2010 IEEE International Conference on Robotics and Automation
Anchorage Convention District
May 3-8, 2010, Anchorage, Alaska, USA

Variable-resolution Map Building and Real-time Path Planning of Omni-directional Mobile Robots

Jingyu Xiang, Yuichi Tazaki, Shinkichi Inagaki and Tatsuya Suzuki

Abstract— **This research addresses a method for mobile robots that simultaneously performs map building and path-planning on line. A graph representation of a workspace with variable resolutions is constructed using measurement data obtained by omni-directional distance sensors. At the same time, a real-time search for a feasible path to the goal is executed on the constructed graph-map. The proposed method is evaluated through experiments using an omni-directional mobile robot equipped with laser range finders.**

I. Introduction

In path-planning of mobile robots, precise self-localization and map-building of the surrounding environment is of crucial importance. Off-line approach, which generates and stores these information a priori, is not applicable in dynamically changing environments. The framework of Simultaneous Localization And Mapping (SLAM) was proposed by Dissanayake [1]. In SLAM, a mobile robot (or a troop of mobile robots) executes self-localization and map-building simultaneously on line. Until today, there have been extensive studies on this topic.

One issue that has been somewhat overlooked in SLAM is path-planning. In most SLAM-related works, a mobile robot is either tele-operated or merely traces a prescribed route. When a mobile robot is required to reach a goal autonomously in a purely unknown environment, it has to not only execute SLAM but also plan a proper path to the goal using an incomplete map. Efficient path-planning is also important even when map-building is the robot's primary task. Motivated by this background, apart from SLAM, this work focuses on simultaneous execution of map-building and path-planning.

In online map-building, a map representation that can be updated incrementally based on sensor information is needed. Existing methods for the representation and creation of maps are mainly categorized into partitioning-based methods and roadmap-based methods. Partitioning-based methods ([3][4][5]) subdivides the workspace into cells, often in a hierarchical manner, and marks each cell either collision-free or occupied. On the other hand, roadmap methods constructs a graph structure in the collision-free region of the workspace[6]. Among various roadmap methods, the Probabilistic Roadmaps method (PRM) [7] has been popular in recent years. Roadmap methods are suitable for incremental-map building, because maps can be expanded simply by adding new nodes and connecting them by links with the existing map. However, most existing techniques seem to have a common shortcoming that it requires some geometric representation of the workspace a priori.

All authors are with the Department of Mechanical Science and Engineering, Nagoya University, Nagoya, Japan. { k_kou, tazaki, inagaki, t_suzuki }@nuem.nagoya-u.ac.jp

In online path planning, the robot should determine what action to make next in a limited amount of time. Path-planning methods are categorized into two groups: local planning and global planning. Local planning methods such as potential-field-based techniques [2] execute planning purely based on local information and thus does not require a map. As a drawback, there is a fear of being caught in a dead-end. In contrast, global planning methods make full use of map information to obtain an optimal path. However, most of them give no performance guarantee when search time is limited.

This research addresses a problem in which a mobile robot tries to reach a goal in an unknown but bounded 2D workspace with obstacles. For this problem, we propose a map-building method that constructs a variable-resolution roadmap based on omni-directional distance sensor information. Generated roadmaps have variable-resolution in the sense that graph-nodes are densely distributed in the neighborhood of obstacles, while minimum number of nodes are used to express free-spaces. This reflects the fact, from a perspective of safety, that precise movement is required near obstacles. It also makes sense from the viewpoint of sensing; that is, distance sensors have an inherent variable-resolution characteristic. If the sensor samples in all angles with a uniform spacing, in the Cartesian coordinate, close regions are sampled with high resolutions and far regions are sampled with low resolutions. Moreover, the method consists of a small set of rules, and therefore is quite reasonable for real-time application. In addition, it requires no a priori information about the geometry of the workspace.

For path-planning, we employ the Real-time A* (RTA*) search method ([12]). The RTA* search can produce a solution under a limit computation time by adjusting the search depth. At the same time, thanks to its special cost update rule, it prevents the agent from being caught in a dead-lock or a live-lock.

The proposed method has been implemented in a omni-directional mobile robot equipped with Laser Range Finder(LRF)s and has been tested in indoor experiments.

