATED POSITION

Local path execution and learning and updating the composite local model all depend critically on maintaining an accurate estimate of the IMP's current position. An instantaneous estimate of the IMP's position is maintained from the rotary position encoders on the IMP's wheels. This estimated position is monitored and corrected by a process based on comparing the sensor model to the composite local model. Before the composite local model is updated from the sensor model, the correspondence between the sensor model and the composite local model is used to detect and correct any systematic error in the estimated position of the IMP.

As each sensor model line segment is obtained, the correspondence is found to the most likely line segment in the composite local model. The difference in angle between these segments is then computed. When a sensor model has been constructed from a complete scan of the rotating depth sensor, the average error in angle is computed. This average error is computed from the difference in orientation between Sensor Model segments and the corresponding composite local model segments, as illustrated by Fig. 7. The sensor model is then rotated around the position of the IMP by this average error in

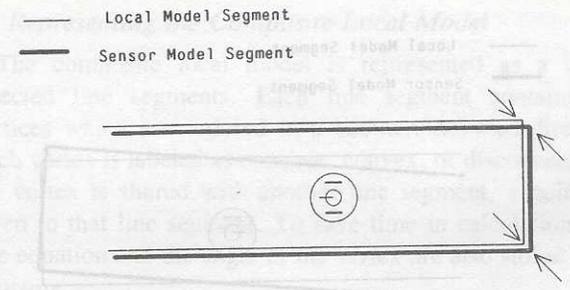

Fig. 8. Position error given by average difference in position between connected vertices in sensor model and corresponding connected vertices in the local model.

angle, as illustrated by Fig. 8, and the average error is subtracted from the estimated orientation.

Next, the average error in position is computed by computing the average x and y errors between connected vertices in the rotated sensor model and the corresponding vertices in the composite local model. This average error in position is then subtracted from the estimated position and from the position of the line segments in the sensor model. The segments in the composite local model are then updated to include the results of matching to the sensor model.

VII. GLOBAL PATH PLANNING AND NAVIGATION

Global path planning is based on the network of places, whereas local path planning and execution are based on the information in the composite local model. The global path planning process uses the network of places to determine the shortest sequence of straight line paths that will take the IMP to a specified goal point. Global paths are planned based on a network of "Adit" points which are connected by straight line paths. The path is then executed as a sequence of straight line movements.

A. Navigation Modes

There are three modes in which the IMP may travel.

Learn Mode: Limited exploratory movements in an unfamiliar environment, with the purpose of learning the environment.

Manual Mode: User specified motion executed by local navigation.

Automatic Mode: Autonomous movement to a named goal point in response to a command of the form "go to ⟨place⟩".

Learn mode permits the IMP to learn the global model from which it constructs the network of places. Automatic mode is designed to permit the IMP to execute navigation tasks in the learned environment. Manual mode is a default mode in which the IMP may travel to a visible point using only local straight line navigation.

B. The Network of Places and the Global Model

The learned domain of the IMP is represented in two related data structures: the "global model" and the "network of places." The global model is the collection of line segments observed by the composite local model while making a tour of the house in learn mode. The global model permits the IMP to recall the surfaces that it should observe at any location in the known world. An example of a global model constructed by the automatic learning process running on the simulator is shown in Fig. 9. The network of places is the structure which serves as a basis for global path planning. The network of places is obtained by dividing the free space in the global model into convex regions. A convex region has the property that any two points within the region may be connected by a straight line that remains entirely within the region. Thus a mobile robot may travel between any two points within a convex region by a single straight line motion. An example of the convex regions for the global model shown in Fig. 9 is shown in Fig. 10.

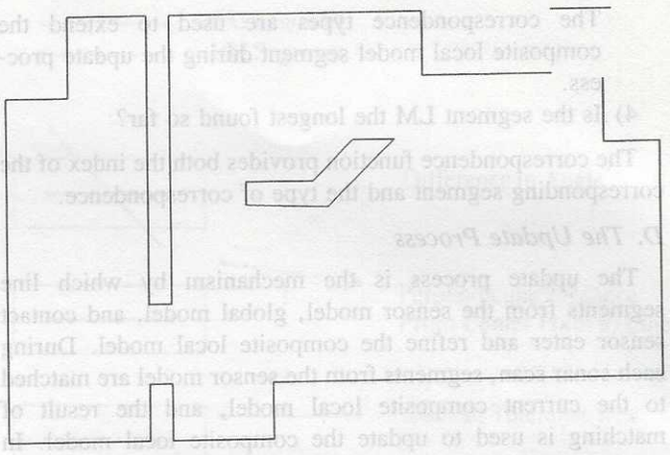

Fig. 9. Global model produced for typical floor plan used in simulator.

