CONF-860415--1

CONF-860415--1

DE8C 002209

Control Algorithms for Autonomous Robot Navigation

C.C. Jorgensen

Oak Ridge National Laboratory
Martin Marietta Energy Systems
Oak Ridge, Tennessee

September 20, 1985

* Research sponsored by the Office of Basic Energy Sciences of the US Department of Energy, under Contract No. DE-AC05-84OR21400 with Martin Marietta Energy Systems, Inc.

To be submitted to the ANS/ENS International Topical on Advances in Human Factors in Nuclear Power Systems, April 20-23,1986, Knoxville, Tennessee.

ABSTRACT

This paper examines control algorithm requirements for autonomous robot navigation outside laboratory environments. Three aspects of navigation are considered: navigation control in explored terrain, environment interactions with robot sensors, and navigation control in unanticipated situations. Major navigation methods are presented and relevance of traditional human learning theory is discussed. A new navigation technique linking graph theory and incidental learning is introduced.

INTRODUCTION

The rapid growth in high technology industries has produced new job environments not anticipated during the early parts of the twentieth century. In particular, workers may be exposed to toxic or hazardous situations for which there are no easy solutions. One result has been a dramatic growth in research to increase machine intelligence in robots by combining improved robot sensors with flexible control algorithms. For future industrial environments, the ability of a robot to navigate autonomously and to assimilate new information will be increasingly important. This paper considers application problems associated with this developing technology. Specifically, we will discuss how experimental robots calculate navigation parameters, how sensors interact with that calculation, and how elements of learning theory are beginning to be used for navigation in unstructured environments. A brief statement will be made identifying needs for increased cooperation between human factors research and robot hardware developers.

In the past, robot navigation has generally concerned how a robot moves in carefully controlled situations and plans paths through known environments to attain higher level goals. To understand the difficulties which must be overcome when a robot maneuvers in uncontrolled environments, let us first examine the information requirements of the task.

Consider first a human exploring an unknown area. A human uses large amounts of sensory data when reaching a goal. Balance, touch, vision, sound, and kinesthetic feedback are all integrated to form a composite dynamic model of the environment. The more unfamilier the situation, the more sensory data are needed to move accurately. If non-sensory information is available, its use can dramatically improve path planning. Stable reference points are more useful for orientation than references that may change position or have few unique characteristics.

In a similar way, depending upon the control requirements of a task, an autonomous robot must depend upon sensors to navigate and select information. Most commercial robotic systems depend heavily on predefined job environments that change only in prescribed ways. If navigation is required, the environment rather than the robot is usually changed to accommodate movement. Corridors can be kept clear, assembly lines can be prepositioned, parts can be preplaced, and in general, variations and unknown obstacles can be minimized.

When such restrictions are relaxed, however, the capabilities on-board a robot must be increased to compensate for unpredictability in the environment. The first step is usually to enhance the sensors that provide data on the environment and internal state of the robot. Because spatial relationships between the robot and the world may now change, two problems will arise. First, we do not necessarily know what the environment looks like. This requires the generation of a sensor map. Second, we do not know where the robot really is at any given moment because in real world situations both robot position and the environment may change relative to each other. The latter involves determination of self-location. Until recently, experiments in autonomous navigation have emphasized mapping rather than self-location. As a result, in research studies, there was usually a fixed reference point somewhere in the environment which robot sensors could detect and use to calculate an accurate self-location. Location has been calculated by homing on a radio source, following a path in a magnetic floor tape, using triangulation on infrared emitters or more recently by visual sighting on specially constructed optical landmarks (Julliere, Marce, and Place 1983). However, such approaches are not practical for uncontrolled tasks like emergency repair of nuclear power plant components. For these situations, a robot must also use relative references as well as stable references. An example of a relative reference would be to sight on a pipe break in a damaged building. In contrast, a stable reference would involve the use of a building blueprint. Once a reference point has been chosen and sensors have generated a local area map, the control algorithms which get a robot from one place to another must still be specified.

A variety of approaches have been proposed to solve robot navigation control equations for explored environments. Called "find-path" problems, they are defined as follows: given an initial robot location and set of known obstacles, find a continuous path from a start position to a goal position which avoids collision with obstacles along the way.

One approach called the configuration-space method by Lozano-Perez (1981) divides navigation space into zones which a reference point of a moving object can occupy without the object colliding with any obstacles. Paths are calculated that make maximum use of open areas between the reference point and a specified goal. For example, Moravec's Mars Rover (1980) bounded three dimensional obstacles on Mars with two dimensional circles stored inside the robot as a spatial map. These circles were then enlarged mathematically to assure adequate clearances for the robot edges. Navigation paths were calculated by solving for tangents to these circles. Udupa (1977) was concerned with robot manipulators and chose to bound obstacles with complex polyhedra. He solved three dimensional paths as graph connections to object corners.