The rest of this paper is organized as follows: The problem of online map building and path-planning will be described in Section II. In Section III, the entire workflow of the method will be shown. In Section IV, the map-building rules will be explained in detail. The algorithm of the RTA* search will be briefly reviewed in Section V. In Section VI, we will show

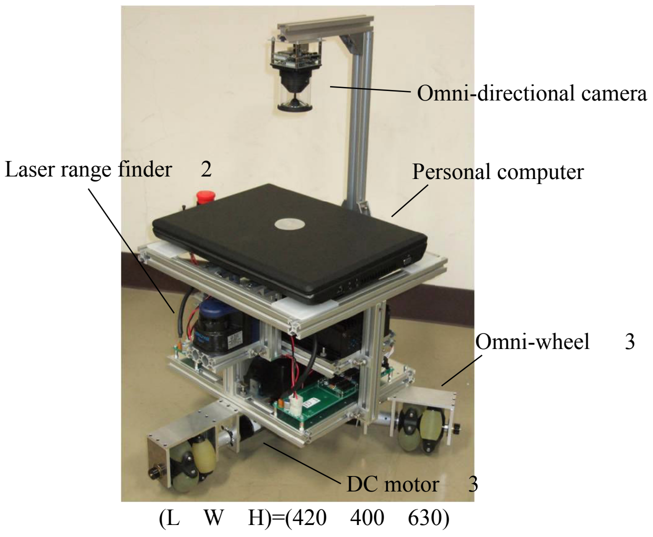


Fig. 1. Mobile robot for experiment

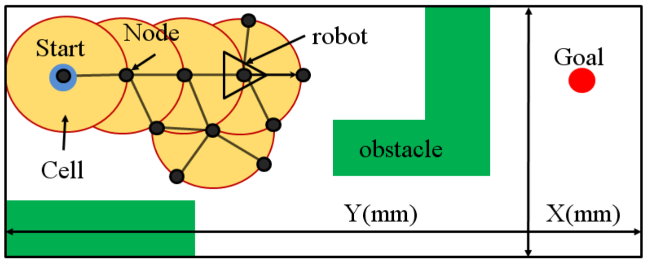


Fig. 2. Workspace of the robot

the results of indoor experiments using an omni-directional mobile robot. Concluding remarks will be made in Section VII.

II. Problem Setting

First of all, we explain the details of the mobile robot used in this research. Fig. 1 shows the mobile robot **Omnia**. The robot propels with three omni-directional wheels, each of them driven by a DC motor. The nominal speed of the robot is 500mm/s. The robot is equipped with two laser range finder(LRF)s, one in front and one in the back, and one omni-directional camera attached to the center.

Fig. 2 shows an example of a workspace, in which the robot is required to move to a specified goal position in a fully autonomous manner. We assume that the workspace is bounded and can be split into obstacles and free-space. The robot can freely move around in free-space, while it cannot enter inside obstacles. Moreover, the robot can measure the relative distance and direction to the goal using its omni-directional camera. Initially, the robot has no information about the obstacle configuration. But it can measure the distance towards surrounding obstacles in any direction using its LRFs.

III. Brief Overview of the Method

As mentioned earlier, the proposed method utilizes the RTA* search method for path-planning. RTA* is a type of graph-search techniques with real-time property. In order to apply graph-search techniques including RTA*, the robot constructs a roadmap, a graph-based representation of the free-space expressed in the so-called configuration space (C-space), based on LRF measurements. The C-space is a space of all possible states (positions and orientations) of the robot. If a point in the C-space corresponds to a configuration of the robot in which the robot's body intersects with an obstacle in the workspace, we say the point is a part of *configuration obstacles*. Otherwise, the point is in the *configuration free-space*.

