UNSYS digital

Journal of Unmanned System Technology

Path Planing for Mobile Robot With Cellular Automata

J. Santoso, B. Riyanto Trilaksono, and O. Setiono Santoso
School of Electrical Engingeering and Informatics, Bandung Institute of Technology
Bandung, Indonesia

***Abstract*—**In this paper we propose a new approach, the robot path planning with cellular automata. The idea is based on maximum clearence technique that preserves the distance of the robot to obstacles as far as possible. An existing approach is implemented using voronoi diagram that generates the candidate paths that are safe from collision with the obstacles. In fact, maximum clearence method can be solved analitically using the deformation retraction, but this approach is applicable for the continuous environment only, and it requires a lot of function computation. Hence, we solve this problem using a particular rule of cellular automata to perform the process of computation that can be done efficiently. Our approach is suitable for path planning in a grid-based environtment.

***Keywords*—**robot path planning; maximum clearance; cellular automata.

I. INTRODUCTION

Building an optimal algorithm for a mobile robot is a challenging task. Many approaches have been studied and developed by many researchers either from academia or industry to find the solutions, eventhough it is not optimal. Initially, the robot path planning was studied and applied to a simple problem which is known as Piano Mover. The algorithm is used to move the piano in the room from one place to another one without making collision with the obstacles. This simple algorithm does not consider the dynamic properties and the differential constraint, and it is merely doing translation and rotation.

Some improvements have been developed to consider many factors such as the factor above and other factors like uncertainties, errors in modeling and optimality algorithm. Among those approaches, there is one common goal, i.e. finding the best and optimal trajectory as a result of the algorithm. Some algorithms use discrete and probabilistic approach [1] that is integrated with the basic theory of searching and dynamic programming with some minor changes [2], [3]. In this paper, we use the discrete approach to design the robot path planning with cellular automata.

Cellular automata is a discrete model that has various structures and characteristics according to its given rules [4], [5], [6], [7]. Cellular automata consists of several cells, one cell as a center and the remaining ones as neighborhoods. Each cell has a state that is represented by 0 (dead) or 1 (alive) for two states cellular automata. By exploiting the characteristic of rules, the process of updating the states can be done efficiently with a simple calculation.

Robot path planning is complex in nature due to several factors that must be considered, i.e. kinematics and dynamics properties of the robot, the model of environment where the robot is moving (world), and other existing constraints. Because of those factors the optimal planning can not be solved easily, but by simplifying the model and computation good solution can be achieved.

This paper is structured as follows: in Section II, we discuss the related works; in Section III, we review the theory of cellular automata; in Section IV, we discuss the search strategies; in Section V, we integrate and implement our search strategies with the cellular automata, followed by Section VI showing simulation results; in the last section, we conclude our work.

II. RELATED WORKS

Robot path planning with cellular automata was used by Behring [8] to compute distance from any position to goal position in a grid environment. The distance was computed from the goal position incrementally using a simple rule of two dimensional cellular automata. In this case, minimum distance was not considered yet. In other approach, the distance was computed in two global schemes [9]. In the first scheme, the distance at any point to the goal position was computed with cellular automata. Whereas in the second scheme, optimal direction to the goal position from any position is determined and depicted in a map.

Robot path planning with non-holonomic properties moves along trajectories following the minimum on valley of a potential hypersurface [10]. Computation was implemented with cellular automata consisting five layers that were grouped in two major subsystems: input layer and output layer. The input layer had three components i.e. initial pose layer, goal pose layer, and obstacles layer. The output layer consisted of two important layers: path extraction layer and attraction layer.

Garido [11] designed the robot path with a Voronoi diagram. The diagram represented the environment with its obstacles,

Corresponding author: Bambang Riyanto (e-mail: briyanto@lskk.ee.itb.ac.id)
This paper was submitted on July 16, 2013; revised on August 18, 2013; and accepted on August 22, 2013.

and it was constructed using data from sensors. The Voronoi diagram described the maximum distance of the robot to the nearest obstacles. The optimal path was determined by the fast marching method along the Voronoi diagram. The path planning that is free from collision was done by Ioannidis [12]. Several robots made a formation and moved together to the goal position while maintaining their formation. The algorithm was implemented with cellular automata.

