Robotica (1998) volume 16, pp. 659–667. Printed in the United Kingdom © 1998 Cambridge University Press

Sonar Based Systematic Exploration Method for an Autonomous Mobile Robot Operating in an Unknown Environment

Jong Hwan Lim* and Dong Woo Cho†

(Received in Final Form: February 17, 1998)

SUMMARY

A new systematic exploration method is addressed that permits a mobile robot to effectively acquire the information on an unkown environment without wasting time. The algorithm is composed of following three modules: the first is the decomposition of a workspace into several sub-nodes by employing the concept of Quadtree. These nodes are chosen as sub-goals to be reached successively. The second is that controls the robot to follow the boundaries of objects in the environment in order to reach each sub-goal avoiding the situation of the robot entering a local minima. Here we utilize the imaginary distance forces exerted by objects. The third is the node conditions for estimating the map quality for each node, which enables the robot to remove sufficiently informed nodes from the Quadtree. Two conditions are defined for the estimation of the map quality; entropy and maximum level of Quadtree. The proposed approaches were successfully implemented to our mobile robot equipped with sonar sensors for constructing a sonar map of an unknown environment in a real world.

KEYWORDS: Mobile robot; Unknown environment; Sonar sensors; Quadtree

1. INTRODUCTION

An autonomous mobile robot has to acquire information on the workspace whether it is completely known one or not. Particularly, it is inevitable for the robot operating in an unknown environment to effectively investigate its entire work space in order to collect the information necessary for map construction or for performing given tasks, which is termed exploration. Problems involved in a systematic exploration of a robot's workspace are how to select a series of sub-goals to visit that eventually makes the robot acquire information on the work space, and how to plan a path and control the robot to reach those sub-goals given no prior knowledge.

Although there have been numerous researches on the autonomous mobile robot, few have been done on plannig a systematic exploration of a robot's workspace. Taylor and Kriegman[1] suggested a method to visit all the landmarks placed in the work space. That is, the robot maintains a list of all landmarks it has seen, and repeatedly selects landmark as a next goal. This, however, cannot be applied to an unknown and unstructured environment because it uses landmarks placed in the workspace in advance.

On the other hand, one can find a variety of methods for planning a path and its tracking[2–7]. Most of them are dealing with the path planning with complete information known as "piano mover's problem" in which perfect knowledge about the obstascles and robot's position are assumed. As far as exploration of unkonwn environment is concerned, two kinds of problems arise to apply these methods. First, the robot does not have any knowledge about the locations and shapes of objects; it only knows those that it has seen. This makes it impossible to genrate a path in advance. Second, it is not clear whether the goal can be reached or not. If the goal is not reachable, then the robot would never stop trying to reach the goal.

Path planning with incomplete information is the one when the information available to the robot is limited to its own current coordinates and those of the target position. This kind of path planning has been addressed in a couple references; Borenstein and Koren[8] and the author's previous work[9]. The methods are, however, only a collision avoidance procedure, and cannot cope with trap situation due to local minima. Lumelsky[10,11] presented an algorithm, based on the assumptions that the robot is a point and can follow object boundaries with the aid of tactile sensors, that can solve the trap situation in path planning with incomplete information. Another feature of their method is that it informs the robot whehter the current goal is in a reachable region or not. From an application standpoint in a real world, however, some difficulties arise directly from the assumptions; a robot has a finite dimension with various shapes, and it is never easy for the robot to follow an object boundary using only sonar data.

The control methods to follow an object boundary using sonar sensors have studied by some researchers. Gat[12] turned the specular reflection property of sonar waves to use by following a wall with two sonar sensors. If one of them fails to detect the wall due to the specularity of sonar beam the robot rotates in such a direction where both sensors can detect the wall. Turennout[13] presented a method that uses the range difference during a given time interval with only one sonar sensor. This difference was then utilized to estimate the orientation of a robot relative to a wall, and the robot was controlled to follow the wall using the estimated orientation. Skewis and Lumelsky[14] used the smallest range reading from the object boundary to give an angular velocity commanded to the robot. These are very simple and fast, but the dependency on only one or two raw sonar data may

* Department of Mechnical Engineering, Cheju National University, Ara-dong, Cheju 690–756 (Korea)
† Department of Mechnical Engineering, Pohang University of Science and Technology, San 31 Hyoja-dong, Pohang 790–784 (Korea)

https://doi.org/10.1017/S0263574798000587 Published online by Cambridge University Press

result in a poor performance or collision with an arbitrarily shaped object because of the uncerainty and multipath effect of sonar sensors.

Our paper addresses a systematic exploration rule, based on the concept of quadtree, for the robot operating in an unknown environment. We also suggest an efficient control method for a robot to follow an object boundary in order to execute the planned path during exploration procedure. The object following control is based on the local map which represents a limited area around the robot. The method will be robust in the presence of noise because it does not use the raw data themselves that may be erroneous due to the uncertainty and multipath effect, but it uses the accumulated information in a map.

2. CONSTRUCTION OF A SONAR MAP (CERTAINTY GRID REPRESENTATION)

A certainty grid (or occupancy grid) consists of 2 dimensional divisions of space into cells. The cell contains a probability of being occupied, $p(o)$, and that of being empty of the coresponding region of space (see Fig. 9 as an example). The occupancy probability of each grid is estimated from the well-known Bayesian updating model[15–17] utilizing the Bayesian likelihood ratio.

The authors' recent research[18] has shown that the Multipath Bayesian Model (*MBM* model) is efficient for the construction of a high quality sonar map of an environment. This is a modified version of the earlier Bayesian model by introducing the concept of cell orientation in order to reduce the multipath effect inherent in sonar sensing. During the exploration procedure, the robot continuously acquires sonar range data and updates the sonar map of the workspace using this *MBM* model.

3 EXPLORATION

3.1 Quadtree Representation of a Workspace

A quadtree[19] is a recursive decomposition of a two-dimensional space into uniform 2 x 2 blocks, as shown in Fig. 1. In the figure, the root (level 0) of the tree is composed of 4 nodes of level 1, and each node has again 4 nodes of level 2, respectively. A maximum level of subdivision is genrally fixed beforehand. In image processing, each node represents a uniformly colored (or gray level) region. If this is not true for a certain node it is further divided into 4 nodes till each node has uniform gray level.