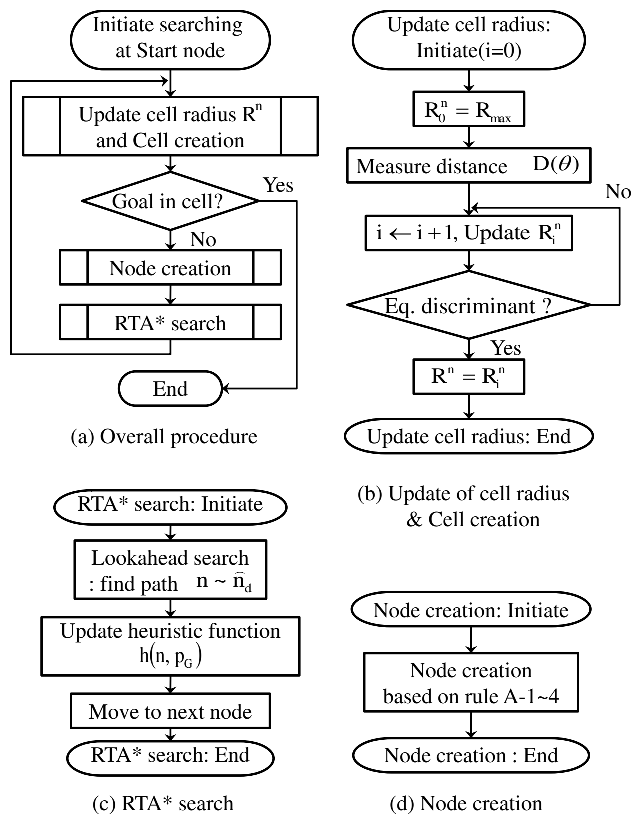
Normally, the configuration of a mobile robot in 2D workspace is expressed by (x, y, θ); x-coordinate, y-coordinate and orientation. But in the case of omni-directional mobile robots that can move and sense in all directions, we can omit θ. In this case, a graph-map constructed in the C-space can be seen as a map for the physical workspace itself.

All discussions hereafter will be made in the C-space and configuration obstacles (free-space) may be simply called obstacles (free-space). We denote the start position by $p_{\rm S} = (x_{\rm S}, y_{\rm S})$ and the goal position by $p_{\rm G} = (x_{\rm G}, y_{\rm G})$. A graph is a collection of *nodes* and *links*. Each node represents a point in the C-space and each link represents a connection between two nodes. If a pair of nodes are connected by a link, then the robot can move from one node to the other along a straight line without colliding with obstacles. Moreover, each node is accompanied with a *cell*. A cell is a region composed of points within a certain distance from the corresponding node but not inside a (configuration) obstacle. The maximum distance from the node to a point inside the cell is called the radius of the cell.

The work flow of the proposed method is shown in Fig. 3(a). At the beginning, a node is created in the starting position, where the robot is initially located. Next, a cell is created and attached to it. Here, the radius of the cell is determined by a rule described in Section IV-A. If the goal is included in that cell, the process terminates. If not, one or more new nodes are created on the boundary of the cell of current node, according to a set of rules described in Section IV-B. After that, path-planning is executed by the RTA* search over the graph constructed so far. The robot then starts moving towards the adjacent node on the obtained path. After the robot successfully arrives at the next node, the whole process is repeated until the robot eventually reaches the goal. The robot travels between two nodes in a constant period, regardless of the distance between them. As a result, the robots moves slowly when nodes are densely distributed around it, and moves rapidly in regions where a relatively small number of nodes are placed.

IV. Graph Construction Procedure

In this section, we will explain the graph-map construction method in detail. Although the method is best suited to



Initiate searching at Start node
Update cell radius R^n and Cell creation
Goal in cell?
Yes
No
Node creation
RTA* search
End
(a) Overall procedure
Update cell radius: Initiate(i=0)
$R_0^n = R_{max}$
Measure distance $D(\theta)$
No
$i \leftarrow i+1$, Update R_i^n
Eq. discriminant ?
Yes
$R^n = R_i^n$
Update cell radius: End
(b) Update of cell radius & Cell creation
RTA* search: Initiate
Lookahead search : find path $n \sim \hat{n}_d$
Update heuristic function $h(n, p_G)$
Move to next node
RTA* search: End
(c) RTA* search
Node creation: Initiate
Node creation based on rule A-1~4
Node creation : End
(d) Node creation

Fig. 3. Flowchart of overall procedure

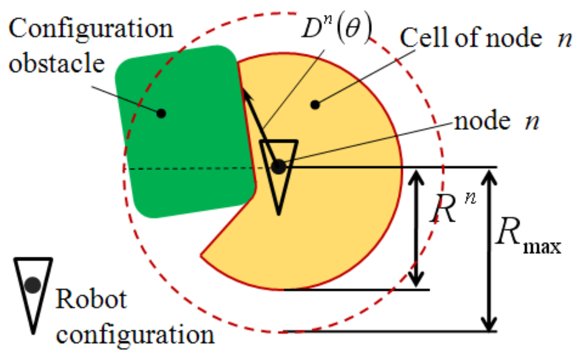


Fig. 4. Measured distance $D^n(\theta)$ and cell radius R^n

mobile robots equipped with full-range distance sensors (most typically, LRFs), it is still applicable to general types of robots as long as they are capable of obtain distance information by other means (e.g., stereo cameras).

