IEEJ Journal of Industry Applications
Vol.5 No.3 pp.253–260 DOI: 10.1541/ieejjia.5.253

**Paper**

# Experimental Verification of Path Planning with SLAM

Yudai Hasegawa* Non-member, Yasutaka Fujimoto* Senior Member

(Manuscript received June 1, 2015, revised Dec. 4, 2015)

Simultaneous localization and mapping (SLAM) is a very popular technique which is used to develop autonomous mobile robots. We have developed an autonomous mobile robot that can perform the SLAM based solely on information it gathers with a laser range finder. Path planning using the A* algorithm is proposed to help the robot determine the shortest path while avoiding obstacles and minimizing travel distance and rotation. In addition to the standard eight adjacent cells present in conventional A* algorithms, the proposed path planning method allows the eight cells that may be reached via the knight move to be defined as additional adjacent nodes. As a result, the achieved path is smoother than those obtained via more conventional methods, as has been experimentally verified.

**Keywords:** SLAM, path planning, autonomous mobile robot, A* algorithm, Dijkstra's algorithm

## 1. Introduction

In recent years, due to aging among the general population, the worker population has significantly decreased. This decrease is expected to continue in the future. Accordingly, the development of robotic technology is essential because it will enable robots to fill in gaps in the worker population for various fields and for work sites that are dangerous or even impossible for humans to enter. For example, load-carrying robots have been developed with the goal of using them in under staffed hospitals [1], and a hydraulically actuated hexapod robot has been developed to move across uneven surfaces at dangerous disaster sites [2]. However, before such robots can be used in real-world applications, they must first be able to recognize their surrounding environment and make decisions on how to act accordingly. This ability is often referred to as "autonomous locomotion".

Autonomous locomotion comprises three important techniques: map building, localization, and path planning. Map building is conducted to reconstruct the environment surrounding the robot; localization is conducted to estimate the robot's position on the map once it is built; and path planning is conducted to determine the best path to a destination while avoiding collisions. Simultaneous localization and mapping (SLAM) is a popular technique that simultaneously generates the map and estimates the robot's position. Currently, several mapping methods exist. The laser range finder (LRF) is the most popular sensor used to measure the distance from a robot to its surrounding obstacles. Most SLAM studies use odometers and LRF sensors. One such study utilizes a Kalman and particle filters to unify the information obtained from both types of sensors [3]. Others build a map of the surrounding environment using camera images; these, however, tend to involve an increased number of required calculations because depth information is not available. Thus, to decrease the number of required calculations in methods using camera images, use of stereo cameras has been recommended [4][5]. However, use of stereo cameras with SLAM has only been confirmed in static, indoor environments; it has not been confirmed for use in dynamic, outdoor applications.

When developing autonomous mobile robots, it is useful to use a large number of sensors to improve the robot's performance. However, doing so increases the cost of the system as well as the calculation costs required. Thus, to reduce the system and calculation costs, and assuming that the working area is a two-dimensional plane, our autonomous mobile robot uses only one LRF as a sensor. We chose LRF because of its low noise levels and its suitability as a single sensor. We use scan matching to calculate the quantity of movement from the geometric characteristic points provided by the LRF. A typical method of scan matching, the iterative closest point (ICP) algorithm, was proposed for use with the robot; however, although the ICP algorithm can conduct localization without odometers, most studies using it have only tested it in static environments. For this reason, we choose to use the particle swarm optimization (PSO) algorithm instead; PSO is one of the most precise and cost-effective optimization algorithms currently available. Using it, our robot is able to perform autonomous locomotion in outdoor environments [7][8]; however, path planning for obstacle avoidance has not yet been implemented.

Low-cost, real-time path planning methods are necessary. One such method uses hierarchical roadmap representations [9] that make real-time path planning possible; however, changes in the robot's head angle at passing points are quite large because the grid map used in that method has a large grid size. Other studies have investigated using the A* and D* algorithms [10][11], which are based on the same underlying algorithm: Dijkstra's. The A* algorithm is useful in finding the shortest path and is, therefore, popular in mobile robot navigation applications [12]. Meanwhile, the D* algorithm adapts well to environments that have moving obstacles and is also, therefore, popularly applied to mobile robot navigation [13][14]. Indeed, if the A* algorithm is used with high frequency, it can also adapt to the environment. Both the

* Yokohama National University
79-5, Tokiwadai, Hodogaya, Yokohama, Kanagawa 240-8501, Japan

A* and D* algorithms plan the same path, but the calculation cost for each is different. Conventional path planning for mobile robots using the A* and D* algorithms face two problems:

1) A sequence of middle goals from start position to goal position is designed beforehand as waypoints, and the A* and D* algorithm are used to find the shortest path from current position to the next waypoint. However, discontinuity of the robot's head angle may occur at waypoints because each waypoint connects two shortest paths. If the change in head angle is large at a waypoint, the robot may stop and pivot turn; such rapid movement is dangerous in public places because of the risk of collisions with people.

2) To find the shortest path on the SLAM-generated grid map, eight cells that are adjacent to the current cell are regarded as adjacent nodes in the A* algorithm. This means that the robot's head angle only has eight possible directions with a minimum unit angle of 45 degrees. Therefore the robot's minimum rotation angle is 45 degrees. Consequently, generated paths are not smooth.