In this study, however, each node repreesnts a normalized entropy of the region estimated from the following equation:

$$E_{nor} = -\frac{1}{N}\sum_{1=1}^{N}[p(o_i)\log_2 p(\bar{o}_i) + p(\bar{o}_i)]\log_2 p(\bar{o}_i] \qquad (1)$$

where N and $P(o)$ denote the number of cells in a node and occupancy probability of a cell in it respectively. E_{nor} has a maximum value of 1, perfectly unknown region, for an initial map where the probabilities of all cells are 0.5, and drops towards 0 (completely known region), as the probabilties go to either 0 (empty cell) or 1(occupied cell). One, therefore, can say E_{nor} represents the uncertainty of the region or the amount of information stored in that region.

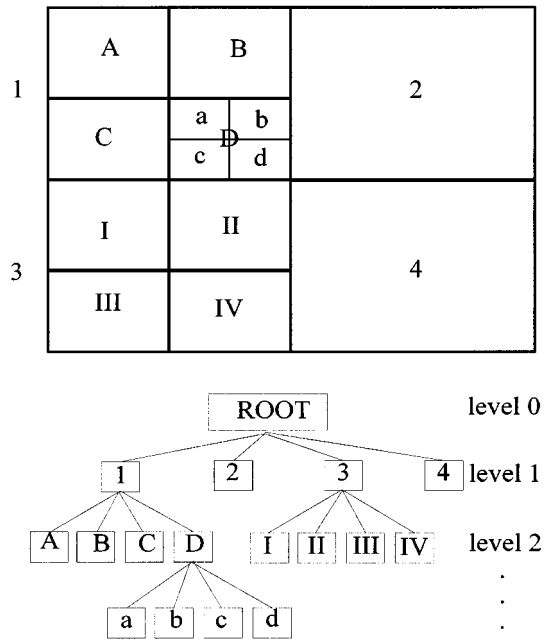


Fig. 1. Quadtree representation of a workspace

3.2 Exploration Rules

A conventional Quadtree aims at the division of the region into free and obstacle regions given sufficient information about the whole space. Our aim is, however, to acquire sufficient information on the work space efficiently together with the aid of relevant node conditions. That is, starting from the root node (the whole workspace), 4 nodes (1, 2, 3 and 4 in Fig. 1) of level 1 are defined. And each center of the nodes is chosen as a sub-goal to be visited. After visitng all sub-goals, i.e. the centers of 1, 2, 3 and 4, the robot estimates the node conditions defined below of each node of level 1. The node that satisfy the conditions are removed from the tree, while the rest are further subdivided into 4 nodes of level 2, e.g. nodes A, B, C, and D in Fig. 1. Again, the centers of the nodes A, B, C and D are chosen as sub-goals, and the robot visits them. If all nodes (A, B, C, and D) satisfy the node conditions then they are removed from the tree structure, which results in the removal of node 1 of level 1. On the other hand, if some of the nodes do not satisfy the conditions they are again subdivided into 4 nodes (e.g. nodes a, b, c and d) while the others that satisfy the conditions are removed from the tree (e.g. nodes A, B, and C). This procedure is continued until there remains no node defined in the tree structure, except the root node.

Node conditions for termination of exploration in one node should be appropriately chosen to acquire sufficient information of a robot's surroundings without wasting time. A simlpe method is that one establishes a threshold entropy value (E_T), and then the nodes below E_T are removed from the tree assuming it is sufficiently informed one. The method, however, would fail to work because there can be a node of which E_{nor} never drops below E_T, which, in fact, can frequently occur. Consider a node filled with an object. E_{nor}

https://doi.org/10.1017/S0263574798000587 Published online by Cambridge University Press

for such a node will remain unchanged ($E_{nor} > E_T$) even though it is sufficiently informed because the probabilities of the cells inside an object can rarely be changed, and if so, the amount of change is very small. This fact leads us to lay down following node conditions that exploration in a node terminates:

Node Conditions: terminate exploration in the node and remove it from the tree structure in one of the following cases:

(i). $E_{nor} < E_T$

(ii) N_s (minimum node size of the tree structure) is reached.

where E_{nor} is the normalized entropy of the node, and N_S is a node size such that the node can be sufficiently informed by visting the centre only once. The values of E_T and N_S will be determiend through a set of experiments.

3.3 Local Path Planning for Exploration

To execute the exploration rules stated above, the robot should have an appropriate means to generate a path. The robot, however, can only have a limited amount of information about its surroundings, and it does not even know whether there exists a solution to reach the goal or not. Lumelsky[11] proposed various methods for this kind of path planning. We use one of them, *Bug2* algorithm, which is more efficient and simpler than others.

The method assumes that the robot is a point, so that an opening of any size between two distinct obstacles is concerned to be passable, and that the robot knows if it is in contact with an obstacle. Let H_j and L_j be the *j-th* occurrence of hit point at which the robot contacts the obstacle, and leave point at which the robot leaves the obstacle respectively. With the help of Fig. 2, the algorithm is represented as (initialy $j=1$):

Step 1. From the point L_{j-1}, move along the straight line (start, target) until one of the following occurs:

(a) The target is reached. The procedure stops.

(b) An obstacle is encountered and a hit point H_j is defined. Go to step 2.

Step 2. Using the accepted direction (left or right), follow the obstacle boundary until one of the folloiwng occurs:

(a) The target is reached. The procedure stops.

(b) The line (start, target) is met at a point Q such that the distance from Q to the target is smaller than that from H_j to the target, and the line (Q, target) does not cross the current obstatcle at the point Q. Define the leave point $L_j = Q$. Set $j = j+1$. Go to step 1.

(c) The robot returns to H_j and thus completes a closed curve (the obstacle boundary) without having defined the next hit point, H_{j+1}. The target is trapped and cannot be reached. The procedure stops.