Other methods approach robot navigation using mixes of previously stored information and new sensor data. Such methods characterize navigation in terms of local and global control. Local control addresses problems of immediate obstacle avoidance whereas global control considers broader information like the plan of a building or long-term goals. For example, Crowley (1985) described global path planning based on stored information kept in relational nets of places. Global navigation was defined as traversals along "legal highways" in known areas with local movement based primarily on avoidance using sensors. Recent work has merged robot exploration and region learning [Jorgensen, Hamel, and Weisbin (1985)]. Learning has been a concept used in other areas of robotics such as manipulator movement (e.g., Dufay 1983) but has only recently been linked to navigation control in unexplored terrain (Iyengar et al., 1985).

<u>Navigation Control in Known Environments</u>

We will begin the methodological presentation with a discusion of find-path algorithms and select a simple case where the obstacle locations have been determined. Figure 1 shows a room with three obstacles placed between a robot and a goal point. The dark borders of the obstacles represent two-dimensional boundary projections of the three-dimensional object shapes. Because the exact locations of obstacles are known, navigation can be performed as follows. A line is projected from the robot to the goal. If the path is clear, the robot uses the line to calculate an angle of turn and a travel distance using the Pythagorean theorem. However, such a simple algorithm will cause problems. For one thing, no allowance is made for the width of the robot, and it will collide with edges of obstacles. Consequently, we must take into account the required clearances and include sources of potential measurement imprecision. Second, if a straight line is not possible, the robot requires a procedure to obtain other paths and select one from among many possibilities much like a traveler would select intermediate stops on the way to a destination.

One way a find-path algorithm is implemented when terrain is known is shown in Figure 2. Here the known boundaries of the obstacles are enlarged (bloomed) by an amount equal to one-half

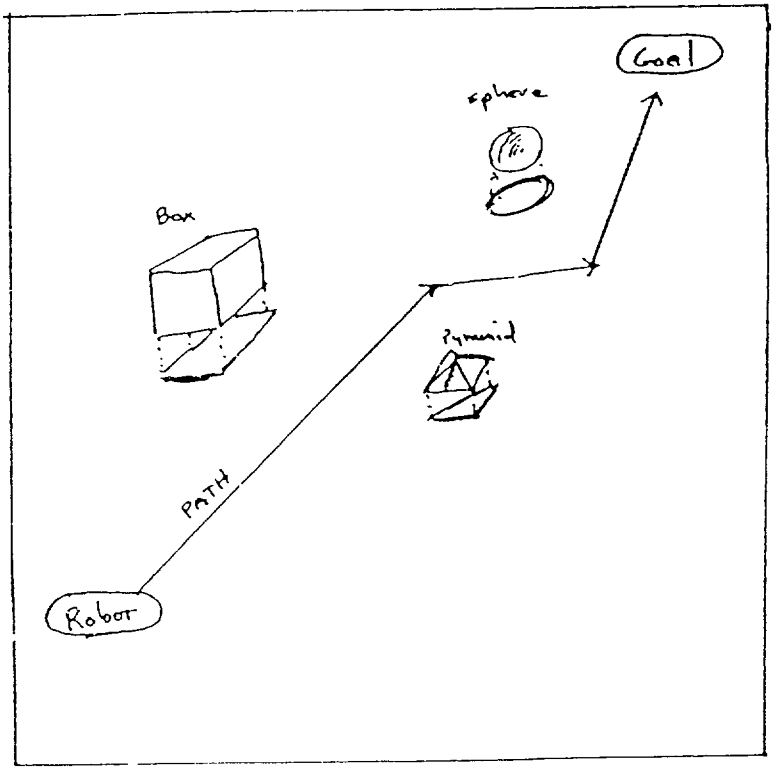
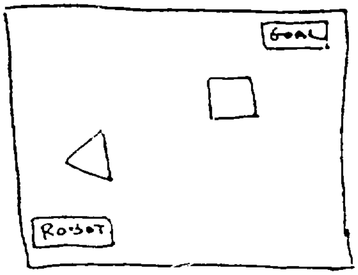
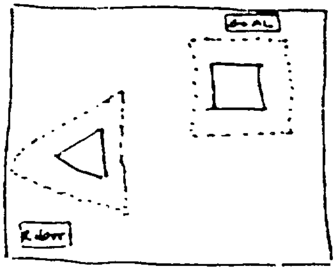


Figure 1

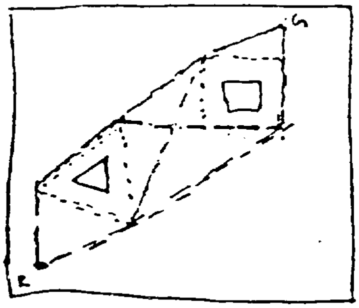
Three dimensional objects projected on a two dimensional surface



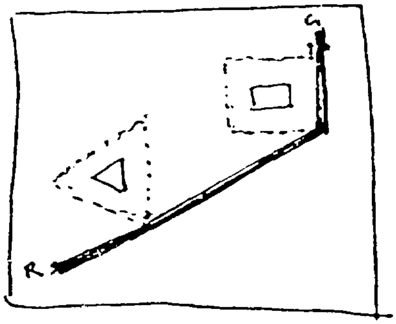
Original
obstacles



Bloomed
Obstacles [illegible]



Shrinking robot to
A point AND
projecting path lines (in blue)



Selecting the
shortest path (in green)

The Find-Path Problem

Figure 2

the diameter of the robot plus an extra amount for uncertainty associated with the robot's position. At the same time the boundaries are enlarged, the robot is mathematically shrunken to a single point. If an object obstructs a straight-line path to the goal, potential lines are drawn from that point to each of the vertices (edges) of the expanded obstacles in direct line of sight of the robot. From each of those points, new lines are drawn to each vertex of obstacles in their line of sight and so on until a line of sight direct to the goal has been obtained. All paths that can reach the goal from the robot's current position are converted into a graph of nodes and edges where each edge is a path segment. Finding the best path requires examining path lengths from the start node to the finish and selecting the shortest sequence. Optimal solutions for find-path problems have been developed which permit the answer to be obtained very efficiently (e.g. Dijkstra's algorithm used in the artificial intelligence community for the search of decision graphs).

