ALGORITHM OF NAVIGATION FOR A MOBILE ROBOT

D.M. Keirsey[†], E. Koch[*], J. McKisson[*], A.M. Meystel[*], J.S.B. Mitchell[†]

[*] Department of Electrical Engineering
University of Florida
Gainesville, FL 32611

[†] Hughes Research Laboratories,
Malibu, CA 90265

Abstract

This study describes the theoretical and practical aspects of the design and computer simulation of a heuristic based navigation algorithm. An algorithm is developed which provides a convenient testing system for generalized navigation strategies on a fixed map which may be known or unknown to a system. A variety of maps are simulated and the navigation results are compared.

Introduction

The multi-level structure of path planning and execution propounded in[1,2] provides a basic framework for dealing with problems in the control of autonomous vehicles. The hierarchical approach enhances the flexibility of feedback loops[3], and also allows to clearly separate the planning of motion from the performance of motion.

There are three basic levels of path planning and execution:

1. The Planner - a global path planner, uses a priori knowledge (a map) to plan a plausible route.
2. The Navigator - a local path planner, uses the plan of the Planner a guide, but provides more precise routing according to locally obtained terrain information.
3. The Pilot - execution of simple vehicle movement routines.

This paper will concentrate on the navigator level of planning.

The difference between the task of planning a rough path by assigning a list of immediate goals or milestones and the task of navigating the rough path is well known in the practice of Artificial Intelligence (AI) mobile systems. The pioneering work of SRI produced "Shakey"[4], generated the A* search algorithm. The results has been expanded upon by JPL's Mars rover[6-10], the French[11-13] and the Russian researchers[14-17]. Hans Moravec[18-21] has obtained another pioneering result in the area of autonomous vehicle navigation. The research of his cart has provided priceless experimental material used by all groups working in this area.

Many of the results are obtained while solving a subproblem of navigation: obstacle (or collision) avoidance which usually implies the planning of paths to avoid intersections of the moving part with any obstacle ("collision avoidance"[22-24]). Many different approaches are used but most are based somewhat on the concept of "growing"[24] the obstacles and shrinking the moving part to a point. Another concept is used by the ROBNAV program[25]: the direction of the next motion step "biased" against corner cutting.

Obstacle models for the different navigation techniques should be considered for an understanding of some of the algorithms. Many navigation algorithms define either the traversable or the nontraversable areas of the map. Objects are often modeled as polygons defined by lists of vertices. A comprehensive treatment of obstacles in three dimensions is given by T. Lozano-Perez in[22].

We are particularly interested in applying the techniques used by previous researchers for the solution of the problem of obstacle avoidance. The following two different cases will be considered as a background to approaching the problem of autonomous navigation for obstacle avoidance.

Case 1 - The map of the terrain is known and the vision system delivers the information which is required primarily for the identification of mobile robot location and proper orientation.

Case 2 - The map of terrain is unknown, the goal is not visible (however, the direction and the distance to the goal is known) and obstacles may emerge unexpectedly. The location and orientation determination are performed by some standard procedure (or device) not considered here. The mobile robot is to find a path from its initial position to a known goal position through a terrain with

unknown obstacles. The mobile robot is equipped with a vision system.

Let us consider these two cases in more detail. The problem of trajectory tracing within the known map does not seem to create too many difficulties. Within a well-defined map (see Figure 1) the entire set of possible trajectories (defined on some finite set of nodes) can be described, then the most preferable path (i.e., the shortest) can be determined. The methods of dealing with such problems are thoroughly studied in the literature.

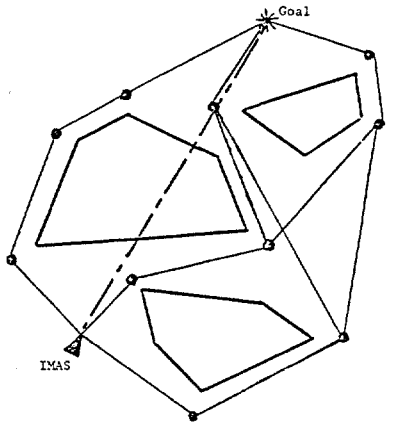


Figure 1. Example of Known Map

In the second case, the actual locations of at least some obstacles are unknown, and in some cases may be inconsistent with given information (in Figure 2 bold lines describe the visible world).

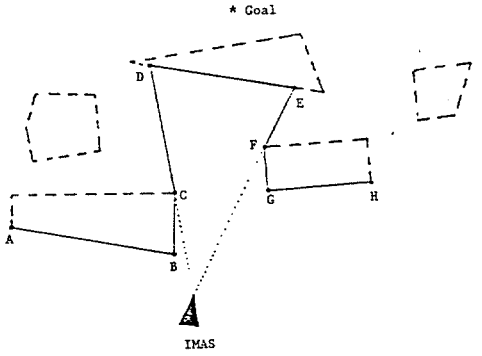


Figure 2. Example of Unknown Map

The broken line ABCDEFGH obscures the area of possible activities from the sensor system. The only thing that can be determined is the immediate interference of the obstacles with the line of sight path. Beyond the obstacles nothing is known, so no path can be preplanned[8].

