Path and Sensing Point Planning for Mobile Robot Navigation to Minimize the Risk of Collision

Keiji NAGATANI and Shin'ichi YUTA

Institute of Information Science and Electronics,
University of Tsukuba,
Tsukuba, 305 JAPAN

Abstract

In this paper, a new path planning algorithm for mobile robotics is proposed. When a mobile robot is navigating, the error in estimated position of the robot increases in proportion to the traveling distance. Because of this error, the robot has a greater risk of collision with objects in the environment. To reduce this error, robot should obtain environment information by using its environment sensors, and modify it's estimated position accordingly. In this paper, the cost function for the mobile robot's path which minimizes the risk of collision, and the algorithm to find the optimal path and sensing points are proposed. The authors implemented the proposed algorithm on real robot system, and made several experiments to verify this algorithm.

1 Introduction

When an autonomous mobile robot navigates to a goal in the real environment, using an environment map, the following steps are needed.

[**1**] Planning a path to a destination by use of the environment model which the robot has.

[**2**] Executing the planned path in the real environment.

Usually, the first step is processed in the abstract world, and the problems to cope with the real physical world are treated in the second step. The mobile robot usually uses a dead-reckoning system, i.e. an internal sensor system, which estimates the position and orientation of the robot by counting the physical rotations of its wheels, and uses some external sensor system to recognize its surrounding environment.

Using these physical sensors, the robot should cope with the following problems during execution of the second step of the navigation.

(1) Unexpected Obstacle

If unexpected obstacles, for which information is not included in the environment map, exist on the planned path of the robot, the robot can not run along the path, and therefore it can not reach its goal.

(2) Fatal Error in the Map of the Environment

When there is a major error in the map of the environment which the robot possesses, the robot can not continue to navigate toward its goal. For example, a closed road in the real environment which is not in the map, the robot can not reach its goal.

(3) Variance of the Estimated Position

Using a dead-reckoning system, the error between the estimated position and the real position becomes significant. In this paper, this error is called **the error in estimated position**. When the traveling distance of the robot is small, this kind of error is small and can be ignored. However, if the distance is longer, a larger error is accumulated. If the robot only uses dead-reckoning, it will leave the planned path on account of this error, and the robot may collide with objects in the environment.

(4) Modeling Error of the Map

When making a map of the environment that a robot uses, physical measurements must be transformed into a computer model. Therefore the chances of errors in the computer model due to inexact or incorrect measurement comes into existence. Because of this modeling error, the robot may collide with objects in the environment.

These problems will prevent the robot from safely executing along the planned path. If the robot has a sufficiently powerful external sensor and processing capability, it can recognize the problem situation and avoid the collision with objects in the environment.

However, it is impossible to realize such a perfect external sensor for a mobile robot working in the real environment. Therefore, the approach should be how can we decrease the possibility of collision with objects in the environment.

Methods to reduce the possibility of collision are considered for each of the above stated problems, as follows.

For problems (1) and (2), it is not possible to anticipate where the robot comes into collision with obstacles on the planning step. So, there is no way other than checking the surroundings of the robot by use of a more powerful external sensor at all times in the execution step.

In the problem(3), the situation of the robot colliding with objects in the environment can be anticipated, this is because the error in estimated position has a certain statistical property which becomes larger cumulatively. So if there are some signs or natural landmarks by which the robot can adjust its estimated position, the size of the error can be reduced, and the possibility of collision can be minimized. The procedure to reduce the error before it becomes large, is as follows:

1. Get information about the sign or landmark in the real environment by using an external sensor of the robot.
2. Compare it with information stored in the environment model that the robot possesses.
3. Calculate robot's position and orientation based on the result of the comparison.
4. Adjust the estimated position and orientation of the robot

The place, where the robot uses the external sensor to observe the sign or landmark, is called a **sensing point**. Navigation in the real environment requires adjusting the estimated position on a regular basis. However, there may not be many good places for sensing signs or landmarks. Thereby, making correction of the position is a difficult problem. Therefore, in the first step of navigation, not only a path but also sensing points should be considered in advance , so that the robot can execute the planned path more safely.