If the available movement corridors are very narrow, more complex algorithms must be used to calculate robot rotations. One such case is called the Piano Mover Problem (see Figure 3) (Schwartz and Sharir 1983). In the find-path problem above, the robot was treated as a point rather than a polygonal body with irregular sides and appendages impacting the navigation space. The robot was given a boundary area which was equal to the widest area the obstacles might occupy. Such an approach works only if there is plenty of maneuvering room. In the case of the Piano

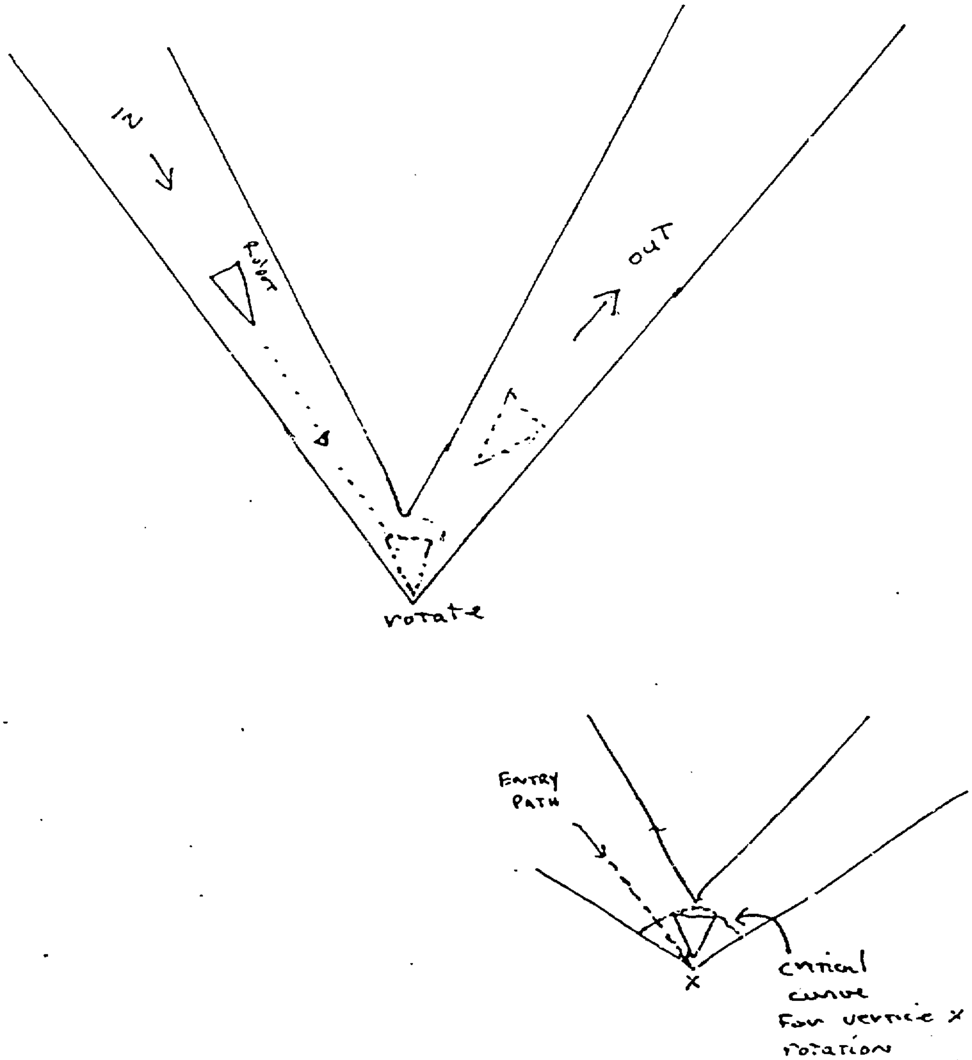


Figure 3
The Piano Mover Problem

Mover Problem, the corridors the robot would traverse narrow so that the robot must also rotate to squeeze through clearances the way a piano mover must make turns and rotations to climb stairs, go through doors, or go around corners. The complexity is increased because in addition to finding a continuous motion that will take a robot from a given start point to an end position, the robot is also subject to geometric constraints during motion. The constraints do not permit any part of the robot body to come in contact with obstacle edges or walls. The task could be artificially simplified by imposing restrictions on the range of allowed robot motions by forcing the robot to move in a fixed orientation or permit orientation changes no more than once during a path traversal, but in general the unrestricted problem is solved as follows. Each corner of the robot perimeter is labeled and treated as an axis around which the robot can rotate. The map of the navigation area is divided into regions of open space formed by intersections of the lines connecting the objects, room walls, and corners. Each region is separated from another by "critical curves" which are the set of points generated when placing each vertex of the robot perimeter at an intersection point of two regions and rotating the robot around that axis until an intersection occurs with region boundaries. The set of intersecting curves formed for all the robot vertices forms a finite connectivity graph which contains all possible boundary-crossing rotations. Algorithms are then applied to the connectivity graph to select a "path" consisting of axis rotations which permit the robot to cross from one region to another. In Figure 3, a simple case is shown where the robot is shaped like a triangle and the corridor to be navigated is bent at an angle. To traverse the boundary, the robot must turn right at the first boundary area and then move backward down the second corridor in order to fit through. Other factors can overshadow such tight maneuvers. Specifically, one factor is the quality of the robot sensors. Therefore, for an autonomous robot to handle more complex or unexplored environments, the interplay of the control algorithms and the sensor capabilities must also be well understood.

Environment and Sensor Interaction

In constrast to known areas, navigation in unexplored terrain will lead a robot to encounter unexpected events such as mazes and dead ends. To maneuver in unknown areas, the robot must use dynamically created sensor maps to select destinations and minimize the expenditure of resources. Once an obstacle location has been determined, the robot must still find the best possible path to a destination. However, if we assume the presence of obstacles that make a straight-line traversal impossible, it may be necessary for the robot to turn and maneuver using only partial or incorrect information. Such situations often occur in real-world environments like large industrial plants or buildings. In these cases, sensor control becomes as important as movement planning and it is necessary to understand how robot sensors interact with both the environment and the control algorithms.