We therefore propose a modified method of path planning that is based on the A* algorithm and that takes the robot's head angle into account. Furthermore, in addition to the eight adjacent cells that are regarded as adjacent nodes in that algorithm, we also regard the eight cells reachable by the knight move as adjacent nodes. Therefore, the robot's head angle has 16 possible directions and the paths generated by the proposed method are smoother than those generated by conventional methods. We have verified the proposed path planning method and compared it with a conventional method through the use of several experiments.

## 2. System Structure

**2.1 Robot Structure** Figure 1 shows our robot's exterior. The robot is based on a commercially available electric wheelchair, on which is mounted an LRF (UTX30-LX, Hokuyo Automatic Co., Ltd), a main PC and a monitoring PC. The LRF is attached to the front of the wheelchair at a height of 580 mm above the ground; its specifications are summarized in Table 1.

**2.2 Robot's Velocity** Our robot is designed for use in public places, where there will be pedestrians who must not be disturbed; thus, the robot must not travel at a velocity that makes pedestrians feel unsafe. We have therefore designed the velocity of our robot to vary based on its distance from the nearest obstacle. Table 2 lists the maximum allowed velocities based on the robot's distance from its nearest obstacle and assuming that the robot is operating on a flat surface. When obstacles exist on the path to the robot's next waypoint, it is necessary to plan an alternative path that avoids the obstacles. In our system, the frequency of the velocity command is 0.1 s and depends on the computational time required to solve the SLAM problem. Therefore, our robot can avoid collisions with moving obstacles whose velocities are less than 8.4 km/h.

## 3. SLAM

Here, we introduce our method for solving the SLAM problem.

**3.1 MAP** A grid map is reconstructed from the information obtained from an LRF. We convert the raw data from the LRF sensor onto an occupied grid-map. The size of a cell represents the dimensions of a square in the real environment corresponding to one pixel of the grid map. The cells of the grid-map are set at 20×20 mm$^2$, and the value of the cell at the coordinates $(x, y)$ is defined as $MAP(x, y) \in \{0, 1\}$. The initial value of $MAP(x, y)$ is 0. When there is an obstacle at $(x, y)$, $MAP(x, y) = 1$.

Our method uses three grid maps: the local-map, current-map, and global-map. The local-map is obtained from the latest single LRF scan, the current-map is reconstructed from the local-map; and the global-map is an old current-map that has been reconstructed in advance. Figure 2 provides a conceptual diagram of all three maps.

**3.2 Localization** The position and head angle of the robot are defined as $(x_0, y_0)$ and $\theta_0$, respectively. The robot's movement in a sampling period $\delta t$ is defined as $\delta x, \delta y, \delta\theta$. Sampled data from an LRF contain movement information. Thus, by matching the latest data to a map built from past data, we can estimate the robot's movement. When doing so, the robot uses the PSO algorithm to minimize Eq. (1). $f(z)$ is the ratio of points that have been matched. Based on the number of matched points, $f(z)$ evaluates how well the local-map matches either the current-map or the global-map:

$$f(z) = \frac{N_{valid} - N_{fit}(z)}{N_{valid}} \cdots\cdots (1)$$

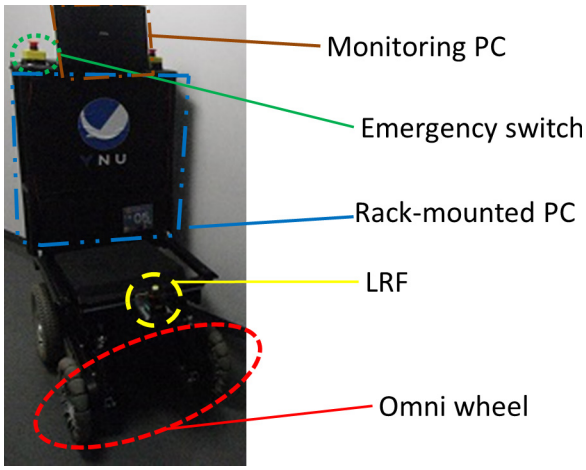



Fig. 1. An exterior of our robot

Table 1. Specifications of UTM-30LX

| provision | specification |
|---|---|
| Size (W×D×H) | 60 mm×60 mm×87 mm |
| Voltage | DC 12 V±10% |
| Current | under 0.7 A |
| detection distance | 100–30,000 mm |
| scan angle | 270 deg |
| precision (from0.1 m to 10 m) | ±30 mm |
| precision (from10 m to 30 m) | ±50 mm |
| angular resolution | 0.25 deg |
| scanning time | 25 ms/scan |

Table 2. Velocity of our robot

| distance to an obstacle | upper limit of velocity |
|---|---|
| 0–300 mm | 0.0 km/h |
| 300–1500 mm | 2.4 km/h |
| 1500–3000 mm | 3.4 km/h |
| over 3000 mm | 4.0 km/h |