Considering problem(4), a possible place where a robot collides in the environment can be partially foreseen. This is because the possibility of collision depends on the safely conditions of the path. For example, the path may be very close obstacles, and so it is dangerous. The Robot can reduce the possibility of collision with objects by planning safest possible path at the first planning step.

As mentioned above, the possibility of collision with obstacles in the environment can be reduced by sophisticated planning of the path and sensing points in advance, i.e. in the planning stage of robot navigation.

In this paper, the authors propose an algorithm that can plan path and sensing points for navigation of a mobile robot in the real environment which minimizes the risk of collision, and report on results of experimentation.

2 Path Evaluation Including the Consideration of Positioning Errors

When a robot is navigated from it's current position to it's goal, we consider that the most important objective is arriving at the goal without colliding with any objects. If the robot has a sufficient external sensor to acquire accurate information about objects in a real dynamic environment, it is not necessary to consider collision with objects. But in the present state of robot sensor technology, the performance of external sensors is insufficient, and the good way of insuring safety is by planning paths with the smallest possibility of collision.

Based on the above discussion, a method for path evaluation which considers the possibility of collision with objects is proposed.

2.1 Proposal of Evaluating Function

The longer a robot runs, the larger errors in the estimated position will be in the dead-reckoning system. Also there can exist modeling errors in the environment map. Let's consider the possibility of collision with objects in the environment caused only by these two errors. In this paper, this possibility is called the **possibility of collision**.

Let the possibility of collision in each place be denoted as $u(x)$, where variable x is the travel distance from the start point. The value of this function denotes a kind of probability of collision. Note that the variable x is not a random variable and $u(x)$ is not a probability density function.

Meanwhile, if the robot does not move, it will not collide with objects in the environment even though errors in its estimated position and environment modeling are large. In other words, the risk of the collision with objects in the environment is generated by the action of the robot's motion. Therefore, the risk of collision on

a path can be thought as the integration of $u(x)$ along the path. Therefore, the risk of collision on the path denoted as J is considered as,

$$J = \int_L u(x)dx$$

where L is the path.

On the other hand, adjusting the estimated position of the robot has a cost. For example, using a sensor takes time. Such costs should also be considered by reducing useless frequent sensing. This sensing cost is expressed by S and it will be incremented for each sensing action.

From the above, the optimum path is the path that minimizes the following function. This function is defined as the evaluation function of the path based on the risk of collision and the cost of sensing,

$$E(path) = k_1 J + k_2 S$$

where k_1 and k_2 are weights for the costs.

2.2 Properties of the Possibility of Collision

The possibility of collision at each place can be considered as the sum,

$$u(x) = u_1(x) + u_2(x)$$

where, $u_1(x)$ is the possibility of collision caused by error in the estimated position, and $u_2(x)$ is the possibility of collision caused by modeling errors in the map.

The possibility of collision caused by the positioning error $u_1(x)$ has the following properties,
(1) the value of $u_1(x)$ depends on the path that robot has executed to its current position, and
(2) value of $u_1(x)$ increases monotonically between sensing points.
If it is assumed that every position adjustment at a sensing point causes a equal resultant error variation, the value of $u_1(x)$ reduces to a fixed value.

The $u_2(x)$ term in the possibility of collision function depends on the amount of free space surrounding the path. For example, the narrower the width of the path, the larger the value $u_2(x)$ has.

Figure 1 is a example of the change of $u(x)$ and the total value of evaluation function for one path. In this figure, the sum of all the shaded areas denote the cost of the path.

In the traditional approach to path planning, the estimated cost of each point on a path is independent of the condition of the robot, i.e. the history of the trajectory. It means that $u_1(x)$ is regarded as 0 and only $u_2(x)$ is considered in proposed evaluation. However, in navigation of the real environment, $u_1(x)$ must be considered because an error in estimated position is more important than a modeling error in the map.

