Online Local Path Planning for Mobile Robot Navigate in Unknown Indoor Environment

Mohamed Emharraf, Mohammed Saber, Mohammed Rahmoun and Mostafa Azizi

Abstract In this paper, the problem of path planning is studied for the case of a mobile robot moving in a priori unknown indoor environment, with static obstacles. There is often a need to replan paths online based on information extracted from the surroundings. The environment is modeled as a grid-map form of environment. The online replanning problem is solved using a new approach for A^* algorithm, called OA^*; Online A^* allowing the mobile robot navigate through obstacles in unknown environment and find the shortest feasible path from an initial position to a target position by avoiding the obstacles. Simulation results show the applicability and the feasibility of the approach.

Keywords Online path planning · Unknown environment · Indoor environment · Mobile robot

1 Introduction

Navigation in a priori unknown environments has a wide spectrum of applications in advanced robotics. Traditionally, this problem has been addressed either by having the robot build a map of the environment [1] (what can be seen from actual

M. Emharraf (✉) · M. Saber · M. Rahmoun
Laboratory Electronics, Computer and Image Systems,
National School of Applied Sciences, Oujda, Morocco
e-mail: m.emharraf@gmail.com

M. Saber
e-mail: mosaber@gmail.com

M. Rahmoun
e-mail: moha1rahmoun@gmail.com

M. Azizi
Laboratory Mathmatiques appliques, traitement du signal et informatique,
First Mohammed University, Oujda, Morocco
e-mail: azizi.mos@gmail.com
URL: http://www.ump.ma

A. El Oualkadi et al. (eds.), *Proceedings of the Mediterranean Conference on Information & Communication Technologies 2015*, Lecture Notes in Electrical Engineering 381, DOI 10.1007/978-3-319-30298-0_8

position) before planning the path, or by applying a deterministic algorithms that are able to cope with unknown environments [2].

Path planning is an important task in mobile robot intelligent control which should be performed efficiently. Planning a path means generate a collision free path in an environment with obstacles and optimize it [3, 4]. The environment map may be imprecise, vast, dynamical or non-structured [5]. In such environment, path planning depends on the robot sensory information about the environment, which might be associated with imprecision and uncertainty. Thus, to have a suitable path planning in such environment, the control system must be adaptive in nature of work area. If the environment is known and static, then the path generate in advance, it called off-line algorithm [6]. The planning is online [6], if it is capable of producing a new path in response to environmental changes. The path planning is the art of deciding which route to take for navigation under environment criterion. The idea presented in this paper is a new approach of A^* algorithm called Online A^*. In the order to update the A* algorithm for the unknown environment, which allow an autonomous mobile robot navigate in static unknown indoor environment based on sensors information about the environment and there position. Without any need such external information or a priori information about the environment. It can help the advantages of A^* in path determination and eliminating many of the drawbacks.

The rest of the paper is outlined as follows. Section 2 gives a description of path planning method. The new algorithm for the path planning is described in detail in Sect. 3. Section 4 provides the result through simulation and Sect. 5 concludes the paper.

2 Path Planning

Basic assumptions used in this approach:

- The robot has a short sensing range compared to the size of the environment.
- Robot senses radially from its position. Some obstacles stop the sensing in there directions.
- Robot knows its current position (coordinates, orientation, using local localization (dead reckoning).

2.1 Environment Presentation

For modeling the system, the local work area must be discretized. Discretizing the environment excludes lots of solutions. In this paper the terrain is described by nodes formed on squares/rectangles of equal size. Then one assigns a cost for a transition between two squares. Another straight forward approach, used by [7],

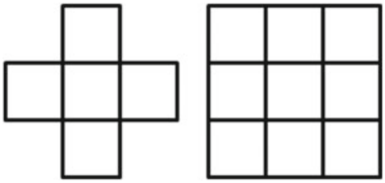


Fig. 1 **a** 4-neighbors, **b** 8-neighbors

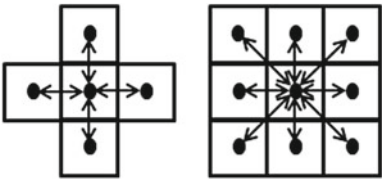


Fig. 2 4-neighbors and 8-neighbors mapped onto a node-arc pattern

is described by nodes connected by arcs. Each arc has a specified cost for moving along it.

A similar description of the environment is to first place nodes in a grid pattern, then assign costs for the transition between two nodes (squares). Two obvious representations are shown in Fig. 1. The two patterns in Fig. 1 are, as can be seen in Fig. 2, almost equivalent to the approach using nodes and arcs squares in a pattern. The squares structure (map grid) allows more planning options and makes complete presentation of the environment. For a map grid, to be an interesting approach. Must have a sensing range longer than the length to the longest considerable neighbor.