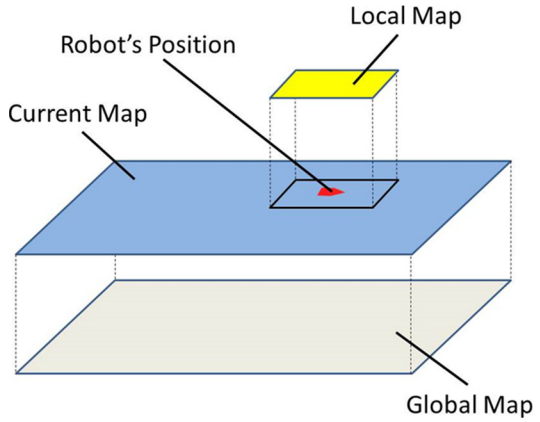



Fig. 2. Conceptual diagram of the robot's maps

where $z = (\delta x, \delta y, \delta\theta)$ represents the robot's movement. $N_{valid}$ is the number of samples obtained from the LRF in a single scan, which ranges in distance from 150 mm to 30,000 mm. $N_{fit}$ is defined as follows:

$$N_{fit} = \sum_{i=1}^{n_{max}} m(d_{pi}) MAP(x_i, y_i) \cdots\cdots\cdots\cdots\cdots\cdots (2)$$

$$x_i = d_{pi} \cos\left(\theta_0 + \delta\theta + \alpha i - \frac{3\pi}{4}\right) + x_0 + \delta x \cdots\cdots\cdots (3)$$

$$y_i = d_{pi} \sin\left(\theta_0 + \delta\theta + \alpha i - \frac{3\pi}{4}\right) + y_0 + \delta y \cdots\cdots\cdots (4)$$

$$m(d_{pi}) = \begin{cases} 0.1 & (MAP(x_i, y_i) = 1,\ 0.15 \geq d_{pi} \leq 5) \\ \dfrac{d_{pi}}{100} + 0.7 & (MAP(x_i, y_i) = 1,\ 5 < d_{pi} < 30) \\ 0 & (otherwise) \end{cases}$$
$$\cdots\cdots\cdots\cdots\cdots\cdots (5)$$

where $d_{pi}$ is the distance from the LRF to the obstacle directly obtained by the LRF; $(x_i, y_i)$ is the coordinates of the point on the map that corresponds to the distance $d_{pi}$; $m(d_{pi})$ is a weighting function of the matching point; $\alpha$ is a minimum resolution of the LRF sensor, $(0.25\pi/180\,\text{rad})$; and $n_{max}$ is the number of valid points (1080).

The PSO algorithm finds the optimal solution $z$, which—since $f(z)$ evaluates the number of matching points—minimizes $f(z)$ and matches the latest data set (the local-map) to the map built from past data (either the current-map or the global-map). In the matching, the number of measured points is important, but that number decreases as the distance between the measured points and the sensor increases—this is why $m$ changes depending on $d_{pi}$.

Once the optimal solution $\delta x, \delta y, \delta\theta$ is obtained, the robot's movement is determined—namely, the robot's position is updated as follows:

$$x_0 := x_0 + \delta x \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (6)$$
$$y_0 := y_0 + \delta y \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (7)$$
$$\theta_0 := \theta_0 + \delta\theta \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (8)$$

These procedures are performed at every LRF scanning period, which occur every 100 ms in our robot.

**3.3 Mapping** Once the robot's position is determined (see Section 3.2), we can update the current-map according to that information as well as the latest scan data (in the form of Eqs. (3) and (4)). The local-map consists of a set of data $(x_i, y_i)$, $i = 1, ..., n_{max}$, that is generated by Eqs. (3) and (4) using both the latest scan data $d_{pi}$, $i = 1, ..., n_{max}$, and the optimized movement $\delta x, \delta y, \delta\theta$. The current-map is updated by superimposing the local-map onto it. We utilize neither an odometer nor GPS information to build the current-map and the global-maps.

When superimposing the local-map onto the current-map, obstacle points in the local-map are added to the current-map. This method can build very precise current-maps for static environments but not for environments with dynamic obstacles (such as pedestrians), in which case the past loci of obstacles remain on the current-map. To improve the accuracy of the current-map, we add the following process to eliminate such loci.

1) For each cell within 50×50 m$^2$ around the robot (in a grid-map of 2500×2500), the number of times that the cell encounters an obstacle is counted. After every ten scans, the obstacle on the cell is eliminated if that number is less than three.

2) Obstacles on a line segment between the robot and the latest obstacle are eliminated.

These processes drastically improve the accuracy of the current-map.

## 4. Path Planning for Obstacle Avoidance

Reference waypoints, through which the robot must pass, are set as far as the final goal. The robot plans a path to its next waypoint using the A* algorithm. Figure 3 shows a flowchart of such path planning with obstacle avoidance. A grid map for obstacle avoidance is first regenerated from a map generated by SLAM. To reduce calculation costs, the number of cells is reduced from that of the map originally generated by SLAM. Next, expansion processing is executed for the grid map to account for the size of the robot. These steps allow us to regard the robot as a point occupying only one cell on the map—instead of needing to consider every cell that the robot actually occupies. At this point, the A* algorithm is used to find the shortest path with obstacle avoidance.

**4.1 Map Generation for Obstacle Avoidance** There are too many SLAM-generated cells on the grid map for real time path planning to take place. Thus, it is necessary to reduce the computational complexity; we do this through expansion. For example, the size of one grid cell is set to be 80×80 mm$^2$ (these dimensions were achieved by averaging the original 20×20 mm$^2$ grid map). Expansion processing is then conducted for an 80×80 mm$^2$ grid map, thus accounting for the size of the robot. The amount of obstacle expansion used is important: too small an expansion will lead to collisions and too large will lead to narrow routes disappearing during the path planning stage. Because the width and length of our robot are 70 mm and 100 mm, respectively, we set an expansion radius of 400 mm, which means that—in an 80×80 mm$^2$ grid map—the maximum expansion radius is 564 mm.

