A Parallel Processor Algorithm for Robot Route Planning.

CM, Nitkowski,
Artificial Intelligence Laboratory,
Department of Computer Science and Statistics,
Queen Mary College,
Mile End Road,
London E1 4NS,
England.

ABSTRACT

This paper presents a fast, uniform, parallel search algorithm for robot route planning and obstacle avoidance. The algorithm is equally applicable to real or synthetic data and overcomes many problems associated with other route finding methods. The time taken to generate a route through a an arbitrarily complex environment has been reduced to an insignificant fraction of the time taken for the robot to traverse the route. Furthermore the time taken to create the route is independent of environment complexity and only linearly proportional to route length. Actual results and timings from running the algorithm on the ICL Distributed Array Processor and executing the resultant path on a mobile robot are presented.

1 Route Planning for Mobile Robots.

One of the most important components of the software for a mobile robot system is a route planner, to navigate the vehicle through a completely, or partially mapped environment. Such programs split into two main sections, the representation of the environment and a method of searching possible route paths between the current robot position and some new location, avoiding known obstacles.

Well established methods of describing the environment include approximating all obstacles as polyhedra, Nilsson [1], or calculating minimum enclosing circles or ellipses, Moravec [2]. In either case the standard approach to the actual planning is to expand the environment representation by a 'centre to edge' distance for the robot. Thereafter the robot may be treated as a moving point, rather than a swept volume. Brooks [3] describes a method in which free space is represented by overlapping 'generalised cones', Matsushiraa and Oda [4] describe a method for robot motion in an unknown environment using a tactile sensor.

II The Mobile Robot.

We have a mobile robot with which various experiments may be performed. It is of a very conventional design. Two stepper motors independently drive wheels equidistantly spaced from the midpoint along the lateral axis. The base, and hence the groundplan, is octagonal measuring 24 inches across the flats. It may therefore be reasonably approximated by a circle when the occasion demands. The vehicle is equipped with various sensors, notably an ultrasonic rangefinder and a standard video camera.

A program residing in the robot's 6502 based microcomputer allows the robot to map its environment using the rangefinder. Figure 1 shows such a map, of one of the offices attached to the laboratory, there is a table in the middle of the room. Eight radial scans are made, each comprising 200 rotations of 1.8 degrees followed by a range reading. The figure represents an area of 200 by 200 inches. Throughout this mapping process and route execution the precision and repeatability of the stepper motors is relied on to maintain navigational accuracy.

It was primarily the poor quality of the sensor data obtained as a map that prompted this investigation. It was considered that it would be less effort and the results would be more reliable if a method could be developed that completely avoided the need for detailed analysis and line fitting to this type of data. In the event, and partly due to the availability of the ICL Distributed Array Processor (DAP) a cellular representation of the problem, coupled to a parallel search that spreads through a lattice of free points was chosen.

Figures 2 and 3 show the results of the algorithm. The total robot environment is represented

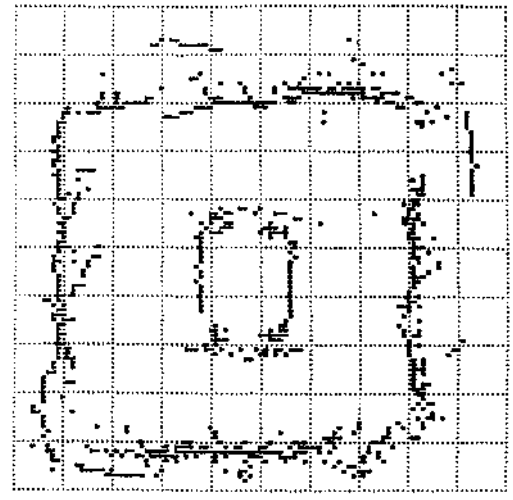

Figure 1 - Rangefinder scan map.

by a 64x64 lattice of points, some of which will be occupied by obstacles that will have to be avoided during robot motions. These points are shown as asterisks "•". Lattice points which become occupied during the expansion process, so that the robot may be treated as a point, are shown as "-"• The current or 'start' position of the robot is shown by a "S" and the desired, 'goal' position by "G". Points indicated by "R" are (possibly many) paths from start to goal with equal path lengths, in terms of horizontal, vertical and diagonal movements. Intermediate level robot software will interpolate from "S" to "G" via those route points labelled "X" during route execution,

III The ICL Distributed Array Processor.