The following considerations are based upon a conceptual model of AI for a mobile robot consisting of PLANNER, NAVIGATOR and PILOT[2]. The Knowledge Data Base (CARTOGRAPHER) contains all initial information and the sensor data.

Navigation

The NAVIGATOR performs the following procedures:

- strategic tracing, which is defined as developing on-line, a trajectory following the milestones assigned by PLANNER;
- tactical planning, which is defined as on-line follow-up actions and correction of the trajectory under the circumstances revealed after the actual motion has begun;
- communication with Perceptual System and KDB of CARTOGRAPHER, which means that the "sensor-motoric" activities should be determined by a command from NAVIGATOR, and some lacking information can be obtained as a result of actual motion (which may deviate from the initial plan);
- call for the replanning of the milestones on the level of PLANNER;
- providing current list of commands to the PILOT including the constraints for the optimization.

In order to create the appropriate string of actions, two similar algorithms were simultaneously developed at the University of Florida and the Hughes Research Laboratory. We will name this algorithm WSA (Wandering Standpoint Algorithm). WSA presumed an ability to judge the situation for three cases: when the map is known, partially unknown, or not given at all. In practice, the amount of information is increasing, e.g., upon arriving at a point the robot has already travelled through part of the map and has thereby enhanced its knowledge of the map. In all of these cases, gates (or passageway entrances PWE) candidates are supposed to be known from the Cartographer, coded and delivered in the NAVIGATOR's map version.

Two versions of WSA have been developed at the University of Florida: discrete, and quasi-continuous. The process of path generation can be informally defined as follows.

Procedure DISCRETE WSA

1. If there is a straight line trajectory or any uninterrupted trajectory in the vicinity of the "line of sight" (LS) from the current position to the GOAL, execute it and terminate successfully.

2. Determine probable candidates for a PWE on both sides of the LS and arrange them in the increasing order of preferability. If none exist, terminate with failure.

3. Select the first PWE (maximum preferability). Make this node the next "standpoint" (SP) (intermediate goal of the motion).

4. If this node is on the previous solution path list and there exists another node on the candidate list, take the next candidate node as the next "standpoint".

5. Execute a path in the direction of the next standpoint. Place the current "standpoint" on the proposed node list. Discard the remaining candidate nodes.

6. Go to 1.

In the procedure CONTINUOUS WSA, the procedure DISCRETE WSA is used repetitively every Δt units of time along the execution of the path to the next SP.

Another version of a wandering standpoint algorithm, called FEASIBLE-PATH, was developed at the Hughes Research Laboratories. Its process of path generation can be defined as follows:

Procedure FEASIBLE-PATH

0. Initialize A = IP, B = GOAL, PATH-NODE-LIST = NIL, PRED(IP) = NIL (PRED(X) denotes the "predecessor" of X).

1. If the segment $\overline{AB}$ is feasible (does not intersect any obstacles; i.e., B is visible from A) then execute a path directly to B.

2. Let CANDIDATE-LIST = {nodes N: $\overline{AN}$ is feasible, N is a local point of tangency with respect to A, and N / PATH-NODE-LIST}.

   Nodes in the CANDIDATE-LIST are probably candidates for a PWE.

3. If CANDIDATE-LIST=NIL, then let NEXT-NODE = PRED(A). Otherwise, let NEXT-NODE = the most preferable node in the CANDIDATE-LIST (as selected by STRATEGY) and set PRED(NEXT-NODE) = A.

4. If NEXT-NODE = NIL, terminate with failure (no feasible path exists to GOAL). Otherwise, push A onto PATH-NODE-LIST, execute the path from A to NEXT-NODE, set A = NEXT-NODE, and go to Step 1.

Here we say that N is a "local point of tangency" of an obstacle with respect to the point A if N is a node of the obstacle such that locally (in the vicinity of N) the obstacle lies entirely on one side or the other of the line passing through A and N (see Figure 3). By considering these local points of tangency and providing backtracking (via "predecessors" of path nodes), FEASIBLE-PATH is able to solve the most general case in which obstacles are nonconvex.

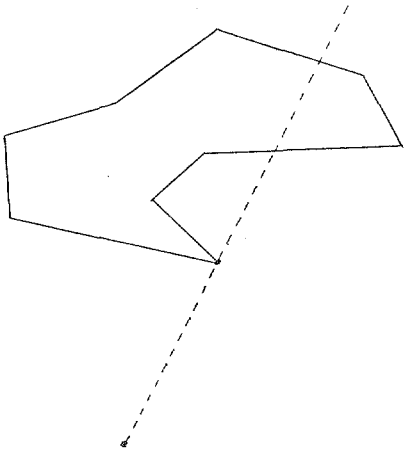


Figure 3. Local Tangent

The preferability of the possible PWE candidates can be determined only if the map is known. In the general case, the map is not known and a heuristic evaluation function should be selected.

WSA is not, in the strictest sense, an A* algorithm even when it employs an admissible heuristic strategy. The nature of the Navigator working with a partially unknown map makes this search computationally nonrecursive. Indeed, the suggested motion can be verified only via the actual motion of the robot and the actual "backtracking" reduces the realized productivity of the mobile device. WSA is also not admissible, since the principle of "biased selection" is recommended for all WSA strategies - PWE candidates are selected from the distinguished segment surrounding the direction to the goal.