**4.2 A* Algorithm** The A* algorithm finds the

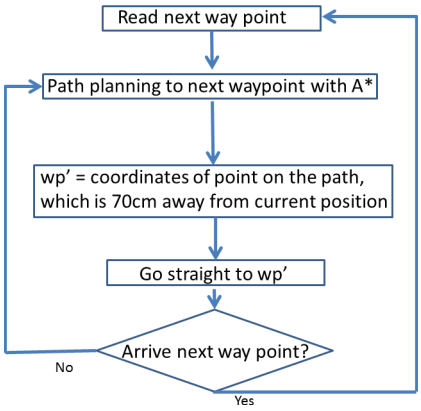



Fig. 3. Flowchart of obstacle avoidance

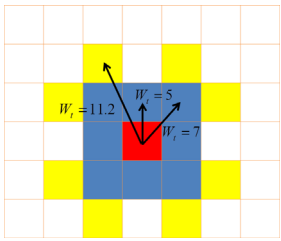



Fig. 4. Proposed adjacent nodes

shortest path, through nodes, from the start-node to the goal-node. Each node corresponds to a single cell of the grid map, and there are an equal number of nodes and cells. The A* algorithm finds the shortest path to the goal-node by keeping the following function $f_A(n)$ as low as possible;

$$f_A(n) = g(n) + h(n) \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots \quad (9)$$

where $f_A(n)$ is the evaluation function in the $n$-th node, $g(n)$ is the lowest cost from a start-node to the $n$-th node, and $h(n)$ is the estimated lowest cost from the $n$-th node to the goal-node. This algorithm is guaranteed to find the optimum solution as long as $h(n)$ is not more than the true cost from the $n$-th node to the goal-node.

### 4.3 New Adjacent Node in the A* Algorithm

Usually, adjacent nodes in the A* algorithm are defined as being the eight cells immediately adjacent to a node in the vertical, horizontal, and diagonal directions. In this paper, however, the eight cells reachable by the knight move are also defined as additional adjacent nodes. Figure 4 illustrates these proposed adjacent nodes. The blue cells in that figure represent conventional adjacent cells, and the blue-and-yellow cells represent the proposed adjacent cells. Using these proposed adjacent cells, the path is made smoother because, from the node, we may search a total of 16 nodes as opposed to the eight nodes conventionally searched.

### 4.4 New Evaluation Function in the A* Algorithm

It is necessary to define the movement cost of each adjacent node and the appropriate evaluation function for each because the goal of the A* algorithm is to find the shortest path with the lowest costs. By not considering the robot's head angle when defining the costs and evaluation function, the path generated by the A* algorithm is guaranteed to be the shortest in terms of total travel distance, but it may not be smooth. We therefore define the costs and evaluation function as follows to prevent quick turn on the path:

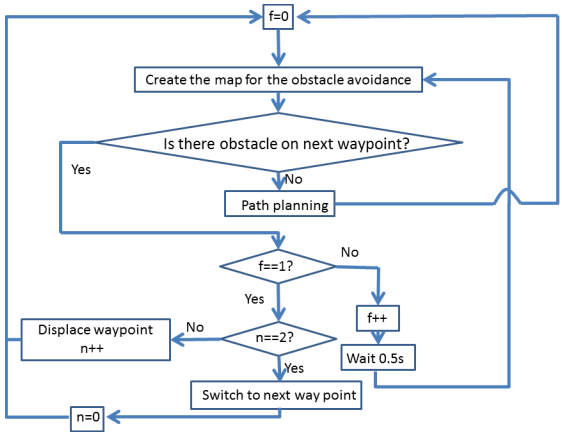



Fig. 5. Flowchart of a waypoint displacement

$$g(n) = W_t + W_r \left|\theta - \theta_{prev}\right| + C_{prev} \cdots\cdots\cdots\cdots\cdots\cdots \quad (10)$$

where $W_t$ is a movement cost, $W_r$ is a cofficient of a rotation cost, $\theta$ is a head angle (degree), $\theta_{prev}$ is a head angle in the previous node, and $C_{prev}$ is a cost of the previous node. $W_t$ is designed as shown in Fig. 4. $W_r$ is designated as 0.16 based on trial and error.

### 4.5 Heuristic Function

In Eq. (9) we must estimate the lowest cost from the $n$-th node to the goal-node, $h(n)$, because the total computation cost can be reduced if $h(n)$ is close to the true cost. In the conventional method, this function can be easily calculated from the Pythagorean Theorem. When accounting for the robot's head angle, however, calculation of this function requires information regarding the robot's position, the next waypoint, and the robot's head angle. At the $n$-th node, calculation of $h(n)$ (taking into consideration the robot's head angle) is defined as

$$h(n) = W_t \sqrt{\left|i_{wp} - i_n\right|^2 + \left|j_{wp} - j_n\right|^2} + W_r\phi \cdots\cdots \quad (11)$$

where $(i_n, j_n)$ is the coordinates of the $n$-th node, $(i_{wp}, j_{wp})$ is the coordinates of the next waypoint, and $\phi$ is the defference between an angle to the next waypoint and the robot's head angle at the $n$-th node.