Robot sensors come in many forms and include stereoscopic vision systems, fixed and mobile sonar range finders, laser range finders, touch, stress and torque sensors, and collision detectors. For exploration, the most frequent use of sensor data is for edge detection. We will briefly consider some sensors used heavily in robotics: vision, laser range finders, touch sensors, and sonars. Particular attention will be given to sonar systems.

Vision systems usually encode picture edges from matrices of grey scale values which are connected through gradient seeking algorithms that consider reflectivity, texture, and shading to produce skeletal representations of scene objects. The skeletal boundaries are then used to direct turn angles and grasping orientations of end effectors. Simultaneous use of multiple cameras permits optical parallax to be used to estimate distance (for an excellent review, see Besl and Jain (1985)).

Lasar range finders allow precise location of edges and are often used in conjunction with other sensors such as vision. Touch sensors permit obstacle avoidance by following edges and work well if the objects are not highly irregular.

Sonars have proven particularly useful in navigation research because of relatively low cost and the capability to cover large areas rapidly but they also have limitations. First, most low cost sonar devices function by sending a multifrequency or "chirped" sound pulse from a transducer outward in a cone-shaped wavefront. The difference between the time of emission and return is measured and an estimated distance calculated on the basis of how far the wave could travel in one-half the period.

Several conditions occur when a robot uses sonar information to construct spatial distance maps from different scanning positions. First, sonar is sensitive to temperature changes. For example, if a sonar were calibrated at 80 degrees F and the actual room temperature was 60 degrees F, a measured range of 35 feet would be overestimated by 7.8 inches simply because of temperature. If that 7.8 inches overlaps with the position of a solid object, the difference could provide a shocking experience for a moving robot and must thus be compensated for if an autonomous robot will operate in varying climatic or temperature conditions. For example, if a robot were used to locate a steam line break, temperature sensors on the robot should also be connected to navigation sensors to adjust readings as the robot explores the building.

Second, sonar is vulnerable to specular reflection and interacts with the texture of materials. This interaction occurred one day in our laboratory when we were going to demonstrate a small mobile robot for some visitors and decided to give the obstacles colorful coats of new paint. The high gloss was attractive to the human eye but also very reflective to sonar. The result was that if the obstacles were not hit almost head-on by the sonar beam they vanished from the navigation map. The robot then rammed objects instead of going around them. After trying cardboard, metal, and other coverings, the highest specular reflection was provided by simple plastic bubble wrap. Visitors now see colorful boxes but they also have a fat layer of bubble packing. In real world situations such as movement through highly reflective metal rooms, a robot may have to switch between vision and sonar systems to navigate properly. Thus issues of sensor fusion are extremely important for applications using autonomous industrial robots.

The detectability of reflected sonar also depends on signal energy and frequency. Frequencies that are useful in medical imaging such as ultra sound are not really practical for robotics because they take advantage of the density of the propagation medium which is usually a fluid or tissue. Most sonar systems using air rely on a carefully selected subset of frequencies designed to minimize absorption by typical materials. Under some circumstances the frequency will be inappropriate even for head-on readings. An example of this effect occurred in some early experiments using robot manipulators that attempted to grasp polyurethane foam blocks with extremely high sonar absorbency. For all intents and purposes, the blocks became sonar invisible.