Mansor [13] designed a path planning with virtual window. The algorithm was computed using information inside the window. Some obstacles were modeled as rectangles with arbitrary size. Path planning using dynamic programming was presented by Chien [14] to find minimum translation. Robot path planning using a modified wave front was proposed by Pal [15]. Expansion of the wave was limited to reduce cost of exploration.

Preliminary results of this work appeared in [16]. In comparison with the works of [8], [9], [10], [11], [13], and [15], the present paper differs in the following aspects. First, we used cellular automata to generate candidate paths by contracting the domain covered by the sensor. Secondly, minimum distance to the goal was computed at the beginning when the global path was determined; however, it needed to be corrected at the next step when local search was applied.

III. Cellular Automata

Cellular Automata (CA) is a discrete model consisting of interconnected cells, and each cell has various states. The state of each cell is determined by the current state of its neighbors including its own state. In one dimensional CA, there are three states neighbors, whereas in two dimensional CA there are nine states. During evolution, a CA will change its states and generate a complex structure with simple arithmetical computations.

CA cell has its own state that can change to another state according to a particular rule. Next state is expressed as a function of its neighbours.

$$q_i(t+1) = f(q_{i-1}(t), q_i(t), q_{i+1}(t)) \qquad (1)$$

$q_i(t+1)$ is the state of the i-th cell at the $(t+1)$-th time step where f *is* a transition function that takes the neighbors as arguments. This function is called rule of CA. The rule is represented in decimal number between 0 and 256.

The number denotes decimal value of output state. For example, rule number 60 (00111100) and 90 (01011010) are defined in **Table 1**.

Table 1 The rule number 60 and 90

input	111	110	101	100	011	010	001	000	
output	0	0	1	1	1	1	0	0	60
output	0	1	0	1	1	0	1	0	90

The neighborhood structure of two dimensional CA consists of nine cells including it itself. This structure is called Moore neighborhood as shown in **Figure 1**. The next state in this structure is determined by the state of the nine neighbors at the previous time step.

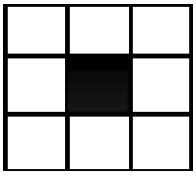

Figure 1 Moore neighborhood

Similarly, state transition of two dimensional CA is written as follows.

$$q_{i,j}(t+1) = f(q_{i-1,j-1}(t),\ldots,q_{i,j}(t),\ldots,q_{i+1,j+1}(t)) \qquad (2)$$

IV. Problem Statement

Suppose a robot $A \subset R^2$ moves in two dimensional Euclidean space R^2 from an initial position to a goal position. Initial configuration C_I contains the initial robot pose and position A(q), and the goal position is in the final configuration C_F.

The robot has a sensing ability to detect obstacles in front of it. Assume that it has no information regarding map of the environment. The problem is to construct paths that satisfy the following criteria:

- It is able to avoid collisions with the obstacles and other robots
- Final paths should be optimal or near optimal (e.g. the shortest one)

In this paper we used two stages search strategy to localize the goal. At the first stage, the goal position was determined by applying backward search. This search would generate global path and estimated distance to the goal. At the second stage, forward search was used to refine the global path due to external disturbances, e.g. dynamic obstacles, to obtain a reliable path.

A. Backward Search

The backward search was applied to find a set of previous states for a given state. Let $s \in S$ be a current state, and suppose $\theta \in \Theta(s,u)$ be a nature action. Back projection of s is defined:

$$B(s,u) = \{s' \in S \mid \exists \theta \in \Theta(s',u) \text{ such that } s = f(s',u,\theta)\} \qquad (3)$$

In general, the back projection of $V \subset S$ is given by:

$$B(V,u) = \{s' \in S \mid \exists \theta \in \Theta(s',u) \text{ such that } f(s',u,\theta) \in V\} \qquad (4)$$

Startting from $s \in G$ after k steps, the algorithms could generate possible path to an initial position if the solution exists. The global path is shown in **Figure 2**.