### 4.6 Waypoint Displacement

If there is an obstacle on the goal-node, the A* algorithm fails to find a solution and to plan a path. To avoid this situation, we have to displace waypoint in such the case. Our system changes position of the waypoint toward the direction of the next waypoint. Figure 5 shows a flowchart illustrating this process. If there is an obstacle on next waypoint, the robot waits 1 s for the obstacle to retreat. If the obstacle does not retreat, the robot displaces the waypoint by interpolating the next two waypoints. For example, the waypoint at $(i_{new}, j_{new})$ may be displaced as follows:

$$i_{new} = \beta i_{wp2} + (1-\beta) i_{wp1} \cdots\cdots\cdots\cdots\cdots\cdots \quad (12)$$

$$j_{new} = \beta j_{wp2} + (1-\beta) j_{wp1} \cdots\cdots\cdots\cdots\cdots\cdots \quad (13)$$

where $(i_{wp1}, j_{wp1})$ is the coordinate of the next waypoint, $(i_{wp2}, j_{wp2})$ is the coordinate of the waypoint after the next waypoint, and $\beta$ is the parameter of displacement, which is designed to be 0.2. If the waypoint must be displaced in a third time, the robot moves that waypoint to the next waypoint.

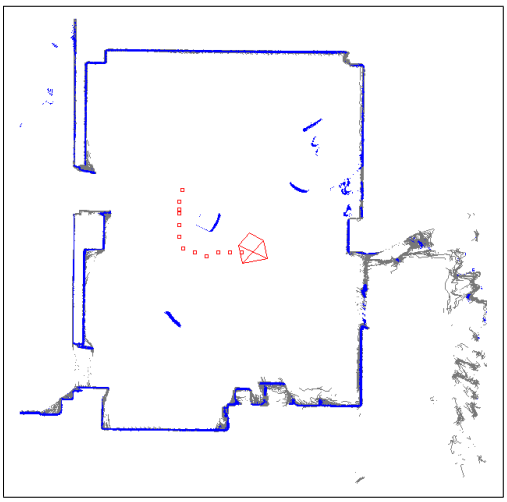


Fig. 6. Route A as generated by the conventional method

Fig. 7. Route B as generated by the conventional method

## 5. Experiment

**5.1 Verification of the New Cost Function** For comparison, we first solve problems according to the conventional A* method. Figures 6 and 7 show two routes as planned according to the conventional method. In these figures, blue indicates current LRF information and gray indicates information from the input map; the pentagon represents the robot, with the sharp apex representing its front; and the dotted line represents the robot's planned path. Two separate planned paths result, each with nearly equivalent costs. In this situation, the robot moves straight forward while shaking its head from side to side, and it may stop moving altogether if its distance from an obstacle becomes too short. In the proposed method, however, because the robot's head angle is taken into consideration (see Section 4.4), larger cost differences exist between paths, which means that the probability of two paths existing with nearly equivalent cost is negligible when the proposed method is used.

Figure 8 shows the path from the current waypoint to the next waypoint as generated by the conventional method. In this planned path, the robot immediately pivots because the change of its head angle is too big, resulting in rough robotic movements. Note that this situation could happen at every waypoint. In comparison, Fig. 9 shows the planned path that results from the proposed method, which accounts for the robot's head angle. In this case, the robot does not have to

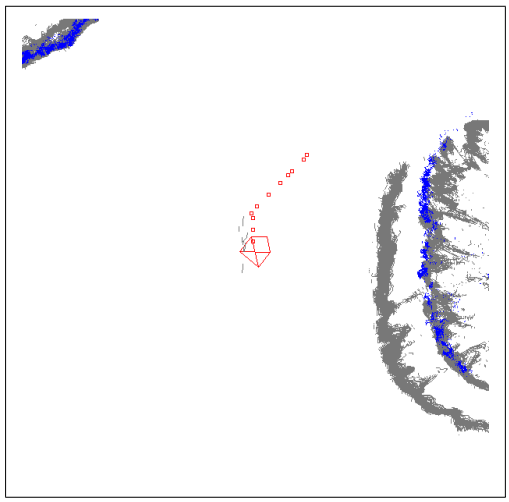


Fig. 8. Planned route without considering the robot's head angle at the waypoint

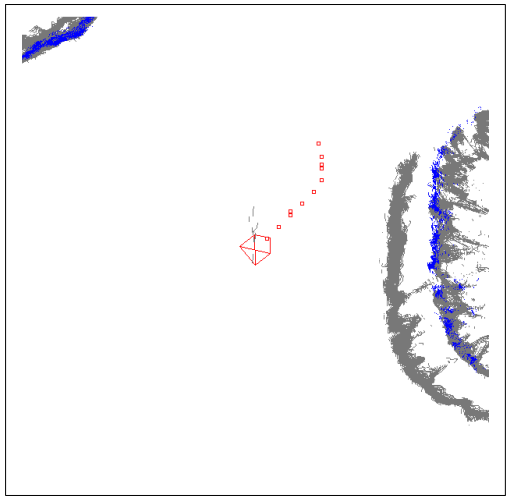


Fig. 9. Planned route resulting from proposed method, which does consider the robot's head angle at the waypoint

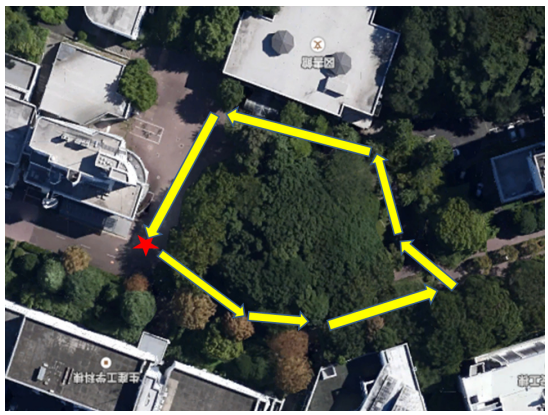


Fig. 10. Experimental environment

pivot because the change in its head angle is quite small.