A. Variable-resolution Cell Decomposition

As illustrated in Fig. 4, we denote the maximum measurable distance by $R_{\max}$. For a node n, the position of the node n is denoted by p^n and the radius of the cell attached to the node n is denoted by R^n. Moreover, we express the relative angle of a point with respect to the robot's orientation by θ. The function $D^n(\theta)$ returns the distance to an obstacle from the node n in the direction θ. Using this notation the cell of node n, denoted by C^n, is formally defined as a set in the following equation:

$$C^n = \{p \mid \| p - p^n \| \leq \min(D^n(\arg(p - p^n)), R^n)\}. \quad (1)$$

Here, $\arg(v)$ returns the direction of the vector v.Since the node position p^n is fixed when the node is created and $D^n(\theta)$ is obtained by sensor measurements, C^n is determined when the remaining parameter R^n, the cell radius, is specified. The cell radius R^n is chosen to satisfy the following criterion:

$$R^n - \frac{1}{2\pi}\int_0^{2\pi} \min(D^n(\theta), R^n)d\theta \leq R_{\text{th}}, \quad (2)$$

which means that the difference between R^n and the average of $\min(D^n(\theta), R^n)$ is below the threshold R_{th}. The parameter R_{th} should be specified a priori. If R_{th} is set as 0, R^n is given as

$$R^n = \min(D^n(\theta)) \quad (3)$$

and therefore is equivalent to the distance to the nearest obstacle. The intention of this criterion is to vary the cell radius according to the distance towards the nearest obstacle. In addition, cell radius should be chosen so that the configuration of surrounding obstacles can be captured from the shape of the cell boundary. As we can see in Fig. 4, a cell boundary is composed of arcs that indicate no obstacle is detected in the corresponding directions, and portions of obstacle boundaries measured by the distance sensor. Since the inequality (2) is difficult to solve directly, we calculate R^n by an iterative procedure explained below. First, we approximate the second term in the left-hand-side of (2) as follows:

$$\frac{1}{2\pi}\int_0^{2\pi} \min(D^n(\theta), R^n)d\theta \simeq \frac{\sum_{k=0}^{m-1} \min(D^n(k \times \Delta\theta), R^n)}{m} \quad (4)$$

Here, m denotes the number of discrete angles and $\Delta\theta$ denotes the spacing width. Given $\Delta\theta$, m is given by $m = [2\pi/\Delta\theta]$ where $[*]$ denotes the largest integer not exceeding $*$. Using this approximation, (2) is transformed into

$$R^n - \frac{\sum_{k=0}^{m-1} \min(D^n(k \times \Delta\theta), R^n)}{m} \leq R_{\text{th}}. \quad (5)$$

Using (5), we calculate R^n in the following three steps:

STEP1 (Initialization): Set R_0^n as

$$R_0^n = R_{\max}. \quad (6)$$

STEP2 (Update): Given R_{i-1}^n, calculate R_i^n by

$$R_i^n = \frac{\sum_{k=0}^{m-1} \min(D^n(k \times \Delta\theta), R_{i-1}^n)}{m}. \quad (7)$$

STEP3 (Terminate): If

$$R_i^n - \frac{\sum_{k=0}^{m-1} \min(D^n(k \times \Delta\theta), R_i^n)}{m} \leq R_{\text{th}} \quad (8)$$