The DAP, Reddaway [5], is a highly parallel architecture machine cast in the Multiple Data Single Instruction (MDSI) mould. It may be best viewed as a matrix of 64 by 64 single bit processors linked to a common control unit, each having I6Kbits of memory with a cycle time of 250nS. This memory is shared by the host 2980 processor, entry to the DAP and all I/O is mediated via this host machine. The DAP may either be programmed in machine code or DAP-Fortran. DAP-Fortran is augmented with a matrix data type (and others) with an implicit 64x64 dimensionality and all elements are processed simultaneously. These may be of type real, integer, character and logical. Various data structures within the program are stored as logical matrices (4096 bits), for instance:

ENVIRONMENT - The map as input to the program.
ROBOT SHAPE - The robot ground plan.
GROWN_WORLD - ENVIRONMENT expanded by ROBOTJSHAPE.
FROM POINT - Single bit set indicating star" node.
TO POINT - Single bit indicating goal node.
ROBOT PATH - Potential path points.
NODE_MASK - Robot turn points.

IV The Algorithm.

The algorithm proceeds in a number of stages, the output from each being a logical mask, each bit then represents some state of its corresponding point in the lattice.

Step 1 - Environment expansion - the ENVIRONMENT mask is shifted (using the DAP's inbuilt planar shift functions) by an amount specified by the various bits set in ROBOTJ5HAPE and the result logically ORed with GROWN_WORLD. The effect of this routine is clearly seen in figure 3.

Step 2 - Endpoint checks - FROM POINT and TO_POINT are logically ANDed with GROWN_WORLD, a truie result indicates that the robot either" is, or is being sent to somewhere it cannot be.

Step 3 - Lattice search - This is a two pass routine. Before each pass an integer matrix (SPREAD) is set up that will indicate the distance of each unoccupied lattice point from the start, it is initialised to some large value.

Step 3.1 - first pass - Beginning with only FROM POINT set, each of its eight directly connected neighbours are tested. The value for each point in SPREAD is updated with a new path distance back to the start point if it is 1) free point, and 2) it is previously unvisited and 3) the new distance is smaller than its current estimate. Each of the

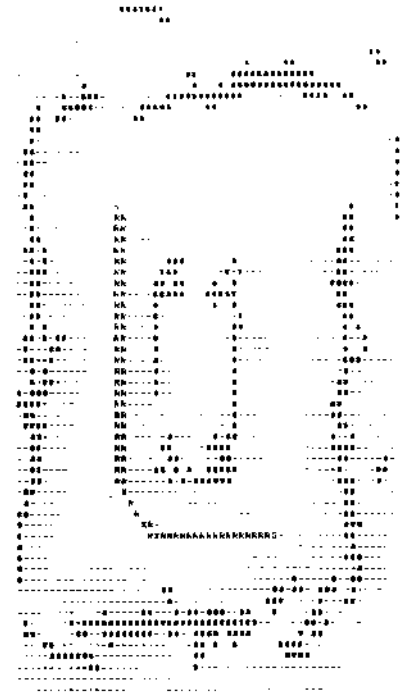

Figure 2 - Path through real data.

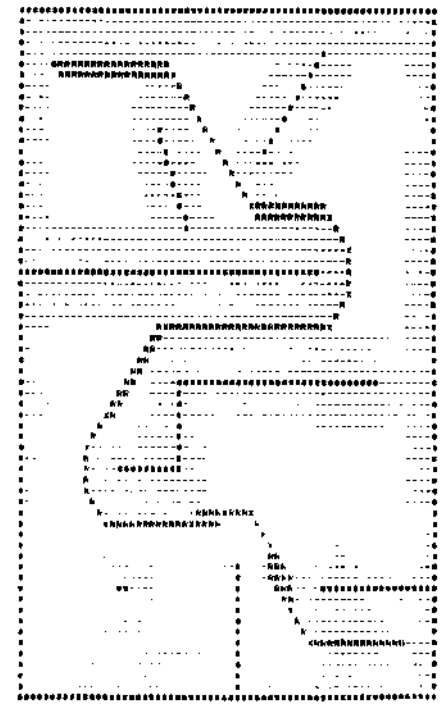

Figure 3 - Path through synthetic data.

eight directions is tried in turn, though all lattice points are evaluated in parallel, diagonals are increased by square root two times the horizontal or vertical. The search then proceeds iteratively from the points newly reached as a spreading wavefront. Termination occurs when there are either no new *free* points (in which case the start was enclosed), or the goal point is reached.