WSA is, however, somewhat related to A* algorithm. Multiple user-selected heuristic strategies include under- and over-estimators. However, lists of OPEN and CLOSED nodes are not maintained. Only nodes of the search which lie on the solution path are retained. Single-level backtracking under

strict control is implemented by a special procedure. The result of combining the heuristically-guided search of A* and the limited information available to a sensor is a biased best-first search algorithm which can be operated in real time for mobile systems in which tactical planning is desired.

Search efficiency is increased and path error is decreased with the addition of nonobstacle related nodes under certain situations. A slight variation of node position often leads to a more advantageous standpoint from which new nodes can be selected.

Structure of WSA Realization

WSA was developed in an attempt to create a guidance algorithm which was to be capable of working in real time without a large computer. The program was to receive input continuously from a range-findng sensor and was to avoid the complications of pattern recognition and exhaustive searches. The concept of generating the search graph during the motion also allows new map information to be introduced easily.

The WSA search algorithm has two basic parts: candidate node generation based on sensor information and candidate node selection based on some criterion or heuristic strategy. In addition to these basic parts, the WSA program provides an interactive user interface with graphics, hard copy, and keyboard control which are useful at the stage of simulation.

The main program controls WSA initializations and provides interactive I/O. The main WSA operators are as follows.

INTERSECT
Intersection coordinates calculation of two line segments using point slope formula. Control of pointer for entering object list.

EDGES
Maximum deviation from goal angle calculation using the obstacle list.

DUPCK
Tests for two oscillation conditions.
Control of backtracking and alternate TP selection.

STRATEGY
Execution of strategies selecting next TP.

SWAP
Setting up next TP from proposed TPs under control of STRATEGY.

MOVE
Control of current SP location.

STOREEXIT
Control of post-pathfinding routines (tying up loose ends).

EVALUATE
Cost functions evaluation for the path found.

OPTIMIZE
Redundant lines removal from the path and possible re-traversal.

Operation

Most guidance algorithms are required to have a search graph[5,18,24]. In some of the existing works, the search graph is constructed in various ways directly from a given map[5]. Other systems use video and range-finding sensors to create a complete internal map which is then searched for some "optimum" path[10,20,25]. Both of these techniques have advantages and disadvantages over the on-line generation of nodes used by the WSA.

In the simulation obstacle boundaries are stored as lists of vertices of convex polygons. The first step of the WSA is to expand the boundaries of each object in the map. The size of the expansion is governed by the physical dimensions of the robot.

The next step of the WSA tests for intersections of the mobile system with the set of expanded obstacles. A line of sight ("test strip") is constructed between the present standpoint and the GOAL. All intersections of this line and the obstacle sides are reported but only the nearest intersection is retained. A pointer to the object associated with this nearest intersection is passed to the EDGES procedures for the second part of the candidate node generation.

Next, each vertex of the obstacle is used to generate an angle of deviation from the test strip. The vertex with the maximum (positive) deviation angle is called E1, and the vertex with the minimum (i.e., most negative) deviation angle is called E2. The corresponding deviation angles and distances to these two vertices are saved in the variables E1.dev, E2.dev, E1.dis, and E2.dis. These quantities are used in the STRATEGY procedure to select the next SP. It should be noted that this algorithm restricts the polygons to being convex (however, FEASIBLE-PATH has no such restriction). A discussion of the various strategies realized in the STRATEGY procedure may be found in the section on heuristic selection. The selection of a strategy equation is done at the time of initial simulation setup.

The STRATEGY procedure sets up the selected node point as a next SP. The variables E.dis and E.dev are used to calculate the known distance and

to estimate the unknown distances (see section on heuristic section). When this next SP is determined, the WSA procedure treates it as a subgoal. The process of testing for intersections is repeated for this subgoal and (usually) the location of the mobile system is moved to the subgoal. The next SP now becomes the new SP and the goal is reinstated as the target for the search procedure. The recursive nature of the WSA procedure guarantees that the subgoal is reachable from the SP before executing the "move".

The INTERSECT and EDGES procedures are the most active in the WSA algorithm. The STRATEGY procedure is the decision-making procedure. The DUPCK (duplication check) procedure guards the system from becoming "stuck" in a loop of decisions. In certain cases, the selection of the next SP will cause the mobile system to move back toward the last SP. In special (but not extremely rare) cases node P will generate node Q as the chosen SP and then node Q will generate node P as its next chosen SP. This "oscillation" condition is prevented by checking the selected node against those already on the solution path list. If the node is on the list, the DUPCK procedure forces the selection of the alternate node.

Use of the WSA Program

Execution of the WSA procedure is guided by the interactive main program. Initially, a simulating map is selected or a new map is defined. New maps are defined by entering coordinate pairs for object vertices in clockwise order terminated by a semicolon. When the map is displayed on the screen, the user is queried as to whether or not any objects are to be moved. Relocation of objects (if selected) is accomplished by adding an offset to each vertex. An option to relocate the initial position (IP) and/or goal is also available.