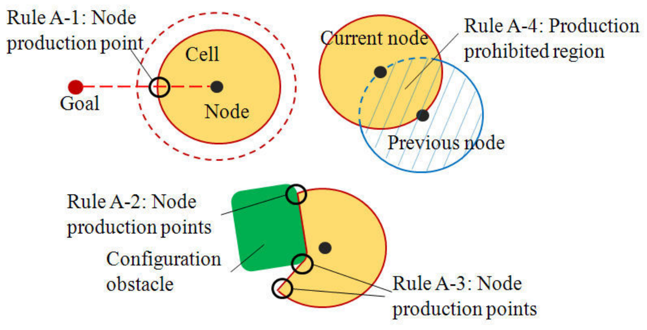

Rule A-1: Node production point
Cell
Goal
Node
Current node
Rule A-4: Production prohibited region
Previous node
Rule A-2: Node production points
Configuration obstacle
Rule A-3: Node production points

Fig. 5. Node production rules

holds, terminate with $R^n = R_i^n$. Otherwise set $i \leftarrow i+1$ and go to STEP2.

Since

$$\min(D^n(0), D^n(\Delta\theta), ..., D^n((m-1)\times\Delta\theta)) \leq R_i^n \leq R_{i-1}^n \tag{9}$$

holds, R_i^n decreases monotonically with respect to i. Therefore, we can obtain R_i^n satisfying (8) in finite iterations. The procedure terminates also when R_i^n goes below $R_{\min}$, the minimum cell radius.

B. Node Creation Rules

After a cell is created at the current position, new nodes are created on the boundary of the current cell, based on a set of rules listed below:

A–1 Create a node on the intersection point of the cell boundary and a straight line connecting the current position and the goal, unless the intersection point is inside an obstacle.

A–2 Create nodes on the cell boundary where two different kinds of line segments intersect; an arc indicating free-space and an obstacle boundary.

A–3 For each dead-angle, create nodes on both end-points of the straight line-segment of the cell boundary.

A–4 Do not create a node inside the intersection of the cell and other existing cells.

Notice that only rule A-4 is an inhibition rule. Therefore, a node is created on a point at which either rule A-1, A-2 or A-3 is applied and rule A-4 does not.

Each newly created node is then connected by a link with the current node. Based on the rules explained above, the robot incrementally explores and builds a graph-map of the workspace.

C. Features of the proposed method

Some map-representations with variable level-of-details already exist in the literature ([8][9]). However, these methods require a priori knowledge of the geometry of the workspace. On the other hand, the proposed method determines the resolution of map representation based purely on sensor information. The Generalized Voronoi Graph (GVG) [10] is also known as a sensor-based mapping method. However, it is reported in [11] that GVG is sensitive to sensor noise because it directly uses the range data. The Thinning-based Topological Map (TTM) [11] improves the robustness of GVG against noise, but it fails when there is no obstacle within the sensing range. On the other hand, the proposed method does not suffer from this problem because it creates nodes on the cell boundary. Moreover, the measurement noise of LRF is proportional to the measured distance. Thus, we can reduce the effect of sensor noise by setting R_{max}, the maximum cell radius, appropriately small.

V. Path-planning Based on RTA* Search

There are mainly the following requirements in the path-planning of a single mobile robot:

1) Limited search-depth; the robot cannot search the workspace in an arbitrary depth.
2) Limited search-time; the robot should decide its next action in a limited amount of time.

Most graph-search methods (e.g., depth-first, breadth-first, A* and so forth) are off-line techniques; they guarantee to output an optimal path if the search-time is unlimited but there is no performance guarantee if the search-time is limited. On the other hand, the Real-time A* (RTA*) search method proposed by Korf [12] is a real-time method. Its computation time can be adjusted by changing the search-depth. Thanks to its unique cost-update rule, it still guarantees that the agent (the mobile robot) will never be caught in an infinite loop, and therefore will eventually reach the goal. The RTA* search consists of two phases: the search phase and the execution phase (Fig. 3(c)).

(1) Search phase

Let n be the current node and let d be the search depth. Moreover, let $\{n_d\}$ be a set of nodes that are reachable from the node n in exactly d steps. Find a path $(n, \ldots, \hat{n}_d)$ $(\hat{n}_d \in \{n_d\})$ that minimizes the cost function

$$f(n, n_d) = g(n, n_d) + h(n_d, p_G) \tag{10}$$

(Fig. 6). In (10), $g(n, n_d)$ denotes the cost of the path from n to n_d. On the other hand, the function $h(n_d, p_G)$ serves as a heuristic estimate of the cost from the node n_d to the goal p_G, similar to those used in the ordinary A* search. The cost estimate $h(n_d, p_G)$ is initialized with the Euclidean distance between $n_d = [x_d, y_d]^T$ and $p_G = [x_G, y_G]^T$:

$$h(n_d, p_G) = \sqrt{(x_G - x_d)^2 + (y_G - y_d)^2}. \tag{11}$$

and it is updated at the execution phase.