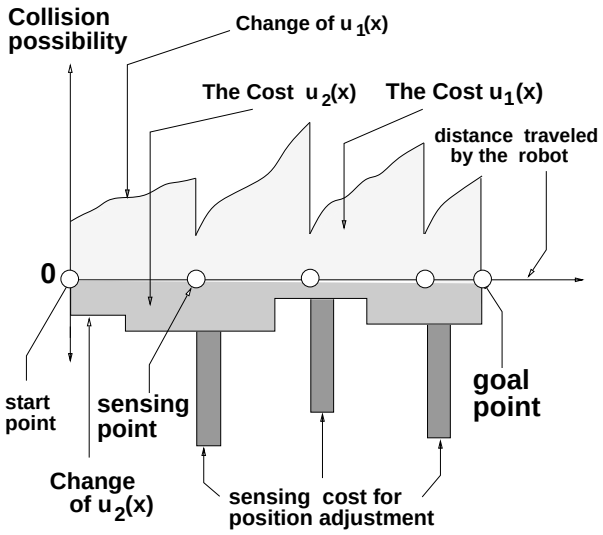

Figure 1: An Example of the Change of $u(x)$ and Total Evaluation Function of the Path

3 Path Planning Algorithm

3.1 A Graph Construction

As an example of an environment, the roadway-network of Figure 2 is adopted. It is assumed there are several possible sensing points to adjust the robot's estimated position.

The network of roads can be expressed by a graph shown in Figure 3 using of following procedure.

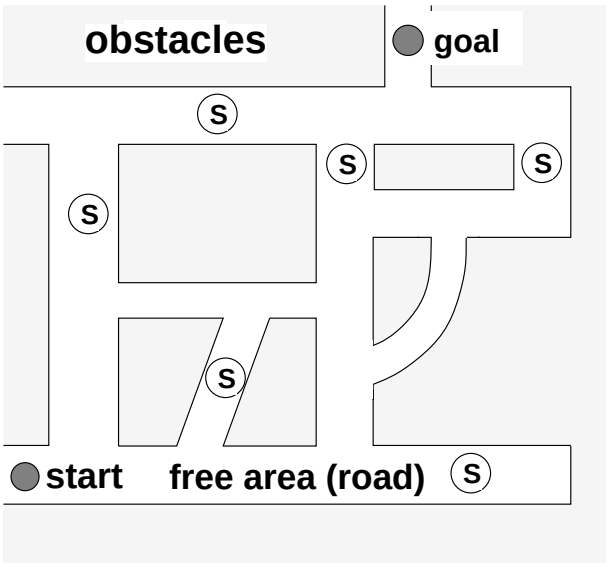

Figure 2: An Example of Environment

- A crossing or branch of road is expressed as a node.
- The current position of the robot and goal point are expressed as additional nodes.
- Sensing points are also regarded as nodes.
- Each road section is expressed by bi-directional arcs which connect two nodes. Each arc stores the length of the path as cost.
- Closed arcs are added to the nodes of the sensing points. Such arcs have a fixed sensing cost.

Using this graph, the path and sensing point planning problem can be regarded as the path search problem on the graph. Path and sensing points are planned by applying the cost evaluation function which is proposed in section 2.