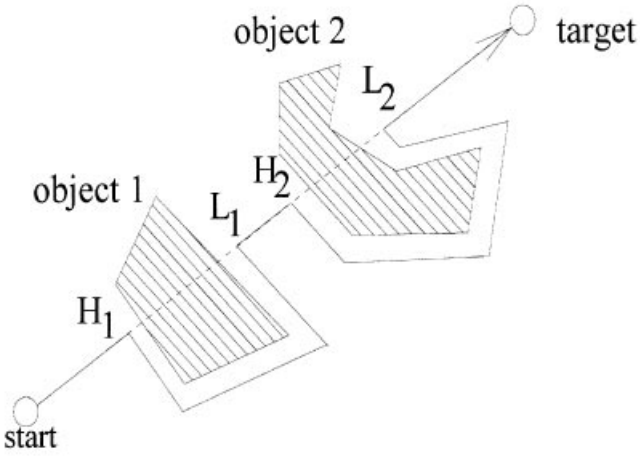
By following the above step, the robot can reach the target point without any *a priori* knowledge avoiding the situation of entering local minima. In a real world, however, the robot has a finite dimension, and it is never easy for the robot to know if it is contact with an obstacle with only uncertain sensor information.

3.4 Boundary Following Control for Path Tracking

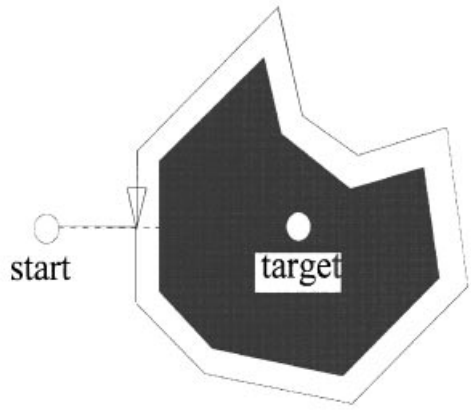
3.4.1 Command Window. Lumelsky originally assumed that tactile sensors mounted on the robot can detect the appearance of an object boundary. We, however, only consider sonar sensors that have much uncertainty in detecting the object boundary compared with touch sensors. To reduce the uncertainty of raw sonar data and hence to use in a real-time control, a local map around the robot is constructed. This local map, which termed command window, provides a steering command to the robot based on the relative configuration of the robot to the object boundary appeared in it.

Fig. 3 shows the dimension of our robot. 9 ultrasonic sensors are mounted on it. The dimension of the command window shown in Fig. 4 is 16×18 cells. Each cell in the window represents a real world square of size 0.05×0.05 m^2. All cells are rotated as much as the robot's current angle with respect to global reference of frame so that the orientation of the robot with respect to the window is identical.

The command window should be able to provide the robot with the information on the boundary of object. If enough range data are available, the map from the *MBM* model can do this well. During the exploration procedure,



(a) Bug2 algorithm



(b) The target cannot be reached.

Fig. 2. Path planning algorithm

https://doi.org/10.1017/S0263574798000587 Published online by Cambridge University Press

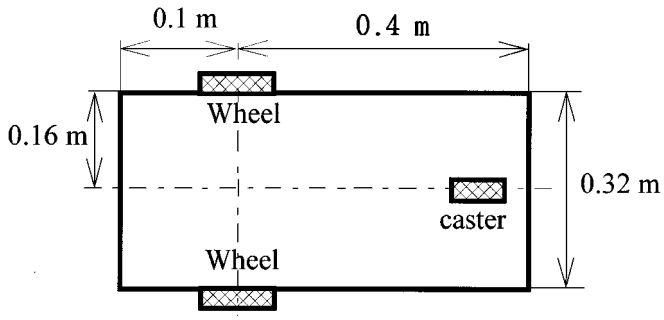


Fig. 3. Configuration of the mobile robot

however, this is not true. Due to the lack of range data, the difference between the occupied cell (possibly an object boundary) and empty one (free space) is so small that the robot can hardly distinguish them. To overcome such a problem, we slightly modified the *MBM* model to update the command window. That is, cells in an empty region where sonar beam pass through, are not to be updated provided they have never been updated before because the occupied cells only can be a boundary of an object. On the other hand, cells in empty region that have already been updated from the previous data are still updated because they could possibly be a phantom object due to false readings. The method will somewhat lower the quality of the resulting map than the original one, but it can provide an effective means for the robot to distinguish the boundary of objects from the free space.

3.4.2 Imaginary Distance Force. The command window is divided into 4 sub-regions; Back (B_r), Left Side (L_r), Front (F_r) and Right Side (R_r) regions. Each occupied cell in the window exerts an imaginary repulsive or attractive force according to its probability and distance between the cell and the boundary of the robot. Fig. 5 illustrates these forces. The distance force f_i exerted by the cells in the *i-th* column belonging to the region of B_r, L_r and R_r is defined as:

$$f_i = \begin{cases} (d_i - d_c)^2 p(o), \; if \; d_i < d_c \\ 0, \; if \; d_i > d_c \; and \; i \in R_r \\ -(d_i - d_c)^2 p(o), \; otherwise \end{cases} \quad (2)$$