Strategies Available

When the final configuration is established, the user is prompted for selection of the decision strategy. The strategies available include Projected Path Length, Minimum Deviation Angle and variants of the Projected Path Length. Projected Path Length is the only admissible heuristic.

The six available strategies are expressed in terms of edge distances (E1.dis and E2.dis), edge deviation angles (E1.dev and E2.dev), and the estimates d1 and d2 of the unknown length from edge to GOAL (Figure 4).

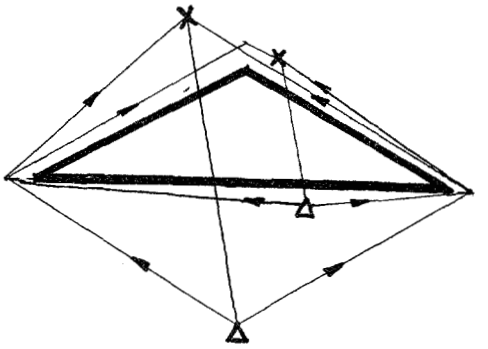


Figure 4. Paths generated by different strategies

The strategies simply select the subgoal to be the edge Ei (i.e. such that {1,2}) of the obstacle which minimizes

1. Ei.dis + di	Projected Path Length
2. Ei.dev	Minimum Deviation Angle
3. $(di + Ei.dis)^2/Ei.dis$	Modified Projected Path Length
4. $Ei.dis + di^3$	Cube Weighted Unknown
5. $Ei.dis + (di/(Ei.dis + di))^2$	Square Rato of Unknown/ Projected Length
6. $Ei.dis + (di/(Ei.dis + di))^2 \cdot di$	Modified Square Ratio

Strategies 3 through 6 create a bias toward the selection of a combined path which has a larger known part. This puts more emphasis on the "seen" part of the path. Strategy 4 heavily weighs the unknown part by cubing it. Strategies 5 and 7 apply the concept of "percent known path" by using the ratio of unknown part to the total path length as a measure of certainty. Clearly, many other strategies can be derived and applied using the STRATEGY procedure.

Output

The graphics screen continuously displays the state of the search. A dashed line indicates the path selected by the search algorithm. At completion, a summary for the distance, time, and energy losses is displayed on the screen and printed at the terminal. The time and energy calculations are based upon the maximum speed and acceleration of the system coupled with a trapezoidal speed curve over the straight line segments. The equation for calculation of the energy consumed depends on two additional parameters of the IMAS - maximum

velocity V_{max} and acceleration A. Energy is calculated for each path segment of length D as

$$ENERGY := (V_{max} \times D/2) - V_{max} \times SQR(V_{max})/(6 \times A)$$

for path segments where distance D allows the IMAS to achieve maximum velocity, and

$$ENERGY := (SQRT(A \times D \times SQR(D)/3)$$

for shorter path segments.

These formulas assume a high velocity versus time function which is trapezoidal. The acceleration is assumed constant during the beginning and end of the motion. This is an oversimplification; however, it can serve as an approximation for performance evaluation. Indeed, any change in direction leads to some speed reduction (not to zero speed, of course). Thus, the results give some rough idea on the energy consumption. The real trajectories of velocity and azimuth are under consideration at the present time.

Testing the Algorithm in Simulation

The strategy for testing the simulation was for two purposes: to investigate and test the algorithm itself and to evaluate the performance of the heuristic strategies developed. In initial runs of the simulation, the special conditions which would cause the algorithm to select the "wrong" way were identified. Other problems were also identified such as the "oscillation effect".

Seven maps were prepared with an assortment of objects in each map. For each map, some initial point (IP) locations and goal locations were chosen. The selection of IPs and goals were chosen to exercise the various distinctions of each strategy.

The three strategies selected for the experiment were all based on the quantities discussed above. The selected strategies were

I. Projected Path Length.

II. Minimum Deviation Angle.

III. Modified Projected Path Length.

These heuristics were applied for a variety of maps. Figures 5-7 show several examples of simulated motion.

The results of simulation can be summarized as follows. No heuristic function is found to work universally well on all varieties of maps. Since the map is unknown, even the admissible heuristic (I) does not provide the optimum path in all cases. In many cases, the nonadmissible heuristics (II and III) give better results. It is found statistically that different strategies can provide minimum path, minimum time, and/or minimum energy consumed. Strategy selection can be possibly determined by the topology of the terrain. (This statement is under investigation at the present time.)

Three different strategies were statistically compared in a variety of obstacle configurations and locations (Table 1). An interesting result should be mentioned: strategy with minimum time of operation does not coincide with the strategy which minimizes path length and average power of the motion.

Table 1 - Comparison of Simulated Navigator Performance (averaged)

Strategy	Total Path Length	Relative Value	Total Time	Av. Speed m/sec.	Av. Energy per Time (Power j/sec)	Energy per Unit of Path (j/M)
I (admissible)	4561.8	1.03	3281.4	1.39	101.3	72.9
II (nonadmiss.)	5007.0	1.13	3570.2	1.40	102.6	73.1
III (nonadmiss.)	4430.3	1	3334.4	1.33	101.1	76.1

Paths for these heuristics are exemplified in Appendix I, (tested on PDP-11 at the University of Florida). Dealing with a trap is shown in Figure 5 (tested on LM-2 Symbolics at Hughes, Inc.)