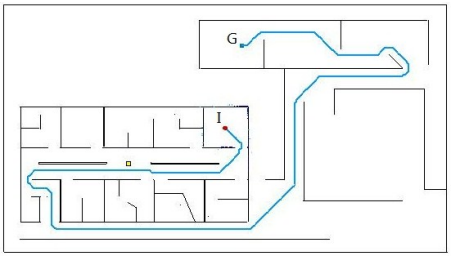

Figure 2 The global path in indoor environment.

B. Forward Search

The forward search was used to find the next states for a given state. Let $s_k \in S$ be a current state, and suppose there was uncertainty that was expressed as $\theta \in \Theta(s,u)$, where $u \in U$ is a set of actions. Forward projection after k steps was defined as follows.

$$S_{k+1}(S_k,u_k)=\{s_{k+1}\in S \mid \exists s_{k+1}\in S_k \quad \text{and} \quad \exists \theta_k \in \Theta(s_k,u_k) \text{ such that } s_{k+1}=f(s_k,u_k,\theta_k)\} \tag{5}$$

The uncertainty occurred due to dynamic properties of environment, e.g. obstacles, errors of the sensor.

V. CA Implementation in Path Planning

We implement our apporach as follows. Let the robot move from the initial point x_I to the goal point x_G. The planning was divided into two stages. First, we generated the global path that guides the robot travelling to the goal states. Minimum distance from any point to the goal states was computed using the cellular automata rules that would be used as the function of navigation for the robot.

Let N_i^t be the neighborhood of the cell i at time t of the cellular automata, and S_i^t be the state of the cell i at time t that represents the distance to goal position. The rule to update the distance was defined as follows.

$$S_i^{t+1}=\begin{cases}\min\{s_x^t\}+1 & \text{if } S_i^t = \text{free cells} \wedge \exists S_x^t\ N_i^t \wedge \\ & S_x^t \text{ is the computed cells} \\ S_i^t & \text{if } S_i^t \text{ is not the free cells}\end{cases} \tag{6}$$

In the second stage, we needed to refine the global path with local planning. The local planning was applied on region of sensor range. By using boundary sensors, the robot would generate a sensor mapping when the states of the robot were given.

Let R_{cov} be an image of the sensor mapping, and L_i^t be alive cells in the neighborhood, and $n\{L_i^t\}$ be the number of alive cells. The path was approximated by contracting the region with the following rule.

$$S_i^{t+1}=\begin{cases}1 & \text{if } S_i^t=1 \wedge (n\{L_i^t\}\geq 6) \wedge (S_i^t \in R_{cov}) \\ 0 & \text{if } S_i^t=1 \wedge (n\{L_i^t\}\leq 5) \wedge (S_i^t \in R_{cov}) \\ S_i^t & \text{otherwise}\end{cases} \tag{7}$$

VI. Simulation

To implement our approach, we used the mobile robot Pioneer 3AT simulator, MobileSim 0.5.0. The simulator was used to simulate the motion planning of an autonomous robot in two dimensional environment. In **Figure 3**, we simulated our robots pioneer 3 AT and were moving in 2D environment. It was assumed that the robot was equipped with a sensor device, SICK laser rangefinder, to detect obstacles in its surrounding. The laser could sense object up to 32 mm and read 180 data in 180 degrees.

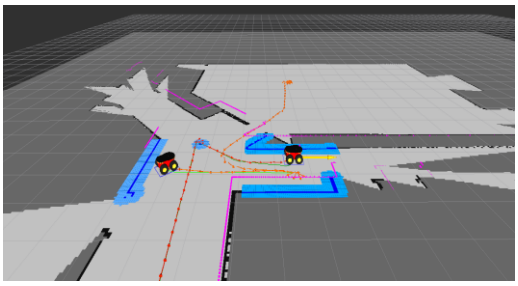

Figure 3 Pioneer 3AT in 2D environment.

The robot moved from an initial position to the goal position, as shown in **Figure 2**, in an indoor environment following the global path computed using (6). This path was used as a global reference, and it would be corrected in real time when the robot was in danger due to a new obstacle. By applying (7), the distance of the robot to the new obstacle could be computed and the new path would be generated.