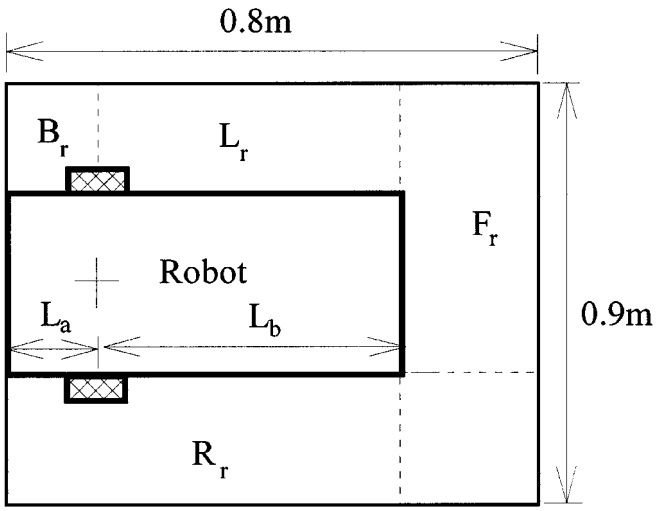


Fig. 4. The command window

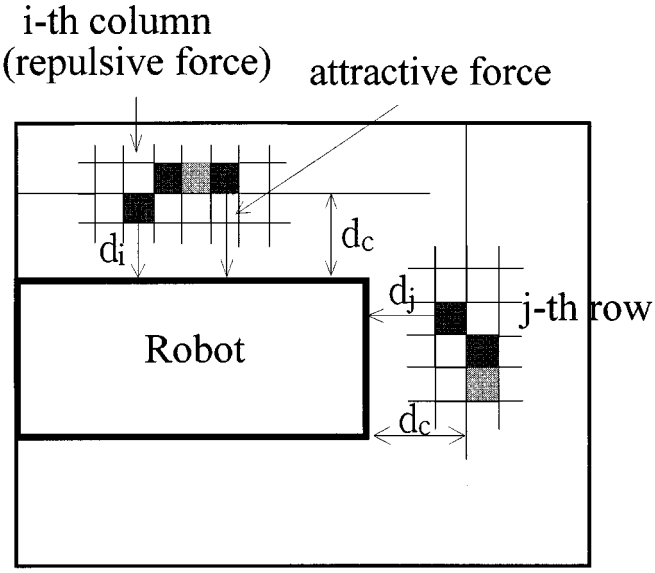


Fig. 5. Illustration of distance forces

where d_i denotes the distance from the robot to the nearest occupied cell in the column, d_c is the distance that the robot should maintain in its steady state (critical distatnce), and $P(o)$ is the occupancy probability of the cell. If there is no occupied cell in it, then f_i is the maximum attractive force, $-d_c^2$.

On the other hand, the cell in the *j-th* row belonging to the region of F_r exerts repulsive force (f_j) only, which is defined as,

$$f_j = \begin{cases} (d_j - d_c)^2 p(o), \; if \; d_j < d_c \\ 0, \; otherwise \end{cases} \quad (3)$$

in the sense that the main control forces are f_i only, thus f_j are auxiliary ones that prevent a collision with an object that may appear in front of the robot.

3.4.3 Control System. These forces belonging to each sub-region are summed into the resultant forces, F_B, F_L, F_F, and F_R. A moment, M_c, acting on the robot at current configuration of the command window is now calculated using the reusltant forces such that,

$$M_c = -F_B \frac{L_a}{2} + (F_L - F_R)\frac{L_b}{2} + k_c F_F \quad (4)$$

where k_c is a design parameter. To avoid the dependency on the choice of the d_c, M_c is normalized with respect to the maximum moment M_{max} which can occur when f_is are maximum ones, d_c^2, and F_F and F_R are zero. This normalized moment, M_n, is used as a command (Ω_c) of the steering rate of the robot, i.e.,

$$\Omega_c = k_p M_n \quad (5)$$

where k_p is also a design parameter. Fig. 6 shows the block diagram of the boundary following a control system. The reference moment, M_{ref}, is zero at all times, which means that the robot maintains the critical distance. Therefore, M_C itself becomes an error signal for this system. Subsequently the robot continuously changes only its running direction

https://doi.org/10.1017/S0263574798000587 Published online by Cambridge University Press

according to the steering command Ω_C with keeping its translation speed constant.

In eq.(4), k_c is required to be set in such a way that the robot can avoid the obstacle newly encountered in front of it without collision and continue to follow the boundary. A critical case is that the moment calculated from F_B and F_L are the maximum attractive one, $-M_{max}$, and the robot is faced with an obstacle in front of it. To avoid the obstacle and thus to continue to follow the boundary, the minimum value of k_c should be such that M_n has positive sign, i.e.

$$k_c > \frac{M_{\max}}{F_F} \tag{6}$$

If the value k_c is much smaller than that of Eq. (6), the robot may colide with an obstacle encountered in front of it. On the other hand, a large k_c will result in a lot of oscillation because the robot would detour around too much from the obstacle.

4 EXPERIMENTAL VERIFICATION

The exploration and boundary following control method were implemented and tested on a real robot in a real world. The robot is driven by two wheels powered by two DC motors at both sides, and has a dummy one at the middle of the front. It is equipped with 9 ultrasonic sensors (Murata, Japan) which have detection range of 0.05 m to 2 m. Sensor information from the robot is processed on an IBM-AT compatible 80586 computer.The sampling time for control was 0.1 seconds. The translation speed and maximum steering rate of the robot are 0.2*m/s* at steady state and 0.5 *rad/s*, respectively. For each sampling time 5 sets of sonar data together with the position information of the robot were acquired.

4.1 Experimental Results of boundary following Control for Path Tracking

Before the exploration experiments, the boundary following control method has been tested in a real world. In these experiments, d_c was set to be 0.25 m, and k_p to be 0.5 that is identical to the maximum steering rate of the robot so as to prevent the saturation of the steering command.

Fig. 7 shows the results of running experiments for a straight wall and a cornet one, respectively. The robot started from the position of 0.25 *m* (critical distance) apart from the wall with an orientation parallel to the wall, that is, no initial error is given. Initially, the command window could not provide sufficient information due to the lack of range data, which results in some oscillation as shown in the figures.