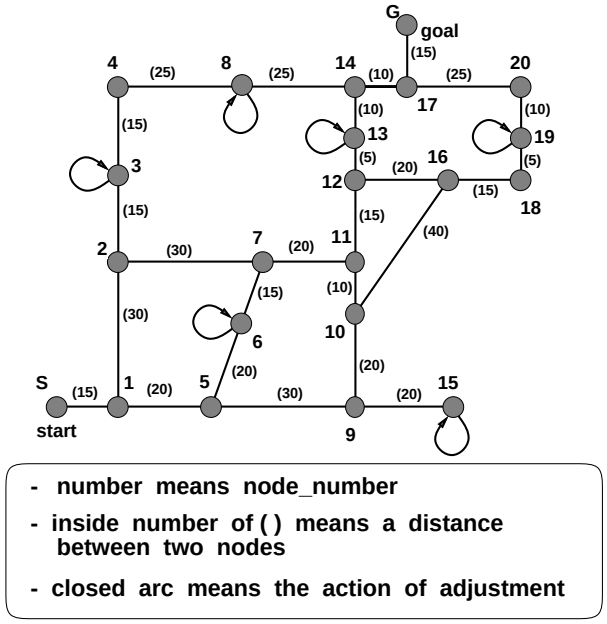

Figure 3: An Example of Graph

3.2 Planning Algorithm

Applying the proposed evaluation function, the cost for an arc is expressed as,

$$\text{cost} = \begin{cases} k_1 \int_{\text{node}}^{\text{next\_node}} (u_1(x) + u_2(x))dx & : \text{roadway sections} \\ k_2 s & : \text{sensing points} \end{cases}$$

where 's' is a fixed sensing cost.

The problem in the graph search using this cost is that the cost is not fixed in each arc because $u_1(x)$ depends on where the robot has run. Therefore, a traditional method for the graph search, such as DIJKSTRA's algorithm [1], can not be applied for this problem.

Under these condition, it is useless for a robot to go twice through the same section of path more than once, and adjust it's position twice in the same place. In other words, if a path is optimal, there is no arc on the graph which is used more than once. Applying this qualification to graph search, the numbers of paths which are connected from start node to goal node are limited. Therefore, the optimal path is found by comparing the total costs of all paths.

However, if a graph is complicated, such full graph search is not good because of combinational explosion of the number of paths. Therefore, the numbers of paths should be reduced by using the following property.

Property:

Assuming that there are two paths, called $path1$ and $path2$, which are connected from a start point to one node(Figure 4). If the next condition is satisfied, then $path1$ can be deleted.

$$u_{path1} > u_{path2} \quad \text{and} \quad J_{path1} > J_{path2}$$

where 'u' is possibility of collision at the connecting node for in each path, and 'J' is the total cost of each path from start node to this connecting node.

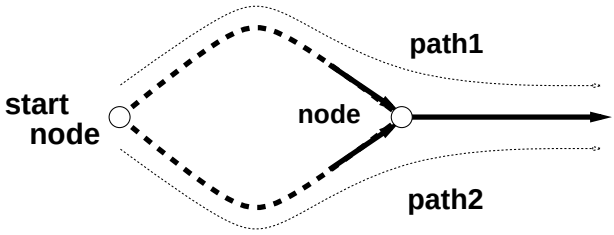

Figure 4: Example of Possibility of Deletion Path

Using this property in the search of the graph, results in a much faster search than in a full graph. Also there is no problem with comninational explosion of the number of paths.

3.3 Example

To show the effectiveness of the path reduction algorithm presented in section 3.2, the number of comparisons of the cost of graph search is compared between the proposed algorithm and full graph search.

For example, consider the simple graph shown in figure 5. When a full graph search is done on this graph, the number of possible cost calculations is 2172. In other words, 2172 represents the number of possible node visits. By using path reduction condition, the number of possible cost calculations become only 61. For other examples, the numbers of possible cost calculations are calculated in Table 1. This table states the algorithm that the proposed in section 3.2 can suppress combinational explosion.