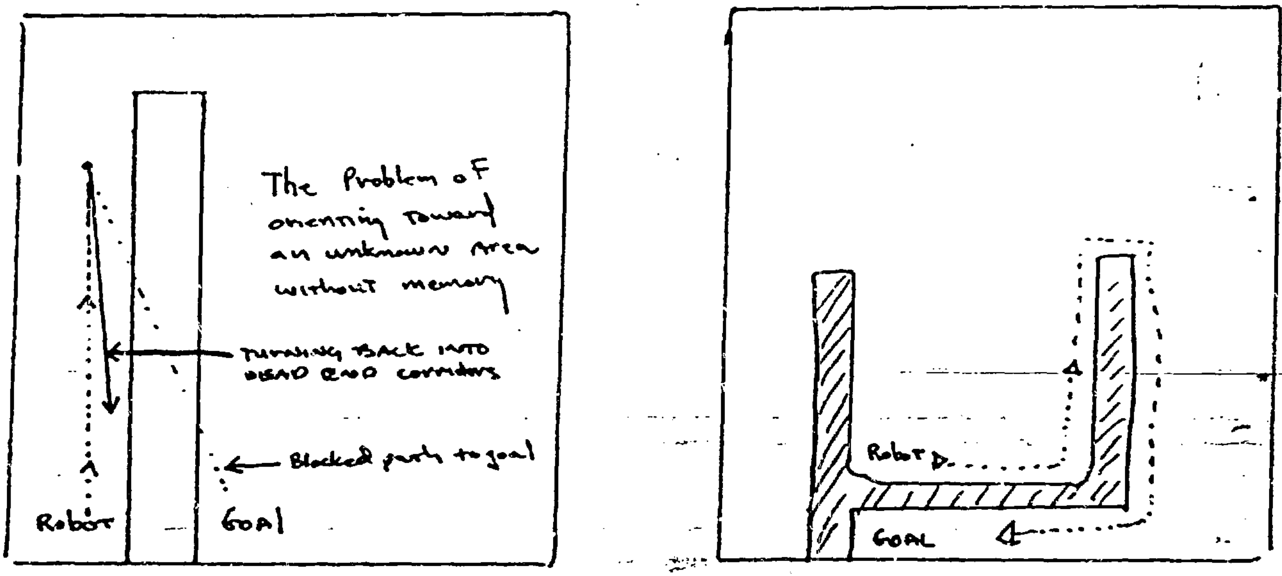
Other sonar problems result from beam shape. The output of a sonar transducer is actually a cone, like the beam of a flashlight. Without a focusing horn a typical sonar cone is about 35 degrees wide. Therefore, sonar map generation algorithms have to take into account that the leading edge of the cone will contact a barrier well before the center axis of the transducer. If a false angular reading is not corrected, the area map that a robot uses to navigate is distorted.

Since robot designers can't completely compensate for the multitude of sensor effects, it is obvious that several sensor types should be used in conjunction with navigation. This is particularly true of exploratory navigation when no previous maps exist of the terrain.

Navigation Control in Unexplored Terrain

When only locally generated maps are available for navigation, the find-path methods above must include more variables such as uncertainty introduced by sensor error, partial object shapes, and tentative destination areas. In the latter case it may not be possible to reach a goal, and numerous unplanned changes in course may be required depending upon what a robot encounters. It is also no longer possible to guarantee that the most efficient path to a goal is being taken. This is because a robot can not always see a goal area. A maze might be generated by rows of recently moved boxes, outdoor canyons, or tangled equipment.

It is not always easy for a robot to recognize a problem situation. Figure 4 shows a simple maze problem. Suppose the robot is given a control algorithm like the following. When in a new area first turn toward the location of the goal you wish to reach. Take a sonar reading to see if the path is clear.



The Need for Memory

Using Edge Following

Figure 4 TWO MAZE PROBLEMS

<u>If a path is clear move. If it is not take the first open path on either side of the line. Go one half the distance to the goal. When you arrive at that location turn back toward the goal and repeat the process.</u>

At first glance such an algorithm would appear usable. The first clear path closest to an ideal straight line is always the one taken. The half distance criterion also assures that if the robot is far from the goal it will move to it rapidly and will make smaller, more careful moves as it gets closer. What is wrong is that the robot has no memory. Look again at Figure 4. The robot's goal is directly on the other side of a wall. If the robot follows the initial algorithm, it will scan the corridor and select the first open move half way to the goal, that will be after about a 90 degree left turn. The robot will begin to move up the corridor away from the goal. After a short distance, the robot will be far enough so that half the distance can be traveled by making a turn back toward the goal. What happens? The robot again moves into the dead-end corridor. As a result, without external memory the robot would recursively loop and never reach the goal. With a memory, previously blocked areas can be set "off limits" for a time so the robot is gradually squeezed out of dead-end situations. Of course there are many ways this problem could have been dealt with. The real difficulty is assuring that a general purpose navigation algorithm considers all the possible traps that can be generated by deficiencies in the robot. Subtle complexities can hide in seemingly simple situations like a need to consider multistep memory to avoid recursive loops, a need to explicitly consider trading off distance traveled versus angle turned, and checking sonar maps to handle changing reflections.