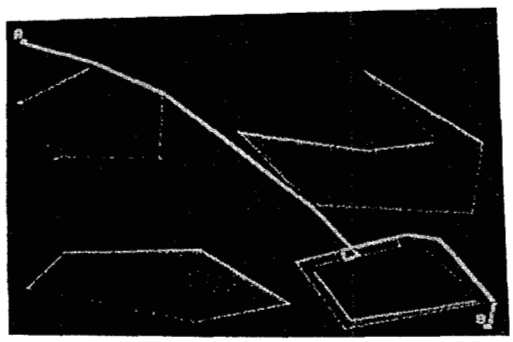


Figure 5. Dealing with a Trap.

Modifications to the WSA Procedure

There are minor differences in the techniques used in the simulation program and those which would be applicable in the real world. A set of modifications to bring the simulation to more realism is presently being implemented.

One of the modifications is in the simulation of the detection of obstacles. The initial version of WSA assumes knowledge of the entire obstacle and uses this knowledge to generate alternatives. In real world situations, only the portion of each obstacle which is actually visible may constitute all the knowledge about a particular obstacle. This is the approach taken in the current version of WSA. This enables WSA to better deal with maps which are entirely or partially unknown.

In order to deal with only the visible portions of obstacles, a sensor simulation has been added to the curent version of WSA. The "sensor" makes available to WSA only those segments of each obstacle which are visible from the IMAS's current position. In addition, the sensor works with a finite range and scan angle which can be modified in order to evaluate the effect of sensor parameters on navigation performance.

Because of the addition of a sensor and the consequent change in the type of data that WSA deals with, a change has been made in the way WSA generates the alternatives of motion. Instead of generating alternatives at the points of tangency of obstacles, the alternatives are generated in what is termed "passageways". A passageway represents gaps in which the IMAS can pass through. They can be found by looking for discontinuities in the range scan data. This approach brings the navigation problem closer to reality and enables WSA to deal with any type of obstacle, whether it be convex or concave.

A second addition to WSA is a modification allowing a quasi-continous search during the motion. This idea is considered valuable in view of the possibility that a better path may be discovered as new information is received in the course of the motion from the first SP to he planned next SP. This is accomplished by stepping a small Δd distance in the direction of the next SP and then obtaining new sensor data and reevaluating the situation ("rethinking").

These current modifications to WSA bring the navigation problem closer to reality and enable us to study the effect of sensor parameters and time of rethinking on the performance of WSA. Figure 6 shows an example of the motion trajectory generated with these current modifications (bold lines show the visible parts of obstacles, the mark "x" denotes the passageways alternatives).

The capability to deal with moving obstacles is also being developed. Two ideas concerned with the technique of this problem solution are under consideration. One simple strategy is to predict when a moving obstacle is to move out of the way. More powerful is the continuous forecasting and reevaluation of the situation for path intersection and collision forecasting and reevaluation of the situation for path intersection and collision using the WSA technique. This technique of collision avoidance requires a close interaction between the PILOT and the NAVIGATOR. Similar approach can be applied for dealing with aggressive obstacles.

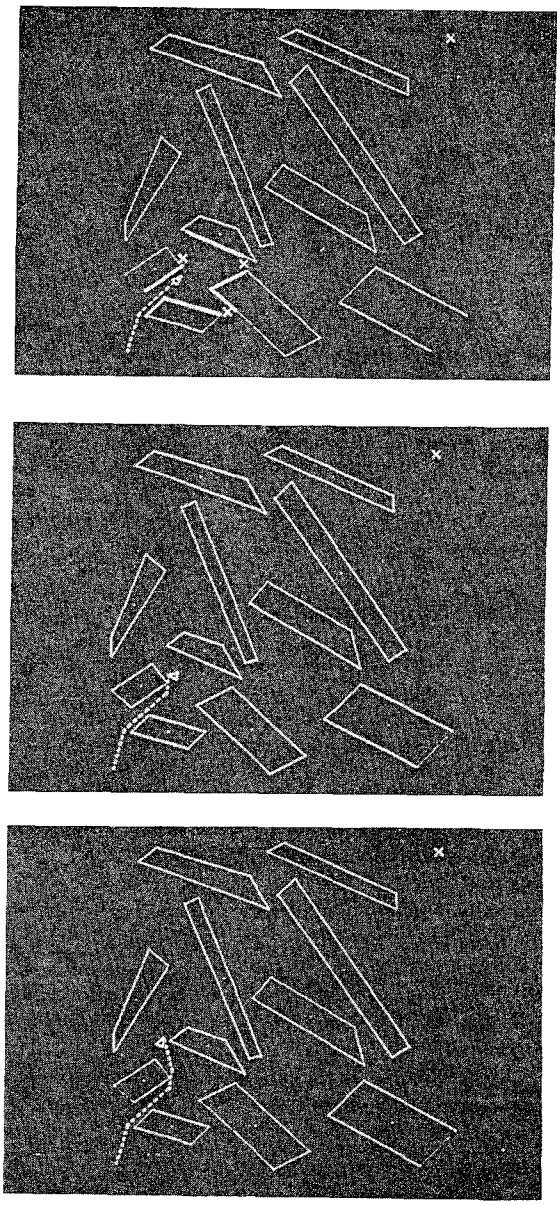


Figure 6. Motion Trajectory Generation.