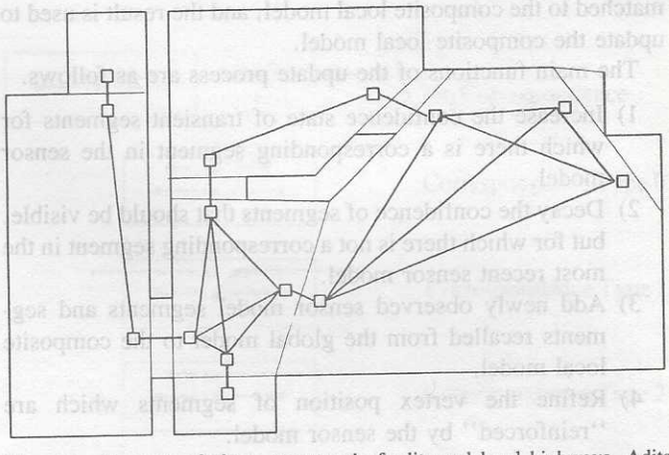

Fig. 10. Network of places composed of adits and legal highways. Adits shown as boxes.

Convex regions are constructed with an algorithm which is designed to maximize the area of the largest convex region [4]. A pair of navigation landmarks, called "adits" (an adit is the opposite of an exit), are created for each cut that is made to partition free space to create the convex regions. The adits are displaced to the sides of the cut so that the robot will pass through the cut at a roughly perpendicular angle. This protects the robot from grazing the edges of door ways and tight spaces.

Convex regions are shrunk by the diameter of the robot to

represent the free space in which the robot may travel. The space inside a doorway after shrinking forms a special region called a "doorway region." Doorway regions are not guaranteed to be convex. Each adit is connected to the adit on the other side of the doorway and to all other adits within its convex region, as shown in Fig. 10. In this figure, the adits to the convex regions are illustrated with boxes.

The network of places is a three level structure. At the top level are a list of user-defined "named places." At the middle is a list of convex regions. Each named place points to a convex region, and each convex region contains a list of named places. Each convex region also contains a list of adits. The adits serve as landmarks to global path planning and execution. The convex regions serve as "legal highways" for planning paths to any named place.

C. Global Path Planning

A global path is planned as a sequence of adits which will take the robot from its current convex region to the convex region which contains a specified goal point. A command of the form "go to ⟨place⟩" provides a pointer to a convex region that in turn provides a list of possible goal adits. The adit closest to the named place is chosen as a goal for path planning. Knowledge of the current convex region gives a list of adits from which to start the path. The nearest adit is selected as a starting adit. The shortest path through the network of adits is determined using a version of Dijkstra's algorithm [1] which halts when a path to the desired goal place has been found. If the start (or the end) of this path leads through two adits in the same region, the first (or last) adit is dropped from the path. Global path execution is then reduced to a three step process in which the IMP 1) moves to the first adit in the path, 2) moves from each adit on the path to the next, and 3) then moves from the last adit to the goal place.

VIII. LOCAL NAVIGATION

A local straight line path is executed by a finite state process which monitors the position of the robot to assure that it remains on the desired straight line within a tolerance. This process also monitors the local model to assure that no unexpected obstacle blocks the path. A recursive obstacle avoidance algorithm is used to plan a path around unexpected obstacles.

A. Local Path Execution

Straight line movement to a goal point is monitored by a relatively simple finite state process. The states of this process are the set {HOLD, DECIDE, TURN, MOVE, WAIT, BLOCKED}. The state transition diagram for this process is shown in Fig. 11. The process waits in the HOLD state for a goal from the global path execution process. When a goal is received, the IMP enters the DECIDE state. In the DECIDE state it first tests the distance to the goal. If this distance is less than a tolerance, it returns to HOLD. If the distance is above the tolerance, the difference in angle between the current heading and the goal is tested. If this angle is above the minimum resolution for turning, the IMP enters the TURN state; otherwise it enters the MOVE state.

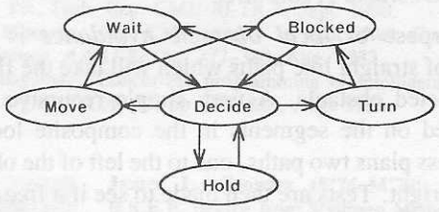

Fig. 11. State transition diagram for local navigation.

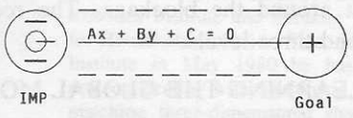

Fig. 12. Legal highway for path execution.