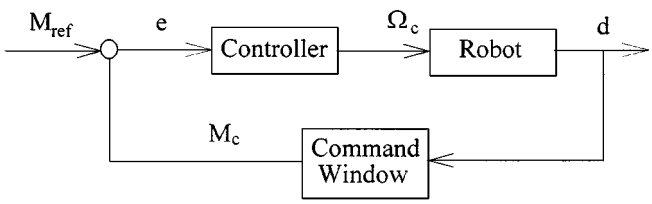
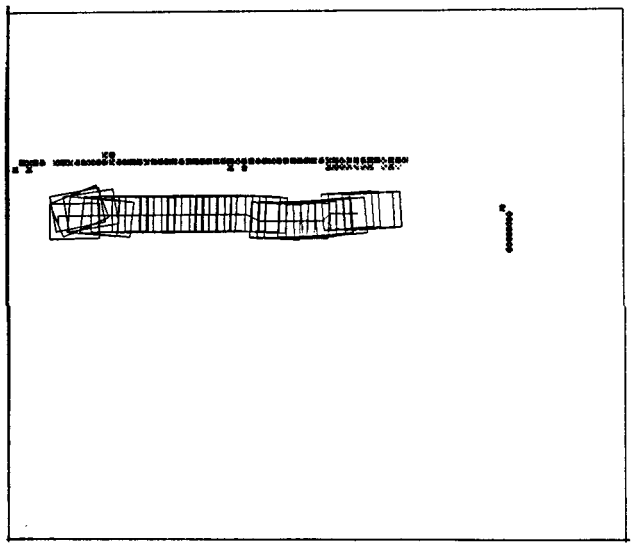


Fig. 6. Block diagram of the boundary following control system (d: distance from the object boundary to the robot)

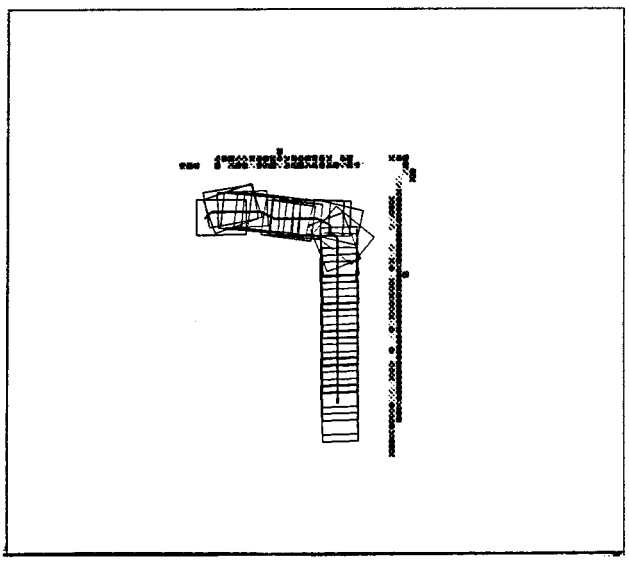
The effects of the initial error on the control performance are shown in Fig. 8. In both cases, the robot could start to trace the desired line after travelling distance of 0.5 m. At steady state, the maximum deviation from the desired path was less than one cell size (0.5*m*) in both sets of experiments. Therefore, the method is regarded to be quite useful for the exploratin of an unknown environment that does not require very accurate control.

4.2 Experimental Results of Exploration

The robot assumes that its initial position is the center of the workspace. The first sub-goal at level 0 is always taken to be outside the virtual boundaries, so that it first searches the



(a)



(b)

Fig. 7. Experimental results of boundary following control; (a) a straight wall, (b) a cornet wall

https://doi.org/10.1017/S0263574798000587 Published online by Cambridge University Press

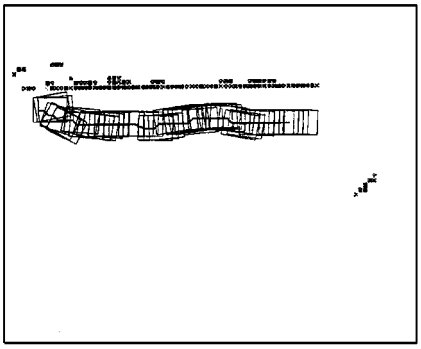
real boundaries of the workspace. After exploring the real boundaries of the workspace, it finds out 4 apexes (left, right, upper and lower ones) from the position data it has traced. The robot then constructs a rectangle using the apexes, which is the root node. If the size of real workspace is larger than the size of available memory in PC, the real workspace is divided into sub-workspaces, and the robot searches them sequentially. In the paper, we assumed that the area of the real workspace is smaller than the size of available memory.

Sensed data and position information from the robot are sent to the PC where a global map as well as a command window is updated. The exploration executor generates a Quadtree, and each center of the node is selected as a sub-goal. The path is then planned and the controller generates a steering command to execute the path. Whenever the searching of a node is completed, the map quality estimator looks into the node conditions to see if it is informed enough to be removed from the tree structure. This procedure is continued till the exploration executor tells the robot that there is no node left to explore.

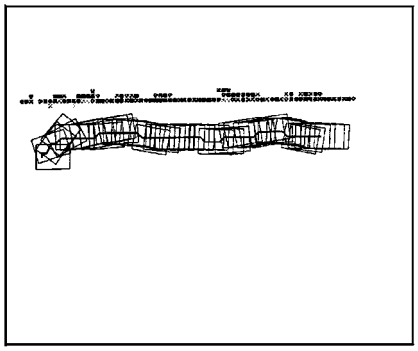
First, the exploration algorithms were implemented in a wide open area not having any obstacle in it in order to determine the node conditions. This is because if there exists at least one obstacle in the workspace, the traveling distance of the robot will be longer than that with no obstacle. This would result in acquiring more information on the environment. Therefore, exploration of a wide open area can be considered to be the case that the minimum amount of information is acquired given the same node size.

Fig. 9(a) shows when the robot returned to the hit point defined previously and decided that the first sub-goal (outside the virtual boundaries) could not be reached. Then it estimated the entropy of the root node, thta is, overall map. The estimates E_{nor} was 0.7 bits. Since E_{nor} was comparatively high the robot defined 4 nodes of level 1, which are represented as rectangles in the figure.