Conclusions

The work described in this paper provides a good framework for continued research in the area of tactical navigation. It addresses the problems of search graph construction and heuristic search procedures.

It was discovered that no single heuristic may exist that will give optimum results in all circumstances, hence, we are led to consider the use of multiple heuristics to be matched to particular

situations. This is under investigation at the present time.

An implementation of the Navigator has been successfully incorporated into a global path planner. In addition, the algorithm has been implemented using simulated laser range scan data; the problem of constructing a search graph from real sensor data is currently being addressed.

At the present time, new and encouraging preliminary results have been obtained. This approach promises to make this system of tactical Navigation more efficient and intelligent.

References

[1] A. Meystel, "Planning in the anthropomorphical machine intelligence," Proc. of Intl. Conf. on Cybernetics and Society, Seattle, WA, pp. 648-652, 1982.

[2] A. Meystel, "Intelligent control of a multi-actuator system," Proc. of 4th IFAC/IFIP Symp. on Information Control Problems in Manufacturing Technology, Washington, D.C., pp. 126-136. 1982.

[3] T.L. Johnson, S.D. Milligan, "Emulation/simulation of hierarchical control systems," Proc. IFAC Symp. on CAD of Multivariable Technological Systems, Purdue University, Sept. 1982.

[4] N.J. Nilsson, "Mobile automation: an application of AI techniques," Proc. of the IJCAI, Washington, D.C., pp. 509-520, 1969.

[5] P. Hart, N.J. Nilsson, B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," IEEE Trans. on Systems and Cybernetics, SCC-4, No. 2, pp. 100-107, 1968.

[6] R.A. Lewis, A.K. Bejczy, "Planning considerations for a roving robot with arm," Proc. 3rd IJCAI, pp. 308-315, 1973.

[7] J.A. Miller, "Autonomous guidance and control of a roving robot," Proc. IJCAI-77, Vol. II, Cambridge, MA, 1977.

[8] A.M. Thompson, "Steps toward practical robot planning," Proc. of JACC-31, Vol. I, Charlottesville, VA, 1981.

[9] J.A. Miller, "Autonomous guidance and control of a roving robot," Proc. IJCAI-77, Cambridge, MA, pp. 759-760, 1977.

[10] A.M. Thompson, "The navigation system of the JPL robot," Proc. of IJCAI-77, pp. 749-757.

[11] G. Giralt, R. Sobek and R. Chatila, "A multi-level planning and navigation system for a mobile robot: a first approach to HILARE," Proc. IJCAI-79, Vol. I, Tokyo.

[12] L. Marce, M. Julliere, H. Place and H. Perrichot, "A semi-autonomous remote controlled mobile robot," The Industrial Robot, December 1980, pp. 232-235.

[13] M. Julliere, L. Marce, H. Place, "A guidance system for a mobile robot," Proc. of the 13th Intl. Symp. on Industrial Robots, V. II, pp. 1358-1369, 1983.

[14] V.O. Belenkov, S.V. Gusev, Y.K. Zotov, V.I. Ruzhanskiy, A.V. Timofeyev, V.M. Fzolov, and V.A. Yakubovich, "Adaptive system for control of autonomous mobile robot," Eng. Cybernetics, No. 6, pp. 37-45, 1978.

[15] V.P. Pyatkin and V.Y. Sirotenko, "Path planning by robot," Electronic Cybernetics, No. 6, pp. 54-59, 1978.

[16] A.S. Narinyani, V.P. Pyatkin, and P.A. Kim, "Walking robot: a non-deterministic model of control," Advance Papers of the 4th IJCAI, Vol. II, Tbilisi, U.S.S.R., pp. 794-797, 1975.

[17] E.I. Kugashev, and Jaroshevskij, "Problems of selecting a gait for an integrated locomotion robot," Papers of the 4th IJCAI, Vol. II, Tbilisi, U.S.S.R., pp. 789-793, 1975.

[18] H.P. Moravec, "Rover visual obstacle avoidance," Proc. IJCAI-81, Vol. II, pp. 785-790, Vancouver, B.C., Canada, 1981.

[19] H.P. Moravec, "Visual mapping by a robot rover," Proc. IJCAI-79, Vol. I, Tokyo, Japan, pp. 598-600, 1979.

[20] H.P. Moravec, "Obstacle avoidance and navigation in the real world by a seeing robot rover," Ph.D. dissertation, Dept. of Computer Science, Stanford University, Stanford, CA, 1980.

[21] H.P. Moravec, "The Stanford cart and the CMU Rover," The Robotics Institute, Carnegie-Mellon University, Pittsburg, PA, January 1983.

[22] T. Lozano-Perez, "Automatic planning of manipulator transfer movements," IEEE T-SMC, Vol. 11, No. 10, pp. 681-698, 1981.

[23] S.M. Udupa, "Collision detection and avoidance in computer controlled manipulators," Proc. of the 5th IJCAI, Vol. II, Cambridge, MA, pp. 737-748, 1977.