**5.2 Verification of New Adjacent Nodes** These experiments were conducted on the campus of Yokohama National University, which is shown in Fig. 10. In that figure, the star indicates the robot's initial position and the arrows show its routes. The robot solved for SLAM prior to the experiment and generated the map shown in Fig. 11, using the methods described in Section 3. Using this map, the robot was then able to travel automatically.

Figure 12 shows the robot's loci as calculated with and without using the proposed additional adjacent nodes (see Section 4.3). Note that the locus obtained by the proposed method is smoother than that obtained by the conventional one. Figures 13(a) and 13(b) show route flows that resulted from the conventional method, which does not use additional adjacent nodes.

As those figures show, the robot occasionally turned in

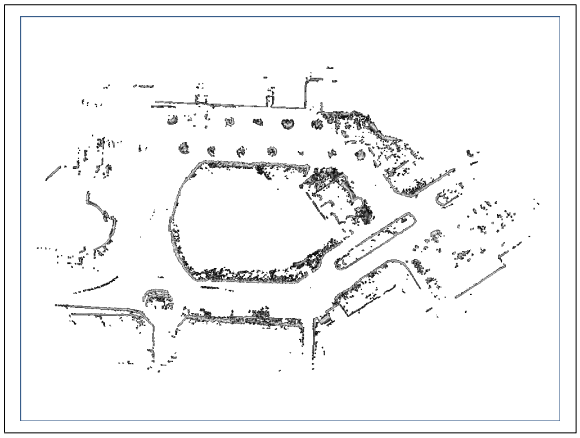


Fig. 11. SLAM-generated map

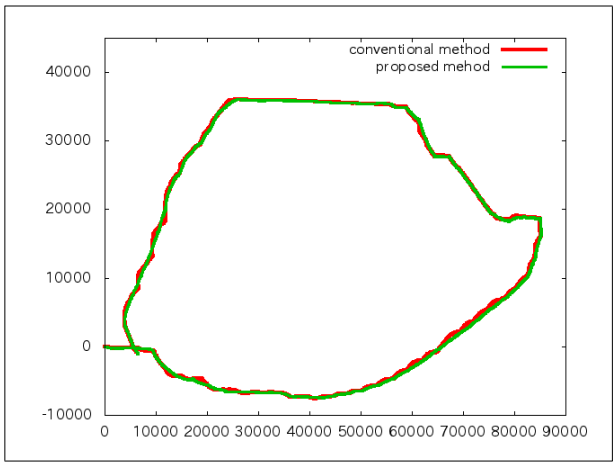
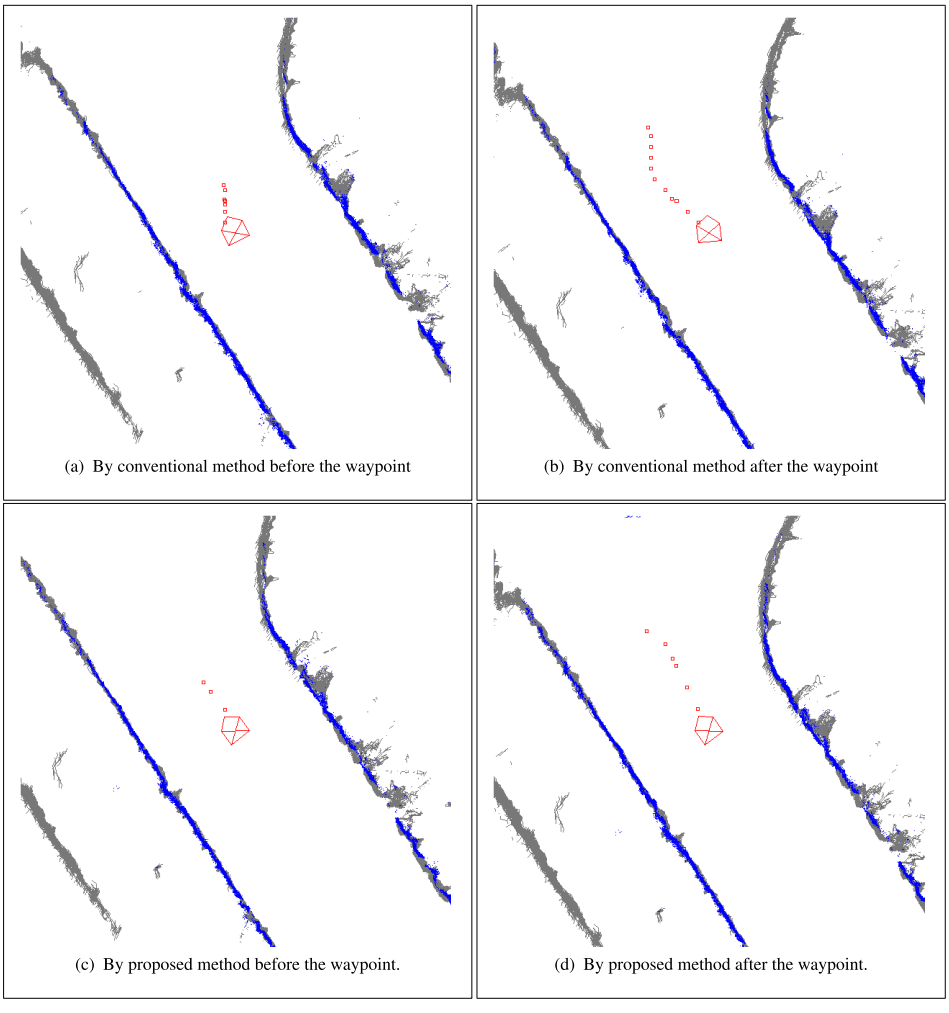