The initial pose when the robot was starting to move is shown in **Figure 4**. At time t_i, the robot would arrive at a position p_i that was indicated by Pose 1 till Pose 6 as shown in **Figure 6 – Figure 11**.

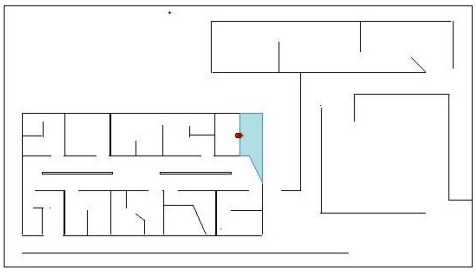

Figure 4 The robot is starting to move

A. Path Construction

The global path was constructed before the robot moved and it would be corrected continuously while the robot was travelling to adapt changes of nearby objects. The process of updating the path used the information (image) from the sensor reading in which it represented a set of cells in range. Cells in the boundary were deleted to reduce the number of states such that the number of path options was minimized.

By applying the cellular automata rules (7) after several steps, the image would be contracted and the path was extracted from the persisting image. The process of contraction was executed until there was no other alternatives of a candidate path. An example of construction process was shown in **Figure 5**. After several steps, the image was shrunk.

Figure 5 The example of image evolution

Several robot poses after 3 steps and 6 steps is shown in **Figure 6** up to **Figure 11** during the robot's movement to the destination. The first **Figure 6(a)** is the original sensor image before it was processed. After applying the rules (7), the image was shrunk and it was smaller than the original one, as shown in **Figure 6(b)**. Continuing the process, we got the image as shown in **Figure 6(c)**. One step of processing took a short time because it was done in parallel. However, it still depended on the processing unit of the onboard computer.

B. Trajectory Construction

A trajectory is the robot path from an initial position to the goal position. A set of trajectories can be generated when the robot is moving. Initially, the robot was given one global path computed using (6), with assumption that the obstacles were static. Once the robot started to move, there were several altenative paths the robot could choose. Many alternative paths appeared because the obstacles changed, but it was difficult to select the best one. One of many criteria that could be used was by collisions avoidance. Hence, distances of the robot to other objects were kept as long as possible.

The process of preserving the distance was done as follows. The robot detected the obstacles in its surrounding. The image of sensor that represented a set of state in a range would be processed by cellular automata (7) to eliminate areas that had little chance for being selected as a candidate path. The process of path construction used the most current sensor reading, and it was done continuously. The final path that was being the best trajectory is shown in **Figure 12 (b)**.

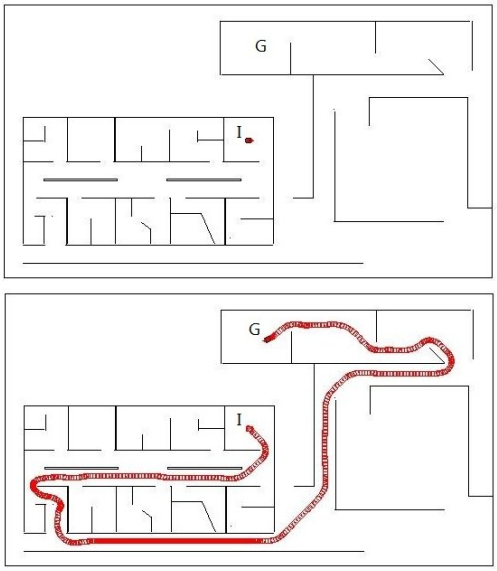

Figure 12 Trajectory construction: top (a) and bottom (b)

VII. CONCLUSION

Path planning for mobile robot should be fast and simple enough with high precision because it must be able to make right decisions in a short time, e.g. to change the moving direction. The proposed path planning with cellular automata make a worthful contribution in dealing with this problem in terms of states processing, particularly when the environment is discrete. In addition, computation process of cellular automata is suitable for being implemented on low cost hardware such as onboard computer integrated with the robot. By using cellular automata, process of updating the states can be done in parallel because of its asynchronous properties, and consequently, the time required to complete the algorithm will decrease significantly.