(2) Execution phase

The robot moves to an adjacent node on a path that minimizes

$$f(n, \hat{n}_d) = \min_{\hat{n}_d \in \{n_d\}} f(n, \hat{n}_d). \tag{12}$$

If there exists more than one path with the same cost, one node is chosen at random. Before moving, the heuristic cost

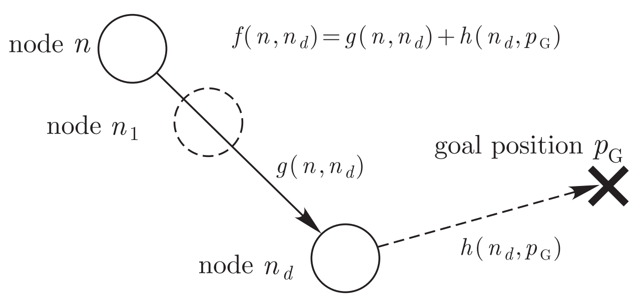

node n
$f(n,n_d)=g(n,n_d)+h(n_d,p_{\mathrm{G}})$
node n_1
$g(n,n_d)$
goal position p_{G}
node n_d
$h(n_d,p_{\mathrm{G}})$

Fig. 6. Cost function of RTA* search

of the node n, $h(n, p_{\mathrm{G}})$, is update according to the following rule:

$$h(n,p_{\mathrm{G}}) = \min_{n'_d \in \{n_d\}\backslash\{\hat{n}_d\}} f(n,n'_d) \tag{13}$$

This update rule replaces the heuristic cost of n with the second minimum path cost. As an effect of this update rule, the heuristic cost of each node monotonically increases each time the robot arrives at that node. Consequently, the robot will get out of a loop after a finite number of cycles and start exploring other nodes.

VI. Experiments

In this section, experimental results using the omni-directional mobile robot Omnia (Fig. 1) are presented. The proposed algorithm needs distance information to the nearby obstacles measured from the center of the robot. Actual measurement data, however, are distance from the center of LRFs. To transform the measurement data to body-centered polar coordinate, the data are first converted to the Cartesian coordinate by

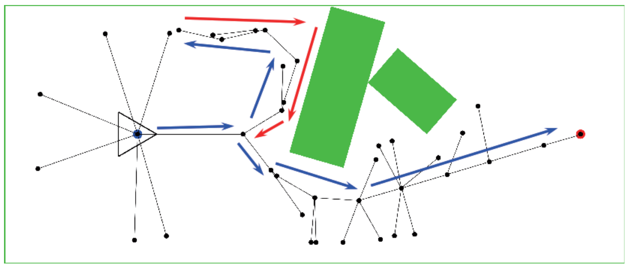
$$\begin{bmatrix} X_R \\ Y_R \end{bmatrix} = \begin{bmatrix} X_L \\ Y_L \end{bmatrix} + \begin{bmatrix} 0 \\ d_{\mathrm{lr}} \end{bmatrix} \tag{14}$$

and then it is converted to the body-centered polar coordinate. Here, (X_R, Y_R) denotes the body-centered Cartesian coordinate and (X_L, Y_L) denotes the LRF-centered Cartesian coordinate. The constant d_{lr} is the offset between the body center and each LRF.

Self-localization is done by dead-reckoning using encoders attached to the wheels. Moreover, an LED landmark is placed at the goal. The robot can detect this landmark using its omni-directional camera to know the direction to the goal from its current position. This information can be utilized to compensate severe drifts caused by the dead-reckoning. The LED landmark is placed in an altitude high enough so that it will always be in the robot's sight.

In the following experiments, the step-size $\Delta\theta$ used for the cell-radius calculation is set as $0.36°$, the cut-off threshold R_{th} is set as 20mm, the minimum cell radius $R_{\min}$ is set as 400mm and the maximum cell radius $R_{\max}$ is 2000mm.