Fig. 12. The robot's loci as calculated by two methods



(a) By conventional method before the waypoint

(b) By conventional method after the waypoint

(c) By proposed method before the waypoint.

(d) By proposed method after the waypoint.

Fig. 13. Verification of proposed additional adjacent nodes

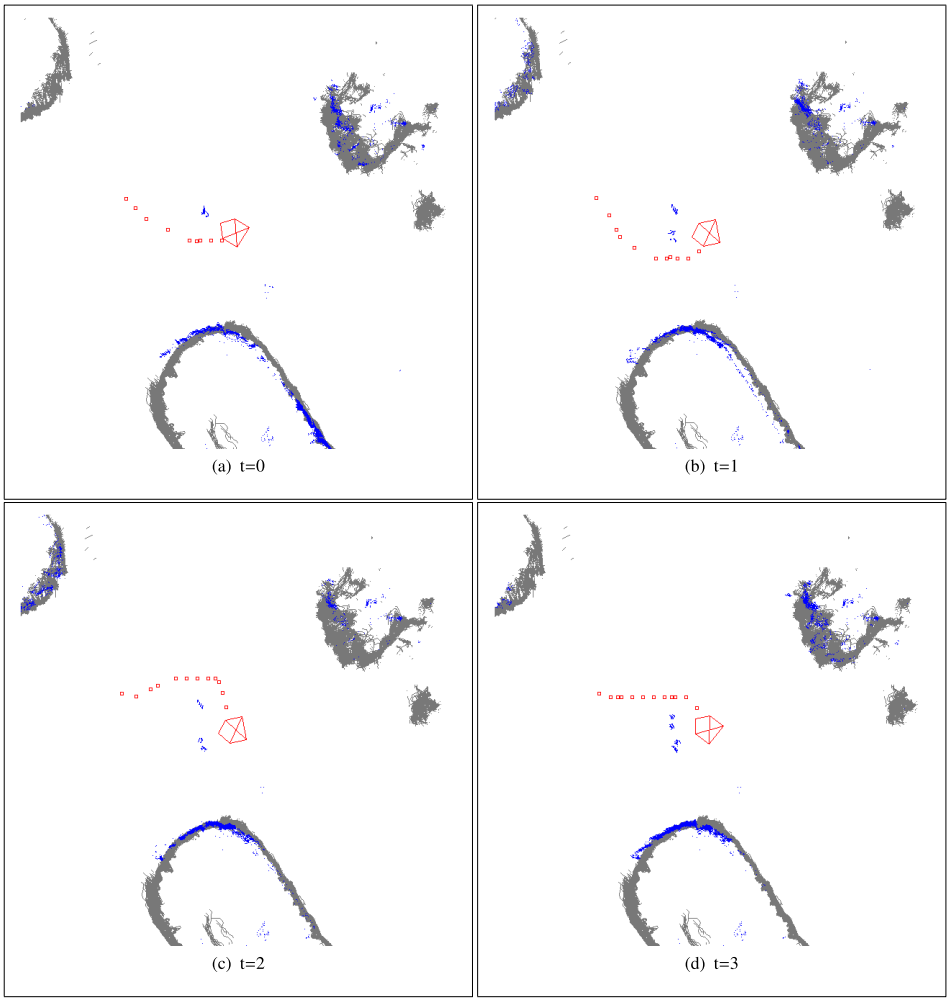



Fig. 14. The time flow of route avoiding dynamic obstacles

place because the change in its head angle was too large before and after several waypoints. On the other hand, Fig. 13(c) and Fig. 13(d) show the route flows that resulted from the proposed method; the same areas were used for these routes as for the routes obtained from the conventional method. Because the proposed method uses additional adjacent nodes, it is able to select more precise directions, thus significantly decreasing changes in the robot's head angle before and after waypoints. Table 3 shows the travelling distances and standard deviation of the rotation angles for both the conventional and proposed methods. Those results also indicate that the proposed method results in smoother path flows than does the conventional one.