ACKNOWLEDGMENT

We gratefully acknowledge support for this work by the Ministry of Education of Indonesia under BPPS scholarship.

REFERENCES

[1] S. Thrun. Robotic mapping: A survey. In G. Lakemeyer and B. Nebel, editors, *Exploring Artificial Intelligence in the New Millenium*. Morgan Kaufmann, 2002. to appear.

[2] I. P. Androulakis. Dynamic programming: Stochastic shortest path problems. In *Encyclopedia of Optimization*, pages 869–873. 2009.

[3] N. T. Moungla, L. Létocart, and A. Nagih. An improving dynamic programming algorithm to solve the shortest path problem with time windows. *Electronic Notes in Discrete Mathematics*, 36:931–938, 2010. CrossRef

[4] F. Bagnoli. Cellular Automata. *ArXiv Condensed Matter e-prints*, oct 1998.

[5] M. Mitchell. Computation in cellular automata: A selected review. In *Nonstandard Computation*, pages 95–140. Wiley-VCH, 1996.

[6] J. Santoso, O. S. Santoso, and B. R. Trilaksono. Matrix characteristics for two dimensional nongroup cellular automata. In *Proceedings of the 2011 International Conference on Electrical Engineering and Informatics*, pages K2–4, Bandung, July 2011. IEEE. CrossRef

[7] S. Wolfram. A new kind of science. Wolfram Media, 2002.

[8] C. Behring, M. Bracho, M. Castro, and J. A. Moreno. An algorithm for robot path planning with cellular automata. In *ACRI*, pages 11–19, 2000.

[9] Y. Tavakoli, H. S. Javadi, and S. Adabi. A Cellular Automata Based Algorithm for Path Planning in Multi-Agent Systems with A Common Goal. *International Journal of Computer Science and Network Security*, 8(7):119–123, 2008.

[10] F. M. Marchese. A path-planner for mobile robots of generic shape with multilayered cellular automata. In *ACRI*, pages 178–189, 2002.

[11] S. Garrido, L. Moreno, M. Abderrahim, and F. M. Monar. Path planning for mobile robot navigation using voronoi diagram and fast marching. In *IROS*, pages 2376–2381, 2006.

[12] K. Ioannidis, G. C. Sirakoulis, and I. Andreadis. A path planning method based on cellular automata for cooperative robots. *Applied Artificial Intelligence*, 25(8):721–745, 2011. CrossRef

[13] M. A. Mansor and A. S. Morris. Path planning in unknown environment with obstacles using virtual window. *Journal of Intelligent and Robotic Systems*, 24(3):235–251, 1999. CrossRef

[14] T. L. Chien, H. C. Lai, Y. C. Lin, and Y. C. Lin. Dynamic programming algorithm based path planning of the multiple robot system. In *ICDMA*, pages 469–474, 2011.

[15] A. Pal, R. Tiwari, and A. Shukla. A focused wave front algorithm for mobile robot path planning. In *HAIS (1)*, pages 190–197, 2011.

[16] J. Santoso, O. S. Santoso, and B. R. Trilaksono. Dynamic robot path planning with cellular automata. In *Proceedings of the 8th International Conference on Intelligent Unmanned Systems,* ISBN: 978-981-07-4225-6, pages 179-183, Singapore, October 2012. SIM University, Research Publishing.

[17] V. Desaraju and J. P. How. Decentralized path planning for multi-agent teams with complex constraints. *Auton. Robots*, 32(4):385–403, 2012. CrossRef

[18] T. Fawcett. Data mining with cellular automata. *SIGKDD Explor. Newsl.*, 10(1):32–39, 2008. CrossRef

[19] F. A. Kolushev and A. A. Bogdanov. Multi-agent optimal path planning for mobile robots in environment with obstacles. In *Ershov Memorial Conference*, pages 503–510, 1999.