In the TURN state, the IMP turns toward the goal point until the difference between the estimated orientation and the direction to the goal point falls below the minimum turning resolution. The IMP then enters the WAIT state to make a complete sonar scan and verify the current estimated position. The sonar scan results in an update in the estimated position and orientation, even if there is no change in the estimated position or orientation. The call to the function "Set-Estimated-Position" signals the completion of the scan and causes a transition back to the DECIDE state. If the IMP is not at the goal, and is turned toward the goal, control will pass from DECIDE to MOVE.

Upon entering the MOVE state, the IMP computes the equation of the line (the path equation) from the current location to the goal point. A cyclic process is then initiated in which the system moves forward while performing the following tests as rapidly as possible.

1) Verify that the distance to the goal point is decreasing. When the distance to the goal stops decreasing, the system returns to HOLD to wait for the next goal. If the distance is ever increasing and is larger than the goal tolerance, then the system will go into the WAIT state to take a clean view of the world.

2) Verify that the perpendicular distance from the current estimated position to the path line segment is below a tolerance. This test is illustrated by Fig. 12. It is performed by evaluating the path equation using the current estimated position, yielding the perpendicular distance to the path equation. If this distance exceeds the tolerance, then the IMP will go to WAIT.

3) Verify that there is a free path to the goal. This is done by projecting parallel line segments in the composite local model. If the path becomes blocked, the IMP will go into blocked state and signal for local path planning to avoid the obstacle.

If the IMP is in the MOVE or TURN States and its contact sensor is triggered, it immediately halts and enters the BLOCKED state. Entering BLOCKED triggers the local path planning procedures to plan a path around an obstacle. A low-confidence line segment is also placed into the composite local model to represent the obstacle.

B. Local Obstacle Avoidance

The purpose of *local obstacle avoidance* is to plan a sequence of straight line paths which will take the IMP around an unexpected obstacle. A very simple recursive process is used, based on the segments in the composite local model. This process plans two paths, one to the left of the obstacle and one to the right. Tests are then made to see if a free path exists in the composite local model from this point to the goal and from this point to the current position. If either path is blocked, the procedure is called recursively to see if it is possible to get around the blockage. The recursion is not continued beyond three levels.

IX. LEARNING THE GLOBAL MODEL

The global model is learned by a process which detects and follows segments in the composite local model using a set of pseudo-sensors which we have come to call "whiskers." These pseudo-sensors are implemented in the composite local model using the VISIBLE function. That is, a test is made to see if any line segments in the local model block a pair of points to the right of the IMP at a distance of 3 ft.

Learning begins by loading the current contents of the global model into the composite local model. The composite local model is then searched for the nearest potential starting point. A potential starting point must be a point to the left of line segment which meets the following conditions.

1) The line perpendicular to line equation and passing through the IMP's position (and the starting point) intersects with the composite local model line segment.
2) The function FREEPATH to the starting point from the IMP's position must return the value true.

The IMP moves directly to the nearest starting point and then begins tracking the wall. The IMP moves in 2 foot steps, calculating each move based on the current segment. After each move a sonar scan is made and then the composite local model is updated.

There are two conditions which can cause the IMP to stop tracking the current segment. The first is where a call to FREEPATH detects a segment blocking the next goal point. In this case the blocking segment becomes the new current segment. The other condition is that where the current segment is no longer visible to the right. In this case the human supervisor is asked if it is OK to turn right. If the answer is yes, the IMP proceeds in search mode, making a sequence of 1 ft moves and 30° turns to the right, while searching for a new segment to the right with its whiskers. If the answer is no, the IMP proceeds forward in moves of 2 ft, searching to the right with its whiskers for a new segment.

Learning terminates automatically when the system comes within 3 ft of the first goal while tracking a segment which corresponds to the first segment. Learning may be terminated by the supervisor at any time. On termination, the composite local model is loaded into the global model and the network of places is computed by convex decomposition.

An example of a learned global model and the path followed during the learning procedure is shown in Fig. 14. The line segments are a global model produced by the learning algorithm. The "+" 's represent the actual position of the IMP as it traveled. The "X" 's represent the estimated position. Errors in the position of segments produced by the sensor model were the source of error in the estimated position.

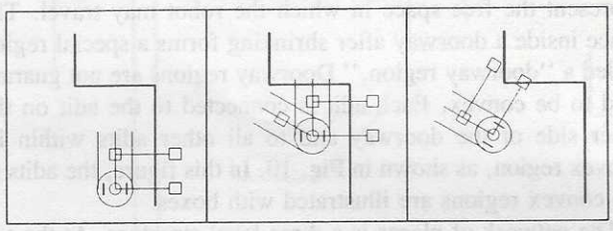

Fig. 13. Learning global model. Left frame illustrates whiskers used in detecting presence of current segment and FREEPATH calculation used to verify next move. Boxes show ends of whiskers and next goal point. Center frame shows new segment detected by FREEPATH function. Right frame shows IMP turning right in search of new segment to track.