### 5.3 Path Planing Velification with Dynamic Obstacles

Figures 14(a), (b), (c) and (d) show the planned paths that result from the proposed method when dynamic obstacles must be avoided. Any obstacle in front of the robot affects the robot's movement. Because dynamic object recognition is not implemented in our robot, all obstacles are regarded as static objects. Despite this, the proposed method is capable of avoiding dynamic obstacles because its path planning and SLAM processes run at high frequencies (see Table 4) and because the robot's velocity is determined based on its distance from the nearest obstacle. Furthermore, paths can be planned based on the latest environmental information

Table 3. Travelling distances and standard deviations of the rotation angles

| | standard deviation | travelling distance |
|---|---|---|
| conventional method | 1.6745 rad | 232.49 m |
| proposed method | 1.5529 rad | 228.76 m |

Table 4. Average computation times

| | average computation time |
|---|---|
| SLAM | 34.1 ms |
| path planning | 19.2 ms |
| velocity command | 100 ms |

available every 19.2 ms on average. In these ways, the robot is able to plan a path that avoids dynamic obstacles and moves accordingly without collision.

## 6. Conclusion

This paper proposes a method of path planning that can be implemented in an autonomous mobile robot using the A* algorithm with a new cost function that considers the robot's head angle and with use of additional adjacent nodes. Experiments have confirmed that the proposed method results in smoother movements than does the conventional method; it has been further confirmed that the proposed method is capable of planning routes that account for dynamic obstacles

and, thus, avoiding collisions with these obstacles.

Future work will involve experiments conducted in a variety of different environments. Since the experiments conducted here took place in a relatively large space with only one obstacle, future experiments will take place in narrower environments with multiple dynamic obstacles.

## References


( 1 ) T. Sakai, H. Nakajima, D. Nishimura, and H. Uematsu: "Autonomous Mobile Robot System for Delivery in Hospital", Industrial Technology, pp.62–67 (2005)

( 2 ) K. Nonami, R.K. Barai, A. Irawan, and M.R. Daud: "Designed and optimization of Hydraulically Actuated Hexapod Robot COMET-IV", Science and Engineering, Vol.66, pp.41–84 (2014)

( 3 ) M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit: "Fast- SLAM 2.0: An Improved Particle Filtering Algorithm for Simultaneous Localization and Mapping that Provably Con- verges", Proc. of the Sixteenth International Joint Conference on Artificial Intelligence, pp.1151–1156 (2003)

( 4 ) K.-H. Lin and C.-C. Wang: "Stereo-based simultaneous localization, mapping and moving object tracking", IEEE/RSJ International Conference on Intelligent Robots and System, pp.3975–3980 (2010)

( 5 ) S. Kagami: "3D Online SLAM from Stereo Camera Image Sequence", Digital Human Symposium, pp.310–313 (2009)

( 6 ) P.J. Besl and N.D. McKay: "A Method for Registration of 3-D Shapes", *IEEE Trans. on Pattern Analysis and Machine Intelligence*, Vol.14, No.2, pp.239–256 (1992)

( 7 ) K. Okada and Y. Fujimoto: "Grid-based Localization and Mapping Method without Odometry Information", Proc. IEEE Industrial Electronics Society Annual Conference, pp.159–164 (2011)

( 8 ) Y. Nakamura and Y. Fujimoto: "Validation of SLAM without odometry in outdoor environment", IEEE International Workshop on Advanced Motion Control, pp.278–283 (2014)

( 9 ) B. Park, J. Choi, and W.K. Chung: "An Efficient Mobile Robot Path Planning Using Hierarchical Roadmap Representation in Indoor Environment", IEEE International Conference on Robot and Automation, pp.180–186 (2012)

(10) P. Hart and N. Nillson: "A Formal Basis for the Heuristic Determination of Minimum Cost Paths", IEEE Transactions of Systems Science and Cybernetics, Vol.SSC-4, No.2, pp.100–107 (1968)

(11) A. Stentz: "Optimal and Efficient Path Planning for Partially-Known Environments", Proc. of the International Conference on Robotics and Automation, pp.3310–3317 (1994)

(12) F. Duchoňa, A. Babineca, M. Kajana, P. Beňoa, M. Floreka, T. Ficoa, and L. Jurišicaa: "Path Planning with Modified a Star Algorithm for a Mobile Robot", Procedia Engineering, Vol.96, pp.59–69 (2014)

(13) S. Koening, C. Tovey, and W. Halliburton: "Greedy Mapping of Terrain", Proceedings of the International Confference on Robotics and Automation, pp.2594–3599 (2001)

(14) A. Stentz and M. Hebert: "A complete navigation system for goal acquisition in unknown environments", Autonomous Robots, Vol.2, No.2, pp.127–145 (1995)



**Yudai Hasegawa** (Non-member) received the B.E. degree in electrical and computer engineering from Yokohama National University, Yokohama, Japan, in 2014. Since 2014, he has been with Yokohama National University where he is currently working toward the M.E. degree in the Department of Electrical and Computer Engineering.

**Yasutaka Fujimoto** (Senior Member) received B.E., M.E., and Ph.D. degrees in electrical anf computer engineering from Yokohama National University, Yokohama, Japan, in 1993, 1995, and 1998, respctively. In 1998 he joined the Department of Electrical Engineering Keio University, Yokohama, Japan as a research associate. Since 1999, he has been with the Department of Electrical and Computer Engineering, Yokohama National University, Japan, where he is currently a professor. He is a senior member of IEEE, and a member of Robotics Society of Japan, SICE, JSAE, IEICE, and INFORMS.