The usefulness of the proposed algorithm is shown by using one example. The graph of the environment is shown in Figure 4. For simplicity, navigation functions and fixed values are defined as,

$$\begin{aligned} u_1(x) &= 2x + u_0 \\ u_2(x) &= 10 \\ A &= 30 \\ k_1 &= 1 \\ k_2 &= 1 \\ s &= 100 \end{aligned}$$

Example of Graph	Node Num	Arc Num	Full Graph Search	Using Proposed Algorithm
Figure5	8	18	2172	61
Figure6	10	24	50280	85
Figure3	22	58	large unknown number	1102

Table 1: Comparison between examples of graphes

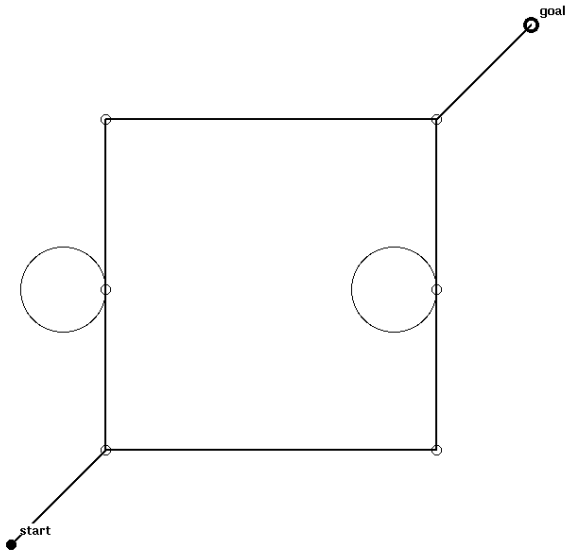

Figure 5: An Example of Simple Graph

Where 'u_0' is the possibility of collision with obstacles caused by an error in the estimated position at entry of an arc. 'A' is a fixed value for the lower bound of the possibility of collision after adjustment at a sensing point. 's' is a fixed value of cost per sensing.

For example, the $u_1(x)$ at $node1$ can be calculated as follows,

$$\begin{aligned} u_1(x) &= 2x + u_0 \\ &= 2 * 15 + 0 \\ &= 30 \end{aligned}$$

and the cost of the path from $start\_node$ to $node1$ can be can be calculated as follows,

$$\begin{aligned} J_{node1} &= \int_{node}^{next\_node} (u_1(x) + u_2(x))dx \\ &= \int_0^{15} ((2x + 0) + 10)dx \\ &= 375 \end{aligned}$$

Consider another example in Figure 3. Here node 7 has multiple ways of reached paths. Let path [s-1-5-6-6-7] be defined as $path1$ and path [s-1-2-7] be defined

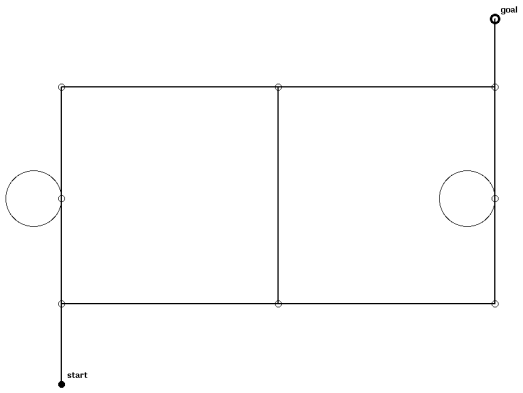

Figure 6: An Example of Simple Graph

$path2$. J of $path1$, J of $path2$, $u_1(x)$ of $path1$ and $u_1(x)$ of $path2$ can be calculated as follows.

$$\begin{aligned} u_1(x)_{path1} &= 60 \\ J_{path1} &= 4500 \\ u_1(x)_{path2} &= 150 \\ J_{path2} &= 6375 \end{aligned}$$

At node 7, following condition occurs.

$$J_{path1} < J_{path2} \quad \text{and} \quad u_1(x)_{path1} < u_2(x)_{path2}$$

Therefore $path2$ can be ignored after graph search.

Finally, the least cost of the path to the goal node can be found by using the calculations shown above. The final path to the goal is shown in Figure 7. In the solution path, $node6$ and $node13$ are sensing points for adjusting the robot's position. Figure 8 shows the change of $u(x)$ and total-cost of the determined path.

4 Adaptation to Real Navigation

This section discusses how to adapt the evaluation function proposed in section 2 to navigation in a system. The authors has implemented a real navigation system on their experimental mobile robot and have made several experiments. In these experiments, the autonomous mobile robot YAMABICO which is developed by our group was used.