The first result highlights the effect of node creation rules and that of RTA* search. It also demonstrates that the map can be reused in different trials. The workspace for this experiment is shown in Fig. 7. The workspace is surrounded by walls so that the robot will never go out of it (walls are detected by the robot as obstacles). A triangle in each figure depicts the initial position and direction of the robot. Note that all graphs and trajectories in Fig.7 and Fig.8 are drawn using actual measurement data. In the first trial (Fig. 7(a)), the robot creates a map while it moves towards the goal. The search depth of the RTA* search d is set as 1, Black dots depict the nodes of the graph-map and solid lines depict the links. The movement trajectory of the robot is shown by arrows. Since the robot starts with no information about the obstacle configuration, it first chooses left (up w.r.t. the paper) by chance, which leads to a dead-end. After several steps, it detects the dead-end and then comes back to the node where it first turned left. Then it tries the other direction and this time it successfully reaches the goal. The dead-end detection and avoidance is accomplished by the following mechanism: When the robot reaches a dead-end, the creation of new nodes will be suppressed after a certain point of time. This is the effect of Rule A-4, which inhibits the creation of new nodes inside existing cells. Then, as an effect of the cost update rule of RTA*, the cost of nodes near the dead-end will increase monotonically. As a result, the robot will be pushed away from the dead-end.


(a) First trial

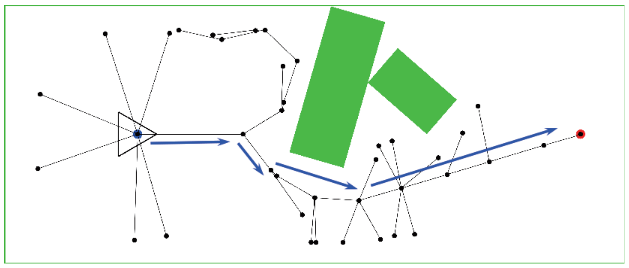
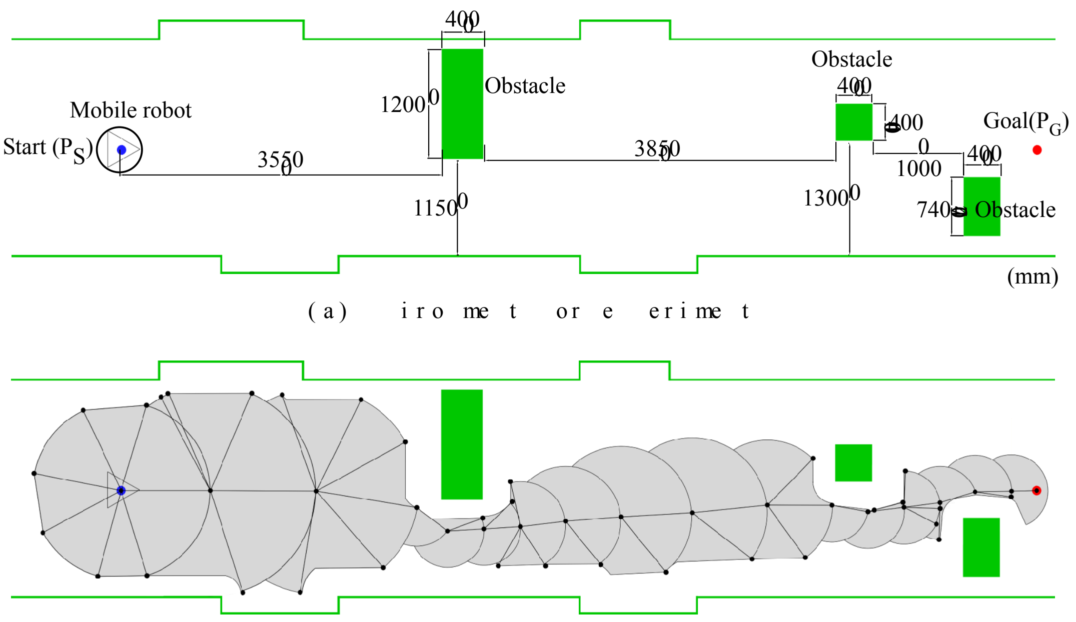

(b) Second trial

Fig. 7. Environment for experiment and Graph-map constructed by the mobile robot

In the second trial, the robot starts from the same initial position but this time it already has a map information. Moreover, the search depth of RTA* is set as 5. As a result, the robot chooses the optimal path, which leads to the goal while avoiding the dead-end. Thus, it has been shown that a map can be stored and reused to facilitate efficient planning for different trials.