Other problems occur when navigation environments change quickly over time. Traversal may require continuous creation of new goals because unexpected obstacles invalidate previously formed navigation plans. Consider a man walking down a hall who is preoccupied with a schedule and takes only occasional glances to determine if something is in his way. A glance may work as long as the unexpected does not occur. As the number of moving people in the hall increases faster and more frequent glances are required and the existing plan to walk straight ahead will have to be modified to avoid collisions. A robot in a dynamic environment is in a similar situation. Sensor processing speed must also be sufficient to recognize changes in the environment before a preplanned action results in a catastrophe. Consider the implications of a Mars rover that could only process the image of a cliff after rolling over the edge. The addition of multiple sensors to acquire information faster only increases the need for information abstraction algorithms to reorganize sensor data into forms useful for high level planning. It is beyond the scope of this paper to consider all of the implied control algorithms in detail. What will be briefly presented, however, is a recently developed algorithm which incorporates incidental learning into the navigation and exploration process (Iyengar, Jorgensen, Rao, and Weisbin [1985])

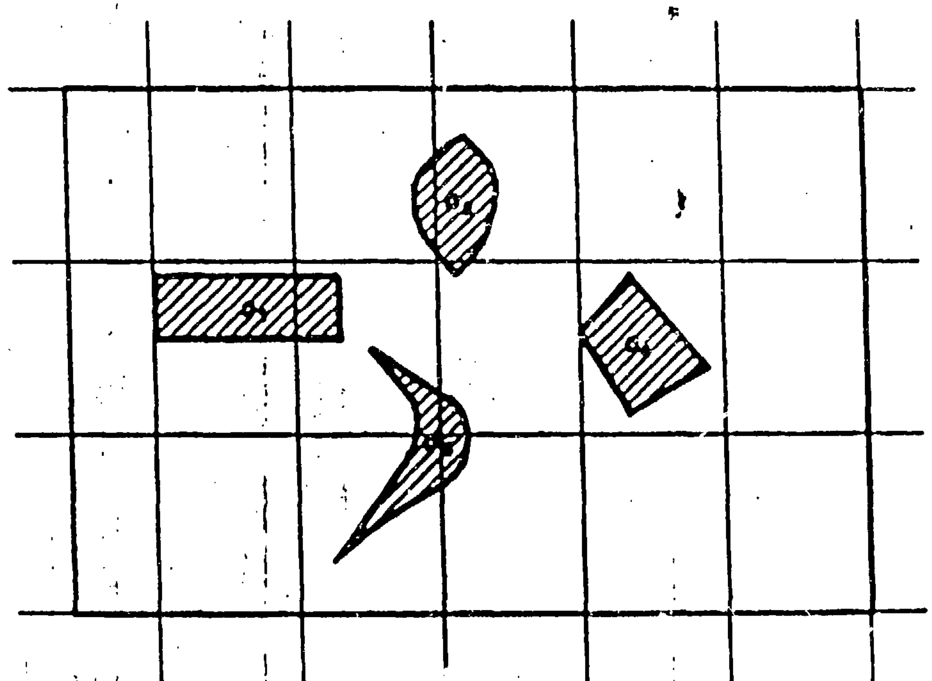
<u>Learning during Autonomous Navigation</u>

Ideally an autonomous vehicle should learn information about the local environment and at the same time build or modifiy a global world model which can be used for more general purposes. To date, most elements of learning as traditionally described by human factors literature have not been incorporated into navigation control. However, Crowley (1985), Giralt, Chatila and Vaisset (1982), and Turchan and Wong (1985) have addressed processes analogous to reinforcement (defined as a confidence increment added to sensor information), decay (lowering of confidence estimates with time), and sensory integration. At present, research dealing with other learning issues such as abstraction, transfer of training, selective forgetting, and proactive or retroactive inhibition is minimal or nonexistent. This is probably because the latter areas are heavily dependent on the data structures selected to represent sensor information. The data structures used to date have included Quadtrees, Voxels, Octrees, Directed Graphs, Linked Lists, or Object Graphs (e.g. Besl and Jain (1985)).

To illustrate how one learning concept (incidental learning) has been applied, an example will be given of a new method for navigation control. Iyengar et al. combine learning with specialized data structures based on graph theory (the reader is referred to the paper for a complete treatment of the algorithms). Learning begins by classifying information about a room which a robot explores. The room (Figure 5a) is assumed to contain obstacles that have not been previously stored in the robot's memory or detected by the sensor suite. As it is explored, information is stored in a special form of attributed spatial graph. The spatial graph maps the history of robot obstacle avoidance movements (Figure 5b) onto a two dimensional coordinate system composed of edges (the paths travelled) and nodes (stopping points, turning points, or path intersections). The spatial graph provides a real time data structure to record past movements, but is not efficient for planning future movements. This is because no inferences are made about the shape of the obstacles, which areas of the room require careful sensory analysis, or which regions are clear for maneuvering. Thus a second type of graph structure called a Voronoi diagram (Figure 5c) is used to bound obstacles with graph circuits which can then be labeled and associated with higher order learning processes. The more a room is explored, the smaller the obstacle bounding circuits become until finally all the room data is in a form readily usable by an autonomous robot for planning. The control algorithm is actually a mixture of local obstacle avoidance and region planning which transitions automatically from local movement to path calculation based on an incidentally learned global map.