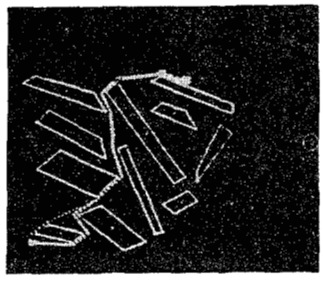
[24] T. Lozano-Perez, M.A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," Communications of the ACM, Vol. 22, No. 10, October 1979.

[25] D.F. Cahn and S.R. Phillips, "ROBNAV: A range-based robot navigation and obstacle avoidance algorithm," IEEE Trans. on Systems, Man, and Cybernetics, Vol. SMC-5, No. 9, pp. 544-551, 1975.

Appendix I

Examples of Path Simulation in an Obstacle Strewn Environment

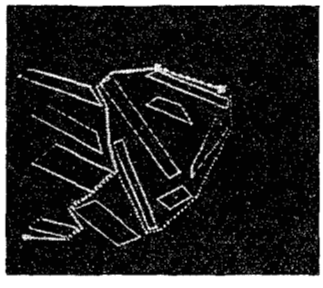
The trajectories of motion are obtained for strategies based upon the projected path length (I), minimum deviation angle (II), and modified projected path length (III).



MAP 1, SITUATION 1

IMS (20,20), GOAL (150,190)

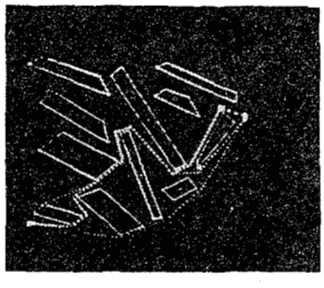
STRATEGY	PATH	TIME	ENERGY
I	(284.2)	(208.2)	(20,611)
II	306.2	224.8	22,195
III	306.2	224.8	22,195



MAP 1, SITUATION 2

IMS (20,20), GOAL (185,175)

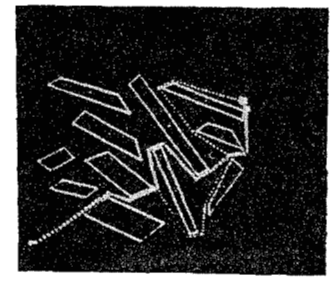
STRATEGY	PATH	TIME	ENERGY
I	(322.3)	(233.6)	(23,466)
II	384.5	277.0	28,064
III	344.3	250.2	25,050



MAP 2, SITUATION 3

IMS (20,20), GOAL (200,140)

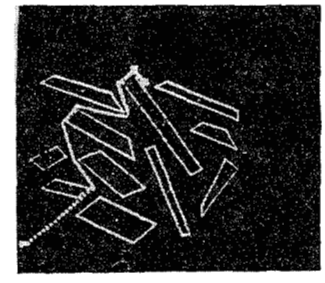
STRATEGY	PATH	TIME	ENERGY
I	313.2	225.7	22,858
II	(252.1)	(175.5)	(18,623)
III	348.9	253.0	25,392



MAP 2, SITUATION 4

IMS (10,10), GOAL (190,160)

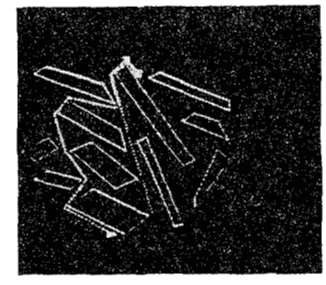
STRATEGY	PATH	TIME	ENERGY
I	(343.4)	(249.6)	(24,981)
II	476.0	334.4	34,521
III	(343.4)	(249.6)	(24,984)



MAP 2, SITUATION 5

IMS (10,10), GOAL (110,180)

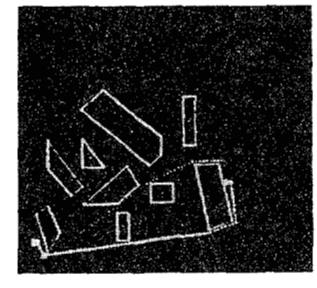
STRATEGY	PATH	TIME	ENERGY
I	295.2	213.7	21,506
II	295.2	213.7	21,506
III	295.2	213.7	21,506



MAP 2, SITUATION 6

IMS (90,10), GOAL (110,180)

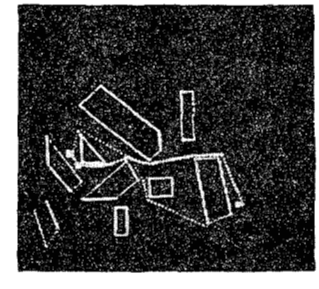
STRATEGY	PATH	TIME	ENERGY
I	(236.5)	(167.0)	(17,382)
II	303.0	220.8	22,022
III	303.0	220.8	22,022



MAP 3, SITUATION 7

IMS (190,80), GOAL (20,20)

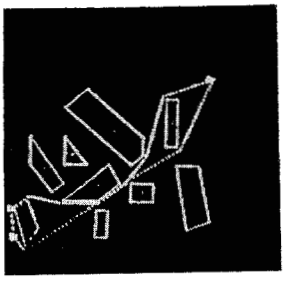
STRATEGY	PATH	TIME	ENERGY
I	228.9	163.9	(16,749)
II	(227.5)	(159.2)	16,779
III	(227.5)	(159.2)	16,779