The next experiment tests the variable-resolution feature of the proposed map building method, in a larger-scale environment (Fig. 8). A graph-map created by the robot



(a)   i r o me t   or e   e r i me t

(b) Gr a   ma   co s t r c t e   b   mo b i l e r o b o t

Fig. 8. Environment for experiment and Graph-map constructed by the mobile robot

after a trial is shown in Fig. 8(b). The cells are depicted by gray regions. Thanks to the variable-resolution feature of the method, we can observe that large cells are created during a few steps after the robot has started off and when the robot travels through a free-space between the first and second obstacles. In particular, the cell radius of the second node is 1300mm. In contrast, when the robot goes through an obstacle and a wall, cells are created with smaller radius or sometimes with the minimum radius (400mm). As a result, the robot realizes precise and safe motion near obstacles while it makes more agile movement when obstacles are far enough.

VII. Conclusion

In this paper, a method for simultaneous map-building and path-planning for mobile robots has been presented, and its has been tested through experiments. The proposed method enables the construction of a graph-map with variable resolutions based on distance sensor measurements by simply applying a small set of rules. Future work includes the extension of rules for map construction, especially rules for creating links between nodes. Extension of link creation rules will enable creating loops, connecting several existing maps together, cope with changing environments, and more. Other important issues are to provide a theoretical guarantee that the robot will eventually reach the goal, to incorporate self-localization, to consider other possible criteria for deciding the cell radius, and to compare the proposed method with different methods through both simulations and real experiments.

References

[1] M.W.G.Dissanayake, P.Newman, S.Clark, H.F.Durrant-Whyte and M.Csorba: "A Solution to the Simultaneous Localization and Map Building(SLAM) Problem", IEEE Trans. on Robotics and Automation, Vol. 17, No. 3, pp. 229-241, June, 2001.

[2] O.Khatib: "Real-time obstacle avoidance for manipulators and mobile robots", International Journal of Robotic Research, Vol. 5, No. 1, pp. 90-98, 1986.

[3] R.Brooks and T.Lozano-Perez: "A subdivision algorithm in configuration space for findpath with rotation", Proceedings of the 8th International Conference on Artificial Intelligence, pp. 799-806, 1983.

[4] S.Behnke: "Local Multiresolution Path Planning", Robot Soccer World Cup VII, LNCS 3020, pp. 332-343, Springer, 2004.

[5] S.Kambhampati and L.S.Davis: "Multiresolution path planning for mobile robots", IEEE Journal of Robotics and Automation, Vol. 2, No. 3, pp. 135-145, 1986.

[6] C.O'Dunlaing and C.K.Yap: "A 'retraction' method for planning the motion of a disk", J. Algorithms, No. 6, pp. 104-111, 1985.

[7] L.E.Kavraki, P.Svestka, J.C.Latombe, M.H.Overmars: "Probabilistic Roadmaps for Path Planning in High-Dimensional Configuration Spaces", IEEE Trans. Robotics and Automation, Vol. 12, No. 4, pp. 566-580, 1996.

[8] H.Noborio, T.Naniwa, S.Arimoto: "A Quadtree-Based Path-Planning Algorithm for a Mobile Robot", Journal of Robotics Systems, Vol.7, No.4, pp.555-574, August 1990.

[9] G.K.Kraetzschmar, G.P.Gassull, K.Uhl: "Probabilistic Quadtrees for Variable-Resolution Mapping of Large Environments", Proceedings of the 5th IFAC/EURON Symposium on Intelligent Autonomous Vehicles, 2004.

[10] H.Choset, J.Burdick: "Sensor-Based Exploration:The Hierarchical Generalized Voronoi Graph", IJRR, Vol.19, No.2, pp.96-125, February 1, 2000.

[11] T.B.Kwon, J.B.Song: "Real-time building of a thinning-based topological map", Intelligent Service Robotics, Vol.1, No.3, pp211-220, 2008.

[12] R.Korf: "Real-Time Heuristic Search", Artificial Intelligence, Vol. 42, No. 2–3, pp. 189– 211, 1990, March.