[20] S. M. LaValle. *Planning Algorithms*. Cambridge University Press, 2006. CrossRef

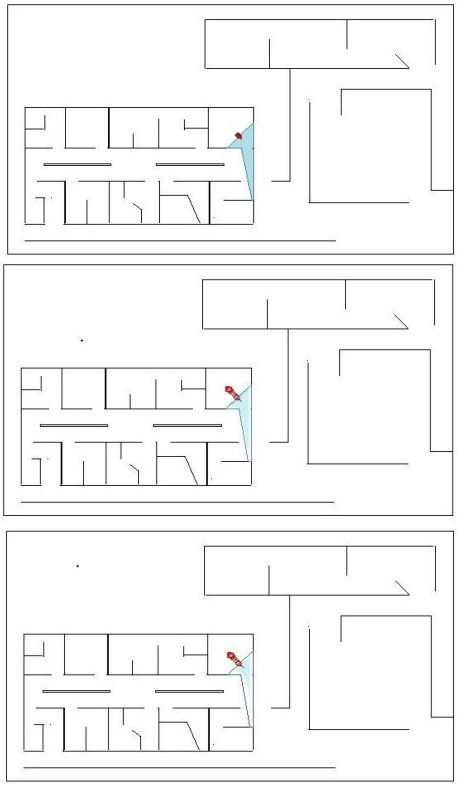

Figure 6 a. Initial pose 1 (top), b. First step (middle), c. Second step (bottom)

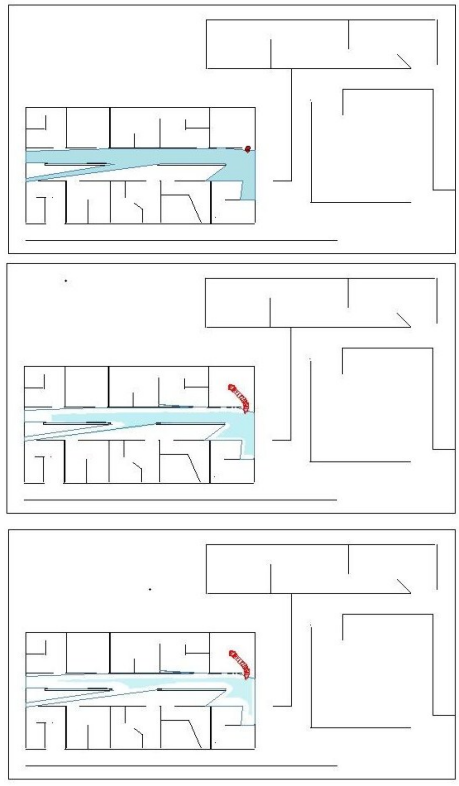

Figure 7 a. Initial pose 2 (top), b. First step (middle), c. Second step (bottom)

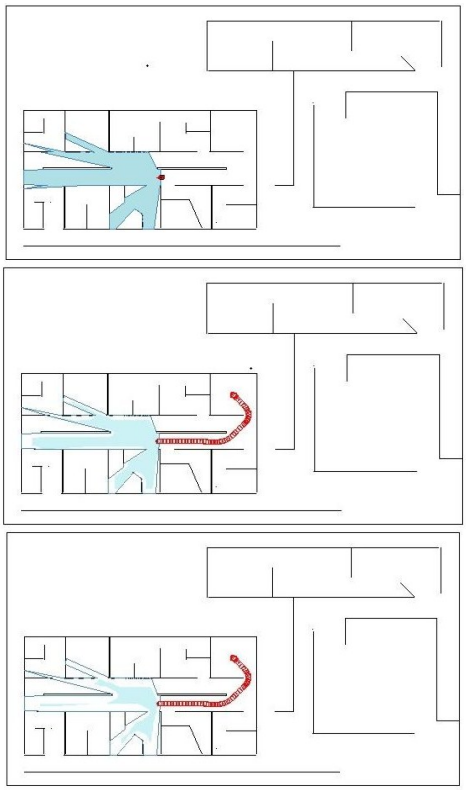

Figure 8 a. Initial pose 3 (top), b. First step (middle), c. Second step (bottom)