When exploring each sub-goal, the line from the robot's current position to the goal is represented as a temporal virtual wall so that the robot can execute the *Bug2* algorithm described in section 3. After reaching each sub-goal (the sequence is 3->2->1->4), as shown in Fig. 9(b), the robot again estimated E_{nor} for each node. The change of E_{nor} and the resulting one for each node are given in Fig. 10 and Table I, respectively. Looking into the binary and edge maps

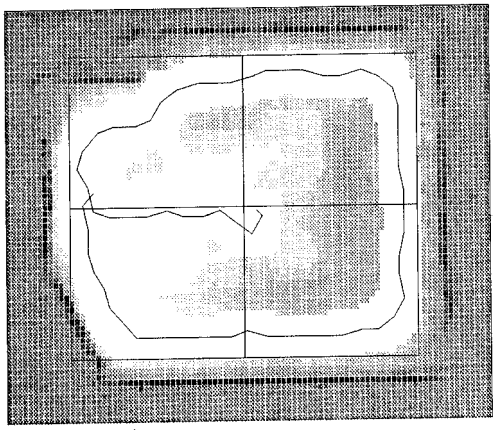


(a)

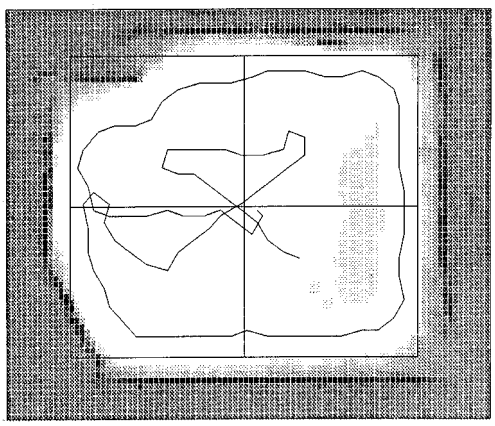


(b)

Fig. 8. Effects of initial error for control performance
a) when the robot started closer to the wall than d_c
b) when the robot started father from the wall than d_c



a) when the robot returned to the previously defined hit point



b) when the robot explored all nodes of level 1

Fig. 9. Exploration of a wide open area to determine the minimum node size N_s. The node definition is the same as that in Fig. 1, and the node sequence for exploration is 3->2->-1>4.
(a) when the robot returned to the previously defined hit boint, (b) when the robot explored all nodes of level 1

https://doi.org/10.1017/S0263574798000587 Published online by Cambridge University Press

shown in Fig. 11, one can find that all nodes are sufficiently informed and no further exploration is needed. From the results of this set of experiment, 0.32 bits and $1.4 \times 1.4\ m^2$ were chosen as E_T and N_S. These values may not be the optimal ones because they depend on the configuration of a robot, kind of sensors to be used, and the number of sensors. However, one can find the optimal ones easily from the method described above.

The node conditions determined in this way are applied to another environment that is composed of a small box, a metal cylinder, an acrylic box and walls. Fig. 12 shows the sequence of the rsulting map, and the edge map is shown in Fig. 13. The nodes defined by the robot were the same as those in Fig. 9 because the 4 apexes from the real boundaries were identical to those in Fig. 9. As expected, the robot's path is much more complicated than that of the previous experiments. All nodes satisfied the node conditions after visiting each center of them, so that the exploration terminated at level 1. Nodes 1, 2, and 3 satisfied the entropy condition, while node 4 satisfied the condition of minimum node size. As shown in Fig. 13, the boundaries of objects are somewhat distorted from the true ones represented as solid lines. This is due to the dead reckoning error that tends to be accumulated over a traveling distance.

5 CONCLUSIONS

This paper has addressed a systematic exploration method for an autonomous mobile robot to acquire information about its unknown workspace effectively. The method is

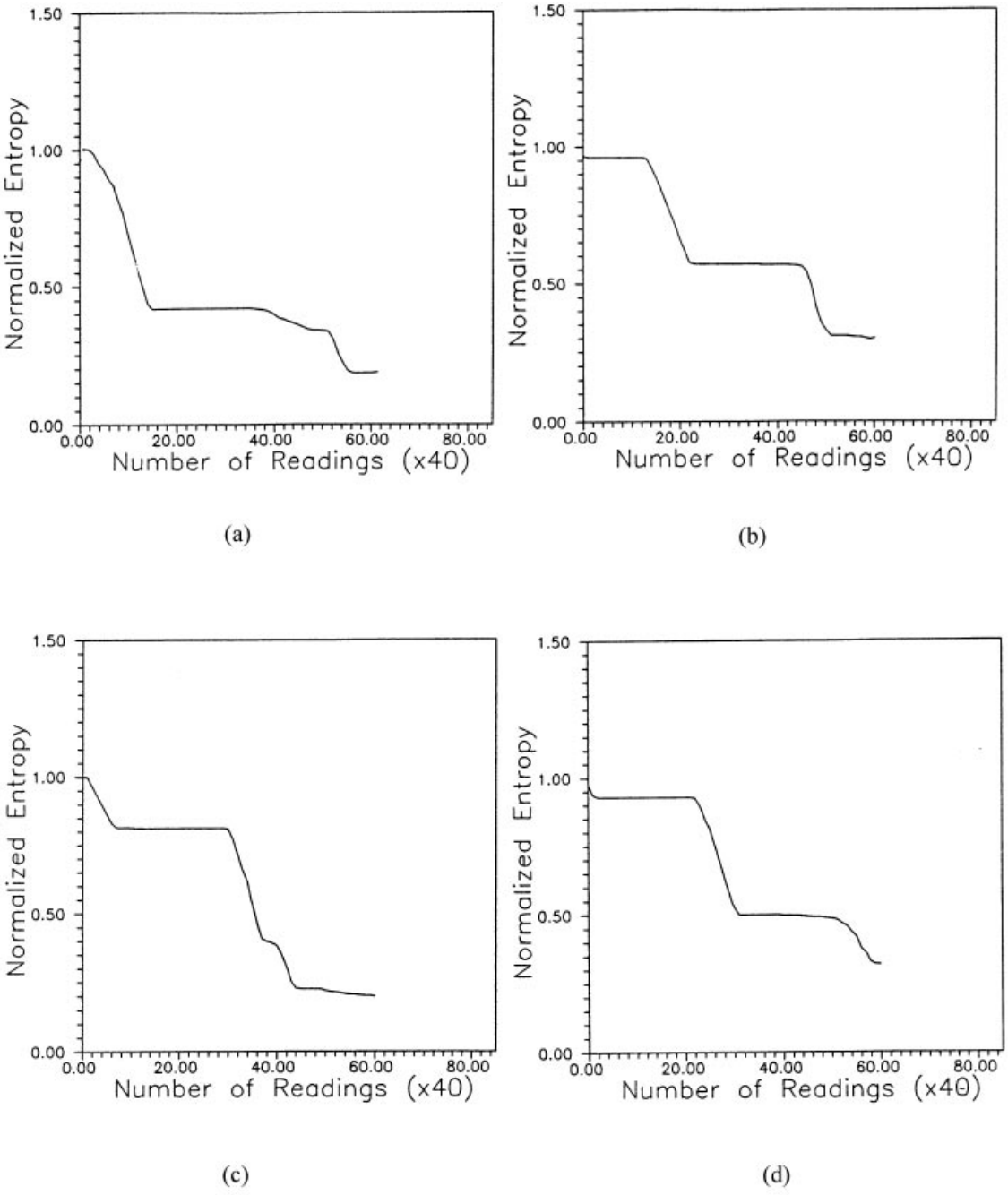


Fig. 10. Entropy change of each node; (a) node 1, (b) node 2, (c) node 3, (d) node 4