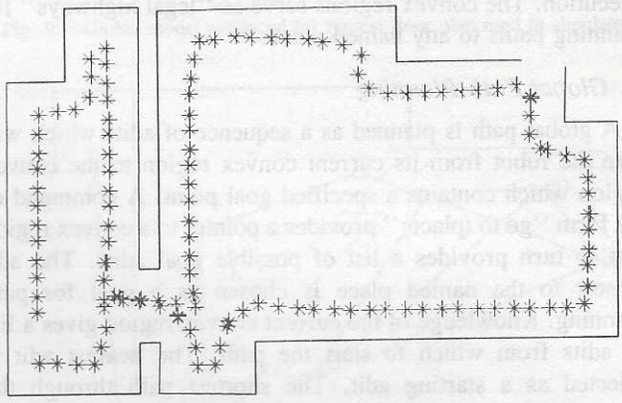

Fig. 14. Trace of IMP during simulated active learning. Line segments represent global model. "+" 's represent actual position of IMP as it traveled. "X" 's represent estimated position. Errors in position of segments in sensor model were only source of error in experiment.

X. SUMMARY AND CONCLUSION

This work describes a navigation system for an intelligent mobile robot. This system is based on maintaining a dynamic model of the external world (the composite local model) using a rotating depth sensor. A side effect of maintaining this dynamic model is an error vector which is used to maintain an estimate of the robot's position as it moves. The dynamically maintained composite local model also supports the functions of local obstacle avoidance, local planning, and learning.

A path planning technique has been described that is based on a pre-learned "network of places." The robot's domain is represented as a network of maximum-area convex regions. These convex regions serve as "legal highways" in which the robot may travel. The cuts which form these convex regions provide "adits" which serve as key points for planning paths through the known environment. Each straight line motion is executed by a finite state process which monitors motion using the composite local model.

These techniques yield an inexpensive navigation system that is suitable for indoor environments. This system plans and executes paths as a sequence of straight line motions. Such

paths are not necessarily optimal in the sense of being shortest; they are, however, a reasonable trade-off between efficiency and safety.

REFERENCES

[1] A. V. Aho, J. E. Hopcroft, and J. D. Ullman, *Data Structures and Algorithms.* Reading, MA: Addison-Wesley, 1983.

[2] R. A. Brooks, "Solving the find-path problem by good representation of free space," in *Proc. Nat. Conf. Artificial Intelligence, AAAI-82,* pp. 381–386, Aug. 1982.

[3] R. Chatila, "Path planning and environmental learning in a mobile robot system," in *Proc. ECAI,* Orsay, France, Aug. 1982.

[4] J. L. Crowley and R. J. Redpath, "An algorithm for maximum area convex decomposition," Robotics Institute, Carnegie-Mellon Univ., Pittsburgh, PA, Tech. Rep. in preparation, 1984.

[5] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis.* New York: Wiley, 1973.

[6] J. P. Laumond, "Model structuring and concept recognition: Two aspects of learning for a mobile robot," in *Proc. Eighth IJCAI-83,* pp. 839–841, Aug. 1983.

[7] T. Lozano-Pérez, "Automatic planning of manipulator transfer movements," *IEEE Trans. Syst., Man, Cybern.,* vol. SMC-11, pp. 681–698, Aug. 1981.

[8] H. P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Carnegie-Mellon Robotics Institute, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-3, Sept. 1980.

[9] H. P. Moravec, "The CMU rover," in *Proc. Nat. Conf. Artificial Intelligence, AAAI-82,* pp. 377–380, Aug. 1982.

[10] Y. Kanayama, "Concurrent programming of intelligent robots," in *Proc. Eighth IJCAI-83,* pp. 834–838, Aug. 1983.

James L. Crowley (S'74–M'79) received the B.S.E.E. degree from Southern Methodist University, Dallas, TX, in 1975, and the M.S.E.E. and Ph.D. degrees from Carnegie-Mellon University, Pittsburgh, PA, in 1977 and 1982, respectively.

He is a Research Scientist in the Carnegie-Mellon Robotics Institute and Director of the Laboratory for Household Robotics. Since joining the Robotics Institute in May 1980 he has completed three projects dealing with measuring, representing, and matching three-dimensional shapes for industrial applications. In addition to his work on navigation and world modeling for a mobile robot, he is currently involved in projects in dynamic 3-D scene analysis and in multiple resolution representation and matching. His research interests include mobile robot navigation, dynamic 3-D scene analysis, 3-D shape representation and matching, and multiple resolution representation and matching of shape.