Higher order learning processes require that the links between such data stuctures and control algorithms be well

ORNL-DWG. 85C-14083

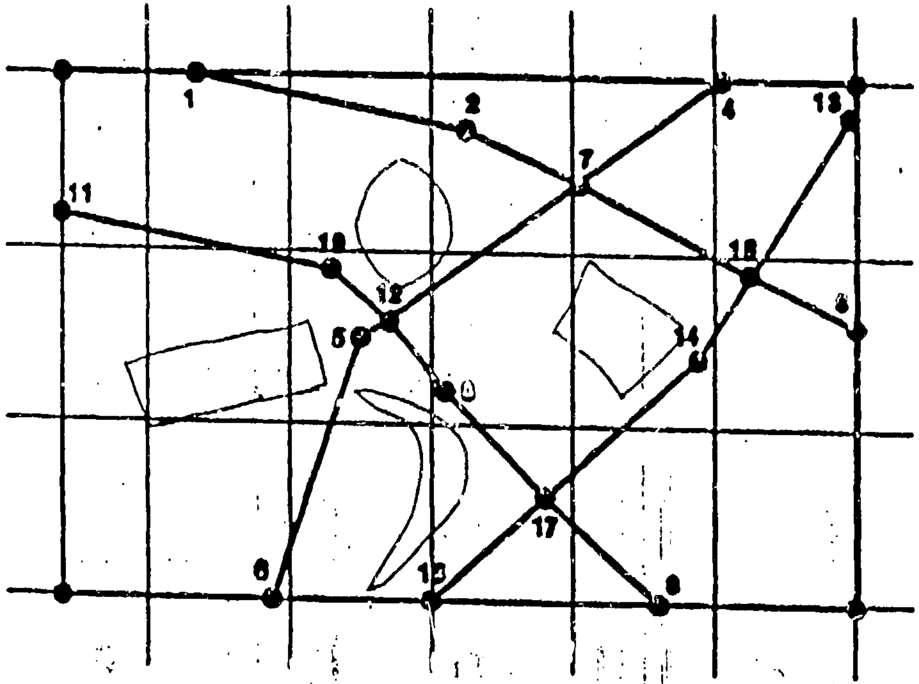


UNEXPLORED TERRAIN

Figure 15a - 4 obstacles before exploration

1399c.35

ORNL-DWG. 88C-18888



THE SPATIAL GRAPH

4 Paths From Previous Traversals around objects

Fig. 5b

ORNL-DWG 85C-12881

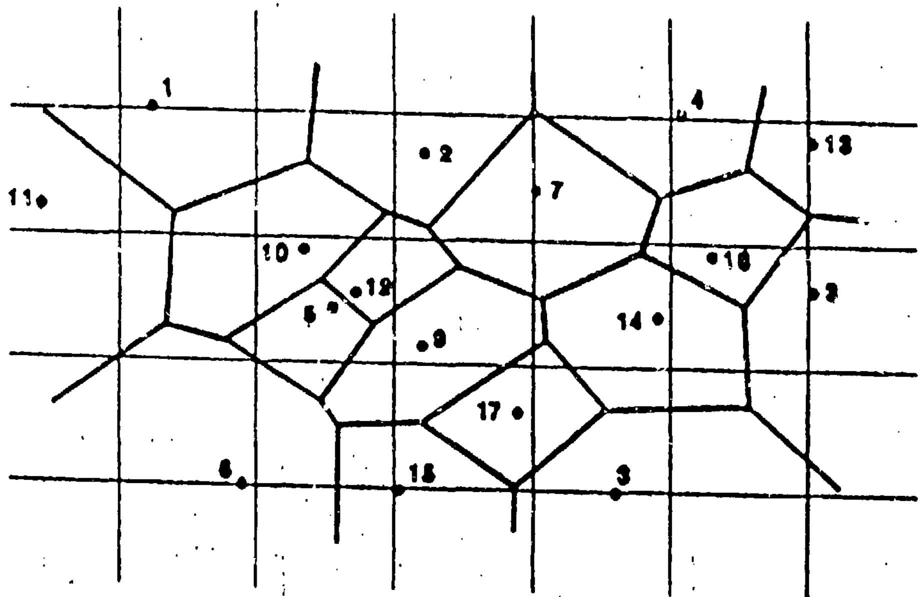
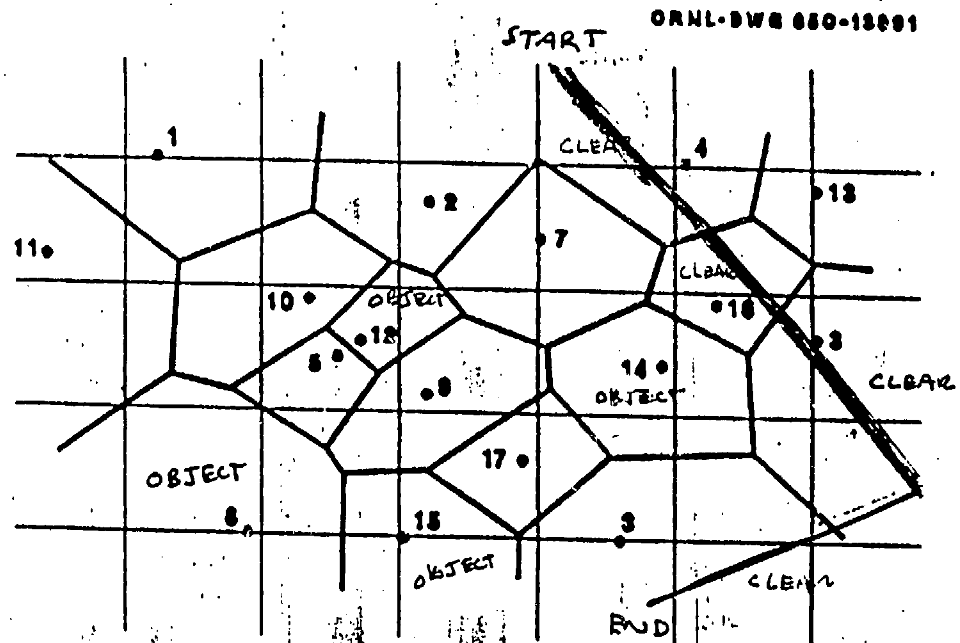