https://doi.org/10.1017/S0263574798000587 Published online by Cambridge University Press

Table 1. E_{nor} for each node of level 1.

Description	*Normalized Entropy (E_{nor})*	*Probability corresponding to E_{nor}*
Node		
1	0.20	0.0311
2	0.32	0.0581
3	0.18	0.0271
4	0.30	0.0532
Maximum	0.32	0.0581

composed of three parts: (1) Quadtree representation of a workspace to generate a series of sub-goals. (2) Boundary following control to visit each sub-goal based on distance forces exerted by the boundaries of objects near the robot. (3) Estimation of map quality by using the concept of entropy and the condition of the minimum node size of the tree.

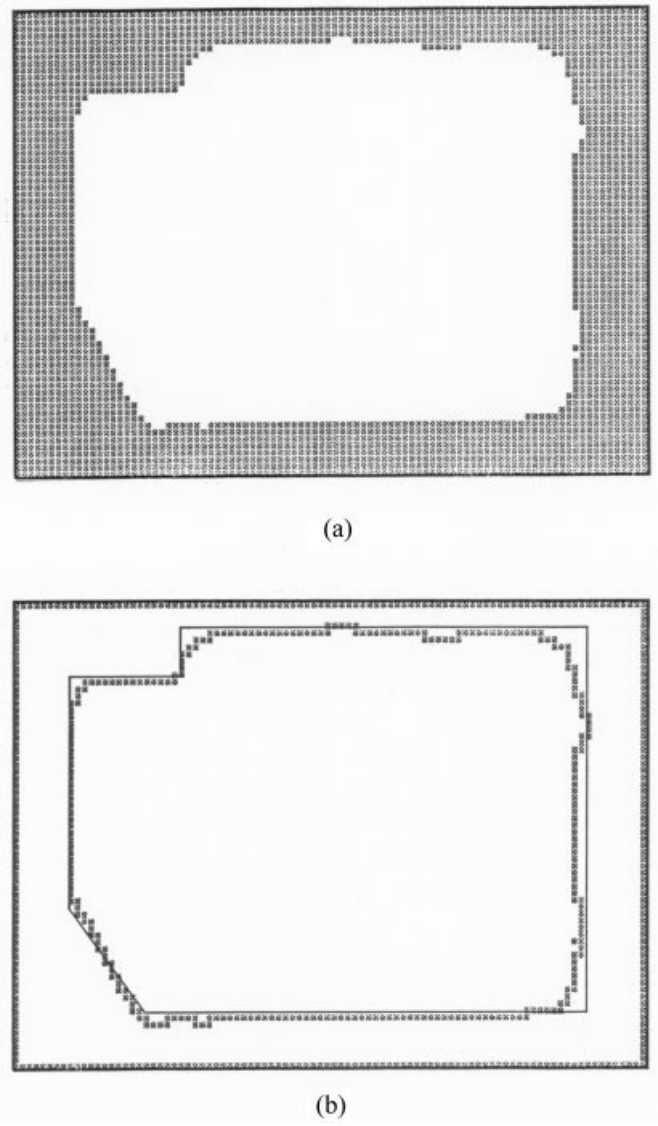


Fig. 11. (a) The bindary map (threshold probability = 0.37), (b) The edge map

The idea of Quadtree was introduced to decompose the overall workspace into sub-nodes. This decomposition allows the robot to generate successive sub-goals to visit, and to estimate the quality of a node independently of the rest. Therefore, sufficiently informed nodes can be dis-

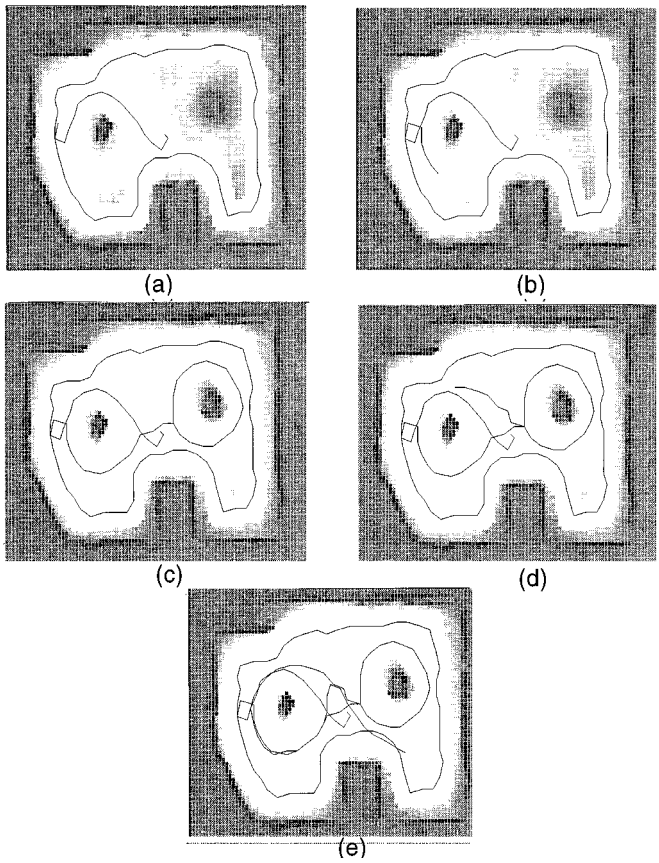


Fig. 12. Experimental Results of exploration (nodes defined by the robot and the sequence are identical to those in Fig. 9)
(a) when the robot reached the first sub-goal
(b) when the robot reached the node 3 of level 1
(c) the robot could not visit the center of node 2 because it was not reachable
(d) when the robot reached the node 1 of level 1
(e) when the robot reached the node 4 of level 1