The YAMABICO is a small-sized robot which uses wheeled locomotion. This robot possesses a dead-reckoning system which counts the number of rotations of its left and right wheels. The robot uses a RADIAN sensor (Rotary Acoustic DIrection ANgle sensor)[6]. This sensor consists of one ultrasonic transmitter, two receivers

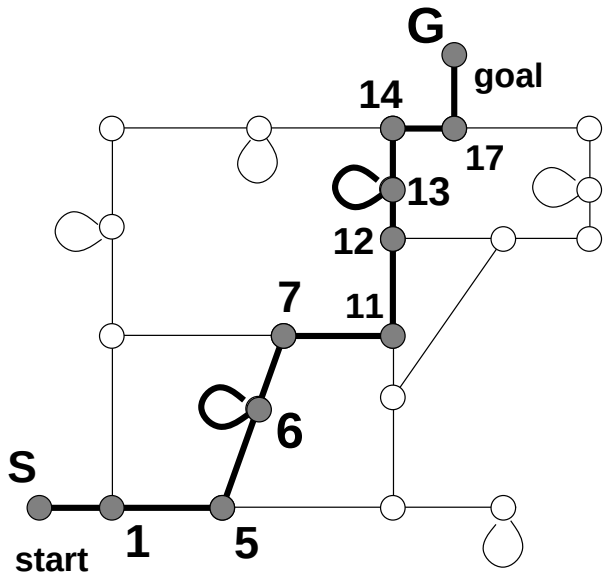

Figure 7: The Result of the Path and Sensing Points Planning

and stepping motors to move the base of this sensor. It can measure not only the distances between robot and flat-wall, but also the orientation of the robot against the flat-wall, by using an ultrasonic pulse echo method.

In the experiments, the YAMABICO can adjust its posture by using the RADIAN sensor. The process for adjusting is :

1. , Estimating its position and orientation by using dead-reckoning and the internal model of the environment.
2. Measuring real distances to the flat wall and real orientation against flat-walls by using RADIAN sensor.
3. Calculating the error in the estimated position and orientation.
4. Modify the robot's estimated position and orientation.

Using the RADIAN sensor, flat walls are easily sensed landmarks. Regarding a walls as a sensing points, a graph representing the map of our laboratory is shown in Figure 9. In this figure, the gray areas are obstacles areas. The black lines which are located on the sides of obstacles are flat walls which the robot can detect with the RADIAN sensor. The lines in free area are the sections of paths. The circles on the sections of path are the places where the robot can adjust its estimated position using the RADIAN sensor.