Figure 5c THE VORONOI DIAGRAM
Voronoi regions are created from previous path points



THE VORONOI DIAGRAM

Fig 5d

Each Point on 5b becomes a region with a label
A path goes through 'clear' areas

integrated. In the above example, the graph structures were built as a byproduct of exploration and sensor control. Because of that integration, a robot could be placed in an unknown environment and through undirected exploration generate a world map usable for efficient global navigation.

Although there is often not a direct analog between human and robot learning processes, there are enough similarities that psychological constructs may prove useful in future research on machine intelligence and control algorithm development. Some particularly promising areas include:

1. The use of selective forgetting to control the growth of extraneous information in a robot data base and to handle the real-time movements of obstacles,

2. The potential for associative memory processes to deal with massively parallel sensor input and data reduction, and

3. The use of perceptual psychology in the design of "sixth" generation fine grain computing architectures for robot brains.

The probability that human factors will have increasing application in robotics appears high not only in the traditional areas of the man/machine interface, but also in the developing overlap between machine and human intelligence characterization. As the robotic control algorithms and data structures grow more sophisticated, an effective understanding of human information handling will grow in importance. Traditional human factors techniques such as task analysis may find new application in robot task specification with a corresponding growth in the partnership between hardware engineers and psychologists. Indications of a potential partnership are already occuring in teleoperations with the handling of hazardous materials in nuclear facilities. The extention of that technology to include increasing machine autonomy seems a natural course.

Conclusion

This paper has considered methods for navigation control of future autonomous robots. In particular, it emphasized the calculation of autonomous movements in unexplored or unanticipated environments. The paper started with movement in known environments and increased problem complexity to include sonar sensor limitations, environment mapping, planning in unexplored environments, and learning. The relation between a traditional human factors learning concept and robot navigation was considered through a brief example of a newly developed incidental learning procedure using Voronoi diagrams to plan paths. Some relevant future research areas were identified. How well that research can be applied in nuclear power plants may well depend on a multidisciplinary partnership between human factors and engineering technologies.

References:

1. M. Juliere, L. Marce, and H. Place. "A Guidance System for a Mobile Robot, " 13th International Symposium on Industrial Robots," ROBOT 7, Vol. 2, April 1983.

2. Moravec, Hans P. "Obstacle Avoidance and Navigation in the Real World by a Seeing Robot Rover," Stanford AIT-340, September 1980.

3. J. T. Schwartz and Micha Sharir. "On the Piano Movers Problem, the Case of a Two Dimensional Rigid Polygonal Body Moving Amidst

Polygonal Barriers." Communications on Pure and Applied Mathematics, Vol. xxxvi, 345-398, (1983).

4. Lozano Perez, T. "Automatic Planning of Manipulator Transfer Movements," IEEE Transactions on Systems, Man and Cybernetics, SMC-11, 1981, 681-689.

5. Crowley, J. L. "Navigation for an Intelligent Mobile Robot," IEEE Journal of Robotics and Automation, Vol. RA-1, No. 1, March 1985.

6. Dufay, "Apprentissage par Induction en Robotique- Application a la Synthese de Programmes de Montage," (in French) these de 3 eme Cyde, INPG Genoble, June 1983.

7. Everett, H. R. "A Multi-element Ultrasonic Ranging Array," Robotics Age, July 1985, pages 13-20.

8. Udupa, Shriram M. "Collision Detection and Avoidance in Computer Controlled Manipulators," Proceedings of IJCAI-5, MIT, Cambridge, Ma., 1977, pp. 737-748.

9. Iyengar, S. S., Jorgensen C. C., Rao, S. V. N., and Weisbin C. R. "Robot Navigation Algorithms Using Learned Spatial Graphs," ORNL Technical Report ORNL-TM-9782, August 8, 1985.

10. Besl, P. J., and Jain, R. C. "Three Dimensional Object Recognition," Computing Surveys, Vol. 17, No. 1, March 1985.