Step 3.2 - second pass - Rather than organise a system of backpointers, in the second pass start is set to T0J0INT and the goal to FROM_POINT and the process is repeated to produce a second SPREAD matrix, whose values indicate shortest path distances to the goal.

Step 3.3 - determining the path - The least cost path, or paths, within the constraints of the lattice representation, is indicated as all those lattice points whose value in the sum of the two SPREAD matrices equals that in the TO_POINT point. These points on the lattice are indicated in figures 2 and 3 by alphabetic characters. Each of these points, saved in ROBOT_PATH, represents a path through the lattice to the goal that be be reached from the start with an equal number horizontal, vertical or diagonal movements.

Step 4 - Generate actual robot path - The algorithm generates a list of lattice points on the route path that most closely corresponds to a route finder which traverses between object corners. To achieve this concavities in the ROBOT_PATH mask are detected, by recognising path points in a 3 by 3 neighbourhood, various patterns of the nine bits indicate these situations. These lattice points are indicated in the NODE_MASK mask. This parallel operation gives no indication as to route order, neither is any indication given of the presence of multiple iso-valued paths.

Step 5 - One further pass is made through the ROBOT_PATH mask from start to goal. The order in which lattice points at which the robot will change direction is found and their coordinates recorded in the robot command buffer. Multiple paths are detected by a blob counting method at each stage in this expansion. A split in the path is indicated by multiple blobs, one is selected arbitrarily, the others tagged as inactive, these tags are propagated along the redundant paths until they merge again with the selected path. Only direction change points from NODE MASK on the active path are transferred to the command buffer. A pretty-map routine is then used to generate the output seen in figures 2 and 3.

V Advantages of this algorithm.

It enables a uniform treatment of synthetic (figure 3) and real (figure 2) data. The real data has undergone only minor pre-processing, two passes of an algorithm that removes 'noise' points, those with less than two neighbours.

The expansion algorithm is independent of environment complexity, rather computational effort is proportional to the number of points set in the robot ground plan.

Search time for the route points is linear with respect to the number of lattice points traversed from start to goal, this is in turn a good measure of the actual distance the robot will have to move.

In general robots with assymmetric ground plans need additional methods, although the algorithm is suitable for assymetric robots which move by pure translation, Lozano-Perez and Wesley [5], this would be trivially achieved by offering a modified ROBOT SHAPE mask as input.

Plans may be generated at different resolutions by a simple rescaling of the environment data. A rescaling has been made between figures 1 and 2.

Very advantageous timings are achieved, which are in part offset by the fact that the DAP may only be used in batch mode.

VI Program timings.

For these 64 by 64 maps the program timings may be split into three parts. First system overheads (conversions between DAP and 2980 Fortran storage modes etc.), about *22* milliseconds/run; second a fixed time component of 20 mS/run, (16 for expansion, one extra for noise reduction on real data). The third component depends on the number of expansions between start and goal, observed timings are 2.035mS/expansion to generate ROBOT_PATH and 0.309mS/expansion for the final pass along ROBOT PATH.

The synthetic data path (figure 3), with 146 expansions took 0.387 seconds of DAP processor time, the robot makes 18 changes of direction and traversed the 481 inch path in 188 seconds. The real data (figure 2), with 50 expansions required 0.158 seconds of DAP processor time, the robot makes five changes of direction and traversed the 151 inch path in 62 seconds. In both cases the lattice is on a three inch grid spacing.

REFERENCES.

[1] Nilsson N.J. (1969) "A mobile automaton: An application of artificial intelligence techniques". IJCAI-1969, pp509-520.

[2] Moravec H.P. (1981) "Robot Rover Visual Navigation". UMI Research Press, Ann Arbor, Michigan.

[3] Brooks R.A. (1982) "Solving the find-path problem by good representation of free space". AAAI 1982, pp38I-386.

[4] Matsushima K. and Oda M. (1982) "Path finding algorithm for a mobile robot with tactile sensor". Proc. 2nd. Intl. Computer Engineering Conf., 1982, ASME. pp49-52.

[5] Reddaway S.F. (1973) "DAP - A distributed array processor". Proc. 1st. Annual Symposium on Computer Architecture, pp. 61-65

[6] Lozano-Perez T. and Wesley M.A. (1979) "An algorithm for planning collision-free paths among polyhedral obstacles". Communications of the ACM, 22-10, pp560-570.