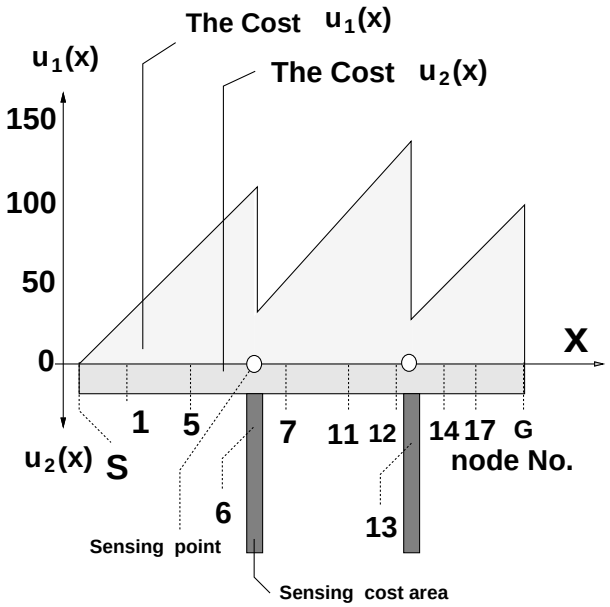

Figure 8: Change of u(x) in This Example

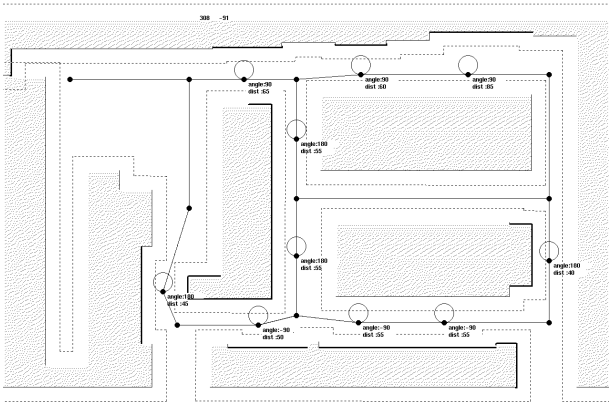

Figure 9: Laboratory Map and Graph

In this experiment, the robot possesses graph model(Figure 9). When a goal position is given the YAMABICO, the current position and the goal position are added to the graph as nodes. Using this adjusted graph, the robot could make a graph search and plan the path and sensing points.

The graph search program provides the determined path and sensing points to the navigation program.

The navigation program moves the YAMABICO from its current point to the given goal using the RADIAN sensor at each planned sensing point[2][3].

The system which included graph search and path execution was fully implemented. The calculation time by an on-board computer(68000,10MHZ) was 3 seconds for planning. The planned path was executed and it required 80 second to arriving to the goal (total length of path of about 12 meters), including about 5 seconds sensing at each sensing position.

We performed many experiments new algorithm on our robot. In every experiment, the robot ran safely and was able to complete its mission.

5 Conclusion

In this paper, the possibility of collision with objects in the environment during robot navigation was considered. A method for evaluate the risk of collision and cost of navigation paths was proposed. Also it was considered how to apply such evaluation methods to actual robot systems operating in the real environments. The proposed algorithm was implemented on actual robot and several experiments were made, which verified the effectiveness of our method.

In general, the rate of reduction of error in the estimated position depends on the orientation of sensing. However, in this paper, the reduction rate was assumed to be same in all orientations to simplify this problem. So, in next step, the reduction rate for each orientation of travel of the robot should be considered. Another subject of investigation is extending the algorithm to cope with other kinds of sensors.

[3] S.Iida and S.Yuta : "Vehicle Command System and Trajectory Control for Autonomous Mobile Robots", in Proc. of IEEE/RSJ Int. Workshop on Intelligent Robots and Systems, pp212-217,1991.

[4] Maki K.Habib and H.Asama : "Efficient Method to Generate Collision Free Paths for Autonomous Mobile Robot Based on New Free Structuring Approach", in Proc. of IEEE/RSJ Int. Workshop on Intelligent Robots and Systems,pp563-567,1991.

[5] K.Komoriya, E.Oyama, K.Tani : "Planning of Landmark Measurement for the Navigation of a Mobile Robot", in Proc. of IEEE/RSJ Int. Workshop on Intelligent Robots and Systems, pp1476-1481,1992.

[6] Y.Nagashima and S.Yuta : "Ultrasonic sensing for mobile robot to recognize an environment", in Proc. of IEEE/RSJ Int. Workshop on Intelligent Robots and Systems, pp805-812,1992.

Acknowledgement

The authors express their gratitude to the members of Intelligent Robot Laboratory of University of Tsukuba, and to Dr. Alex Zelinsky of the University of Wollongong, Australia for his useful suggestions in the preparation of this paper.

References

[1] E.W.Dijkstra, "A Note on Two Problems in in Connection with graph", in Numerische Mathematik, 1, pp269-271, 1959.

[2] S.Suzuki and S.Yuta : "Analysis and Description of Sensor Based Behavior Program of Autonomous Robot Using Action Mode Representation and ROBOL/0 Language", in Proc. of IEEE/RSJ Int. Workshop on Intelligent Robots and Systems, pp1497-1502,1991.