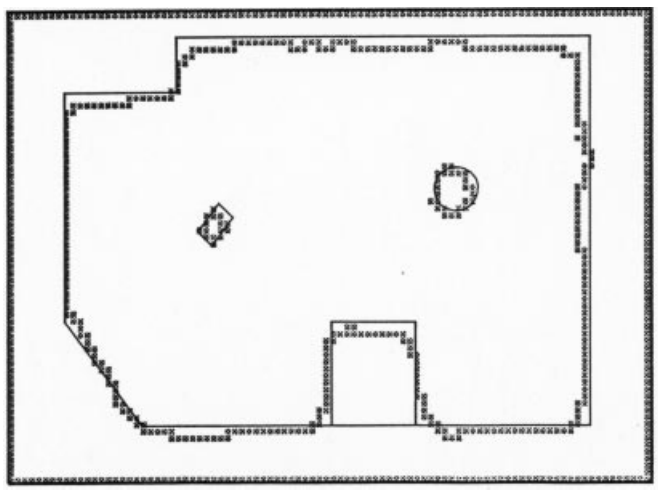


Fig. 13. The edge map (threshold probability = 0.41)

https://doi.org/10.1017/S0263574798000587 Published online by Cambridge University Press

carded from the area to be explored, which results in the reduction of excessive exploration costs.

In the method of boundary following control, the idea of distance forces was introduced to generate a steering command based on the relative configuration of the robot to the object boundaries appeared in the command window. This method is robust to noise under the specular environment because it uses not only the current range data but also the accumulated information. The proposed exploration method was implemented and tested in a real world, and proved to be efficient for the construction of a map of the robot's surroundings.

ACKNOWLEDGMENT

This work has been supported by the Korea Science and Engineering Foundation (No. 951–1008–025–2), 1995 3.1 ∼ 1997. 2.28.

References

1. C.J. Taylor and D.J. Kriegman, "Exploration Strategies for Mobile Robots" IEEE *International Conference on Robotics and Animation* (1993), pp. 248–253.
2. N.J. Nilsson, *Problem Solving Methods in Artificial Intelligence* (McGraw-Hill, New York, 1971).
3. C.E. Thorpe, "Path Relaxation: Path Planning for a Mobile Robot" *Technical Report* (Carnegie Mellon University Robotics Institute, CMU-RI-TR-84-5 1984).
4. T. Lozano-Perez and M. A. Wesley, "An Algorithm for Planning Collision-free Paths among Polyhedral obstacles" *Commun. ACM* **22**, 560–570 (1979).
5. L. J. Crowley, "Navigation for an Intelligent mobile robot" *Technical Report* (Carnegie Mellon University, CMU-RI-TR-84-18, 1984).
6. R. Brooks, "Solving the Find-path Problem by Representing Free Space as Generalized Cones" *IEEE Trans. on System, Man and Cybernetics* **SMC-31**, No. 3, 190–197 (1983).
7. O. Takahashi and R. J. Schilling, "Motion Planning in a Plane Using Generalized Voronoi Diagrams" *IEEE Transactions on Robotics and Automation* **5**, No. 2, 143–150 (1989).
8. J. Borenstein and Y. Koren, "The Vector Field Histogram-Fast Obstacle Avoidance for Mobile Robots" *IEEE Transactions on Robotics and Automation* **7**, No. 3, 278–288 (1991).
9. J.H. Lim and D.W. Cho, "Experimental Investigation of Mapping and Path Planning for an Autonomous Mobile Robot based on Certainty Grids using Sonar Sensor" *Robotica* **11**, Part 1, 7–17 (1993).
10. V. Lumelsky and A. Stepanov, "Dynamic Path Planning for a Mobile Automation with Limited Information on the Environment" *IEE Transactions on Automatic Control* **31**, No. 11, 1058–1063 (1986).
11. V. Lumelsky, J. Vladimir, A. Stepanov and A. Alexander, "Path Planning Strategies for a Point Mobile Automation Moving Amidst Unknown Obstacles of Arbitrary Shape" *Algorithmica*, (1987).
12. E. Gat, "Robust Low-Computation Sensor-driven Control for Task-Directed Navigation" *IEEE Internal Conf. on Robotics and Autonomous*, Scaramento CA, (1991) pp.
13. P. Turennout, G. Honderd and L. J. Schelven, "Wall-following Control of a Mobile Robot" *IEEE International Conference on Robotics and Automation*, Nice France (1992), pp. 280–285.
14. T. Skewis and V. Lumelsky, "Experiments with a Mobile Robot Operating in a Cluttered Unknown Environment" *IEEE International Conference on Robotics and Automation*, Nice France (1991), pp. 1482–1487.
15. D. W. Cho, "Certainty Grid Representation for Robot Navigation by a Bayesian Method", Robotica **8**, Part 2, 159–165 (1990).
16. J. H. Lim and D. W. Cho, "Physically Based Sensor Modeling for a Sonar Map in a Specular Environment" *IEEE International Conference on Robotics and Automation*, Nice France (1992), pp. 1714–1718.
17. J. H. Lim and D. W. Cho, "Specular Reflection Probability in the Certainty Grid Representation", *Transactions of ASME, Journal of Dynamics Systems, Measurements and Control* **116**, No. 3, 512–520 (1994).
18. J. H. Lim and D. W. Cho, "Multipath Bayesian Map Construction Model from Sonar Data" Robotica **14**, Part 2, 527–540 (1996).
19. J. R. Woodworker, "The Explicit Quadtree as a Structure for Computer Graphics," *The Computer Journal* **25**, No. 2, 235–238 (1982).

https://doi.org/10.1017/S0263574798000587 Published online by Cambridge University Press