MAP 3, SITUATION 8

IMS (200,50), GOAL (50,110)

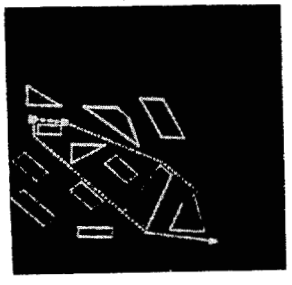
STRATEGY	PATH	TIME	ENERGY
I	229.8	168.2	16,749
II	(200.0)	(146.5)	(14,505)
III	(200.0)	(146.5)	(14,505)



MAP 3, SITUATION 9

IMS (190,190), GOAL (20,20)

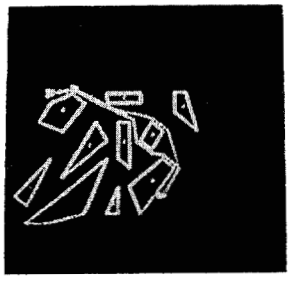
STRATEGY	PATH	TIME	ENERGY
I	276.9	195.9	20,347
II	276.9	195.9	20,347
III	283.1	201.9	20,739



MAP 4, SITUATION 10

IMS (200,10), GOAL (35,135)

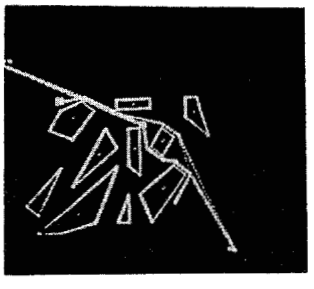
STRATEGY	PATH	TIME	ENERGY
I	222.7	154.1	16,489
II	222.7	154.1	16,489
III	224.2	157.0	16,534



MAP 8, SITUATION 15

IMS (90,110), GOAL (10,120)

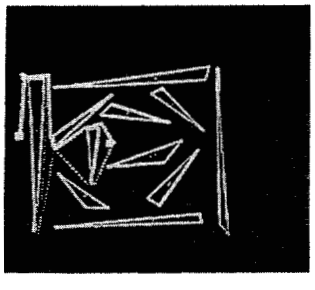
STRATEGY	PATH	TIME	ENERGY
I	231.6	167.5	16,878
II	442.6	308.2	32,702
III	231.6	167.5	16,878



MAP 5, SITUATION 16

IMS (150,50), GOAL (50,150)

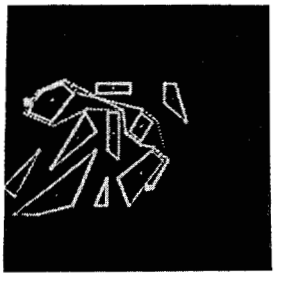
STRATEGY	PATH	TIME	ENERGY
I	191.8	139.1	13,961
II	188.3	138.7	13,631
III	188.3	138.7	13,631



MAP 8, SITUATION 14

IMS (10,120), GOAL (195,105)

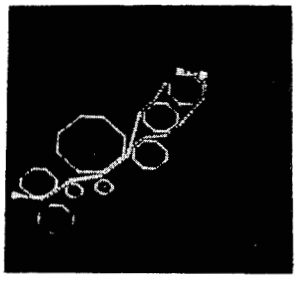
STRATEGY	PATH	TIME	ENERGY
I	402.1	284.9	29,525
II	518.0	354.0	38,499
III	402.1	284.9	29,525



MAP 5, SITUATION 17

IMS (150,50), GOAL (40,140)

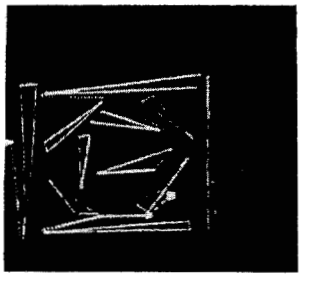
STRATEGY	PATH	TIME	ENERGY
I	212.1	154.5	15,417
II	213.3	159.1	15,363
III	207.3	153.2	14,985



MAP 6, SITUATION 11

IMS (10,60), GOAL (170,190)

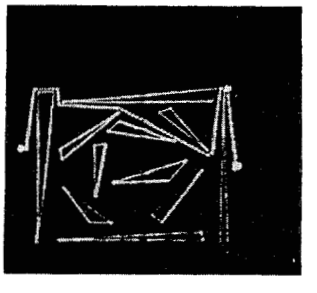
STRATEGY	PATH	TIME	ENERGY
I	232.7	175.7	16,676
II	232.7	175.5	16,676
III	234.5	171.0	17,021



MAP 8, SITUATION 12

IMS (10,120), GOAL (90,110)

STRATEGY	PATH	TIME	ENERGY
I	239.6	167.5	16,878
II	216.9	157.7	15,775
III	239.6	167.5	16,878



MAP 8, SITUATION 13

IMS (10,120), GOAL (150,60)

STRATEGY	PATH	TIME	ENERGY
I	298.8	212.3	21,916
II	251.1	174.9	18,549
III	251.1	174.9	18,549