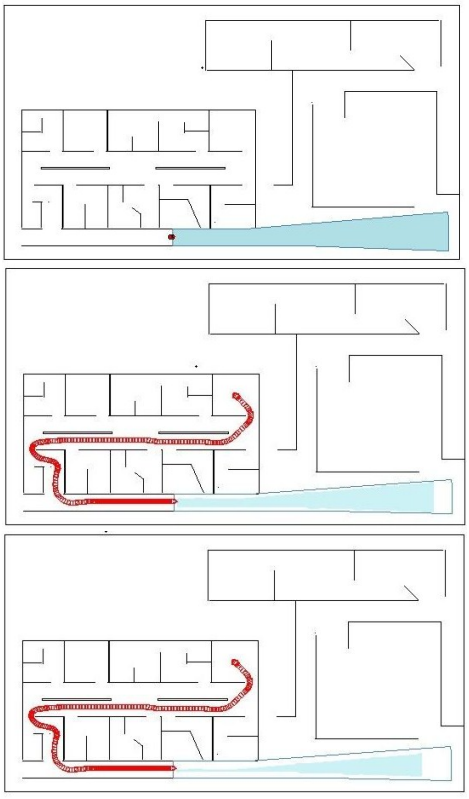

Figure 9 a. Initial pose 4 (top), b. First step (middle), c. Second step (bottom)

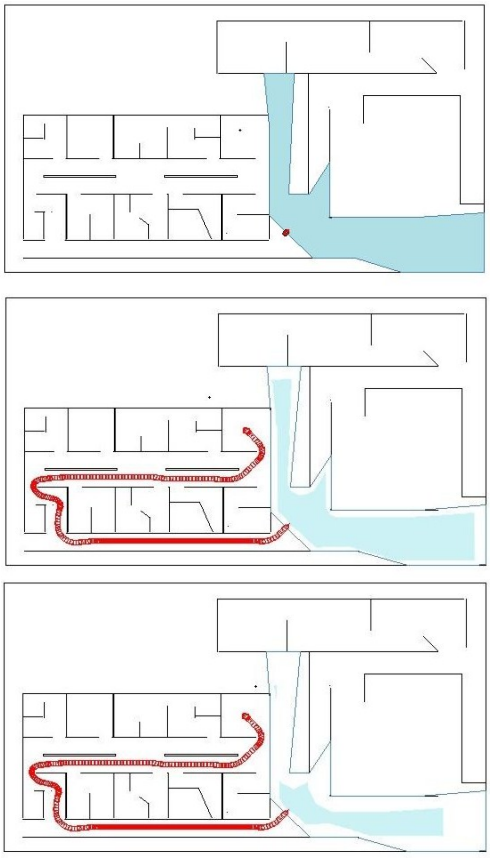

Figure 10 a. Initial pose 5 (top), b. First step (middle), c. Second step (bottom)

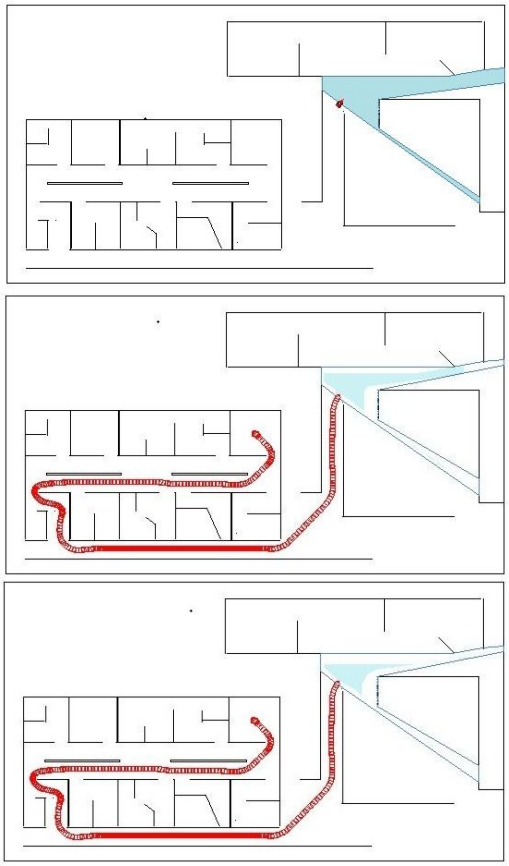

Figure 11 a. Initial pose 6 (top), b. First step (middle), c. Second step (bottom)