2.2 The Classic A^* Algorithm

The A^* operates [8] essentially the same as Dijkstras algorithm except that it guides its search towards the most promising states, potentially saving a significant amount of computation. A A^* plans a path from an initial start state to a goal state. To plan the path, the algorithms store an estimate g(s) of the path cost from the initial state to each state s. Initially, g(s) = ∞ for all states s. The algorithm begins by updating the path cost of the start state to be zero, then places this state onto a priority queue known as the OPEN list. Each element s in this queue is ordered according to the sum of its current path cost from the start, g(s), and a heuristic estimate of its path cost to the goal, h(s, goal). The state with the minimum such sum is at the front of the priority queue. The heuristic h(s, sgoal) typically underestimates the cost of the optimal path from s to goal and is used to focus the search.

The algorithm then pops the states at the front of the queue and updates the cost of all states reachable from this state through a direct edge: if the cost of state s, g(s), plus the cost of the edge between s and a neighboring state s0, c(s, s0), is less than the current cost of state s0, then the cost of s0 is set to this new, lower value. If the cost of a neighboring state s0 changes, it is placed on the OPEN list. The algorithm continues popping states off the queue until it pops off the goal state. At this stage, if the heuristic is admissible, i.e. guaranteed to not overestimate the path cost from any state to the goal, then the path cost of goal is guaranteed to be optimal.

The above approach works well for planning an initial path through a known map. However, when operating in real world scenarios, robots haven't perfect information. Rather, they may be equipped with incomplete or inaccurate planning graphs. In such cases, any path generated using the initial map may turn out to be invalid as it gets updated information.

3 The New Online A^* Approach

The general idea of OA^* algorithm is that a path is planned based on what is known right now. When the robot gets new information it considers updating the path. Gradual learning about the surroundings results in better plans. The information about the environment is translated to nodes costs. If the terrain is completely unknown the nodes can be initially assigned not affected by obstacles costs.

3.1 Operating Principle

The OA^* algorithm determinate the path to the goal in unknown a prior environment by following the next steps:

Step 1: Based on the present knowledge a path from the current node to the goal node is planned using an optimal solution (direct path) Fig. 3a.

Step 2: The robot follows the optimal path and explorer the environment in same time. If they reach the goal, the robot stops. If an obstacle detected in the path to the goal, the algorithm searches a new path to the goal step 3.

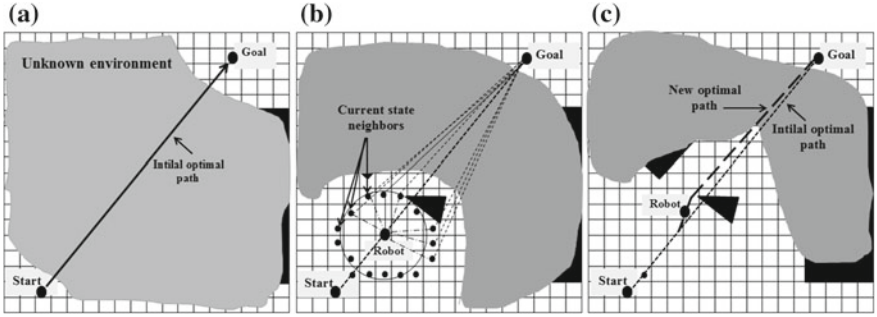


Fig. 3 Online OA star steps

Step 3: determinate all neighbors of the current state Sc in a circle with center Sc and radius the distance between Sc and the detected obstacle as Fig. 3b.

Step 4: compute the cost function for each neighbors determinate in step 3, using the cost function described in Sect. 3.2, and store the result in a queue (OPEN list) where the state (neighbor) with minimal cost are placed in the top.

Step 5: robot move to the state with minimal cost, return to step 1 Fig. 3c.

3.2 The Cost Function

The OA* method use the cost function to determinate the current optimal path to the goal. The cost function can be defined as below:

$$F(n) = G(n) + H(n) + T(n) + O(n) + M(n)$$

where

G(n) is the generation cost of the node i.e. from current node to node.

H(n) is heuristic cost of the node; here we consider the Euclidean distance between the goal and n node.

T(n) is the cost of trajectory node i.e. how often robot visit the node, the T(n) value given by mapping system [9].

O(n) present the occupancy of node i.e. low values for free or unknown node and very high value for occupied nodes, the O(n) value computed by mapping system [9].

M(n) is the cost of node neighbors i.e. low value if all neighbors are free nodes, and become high with number of occupied nodes neighbors.

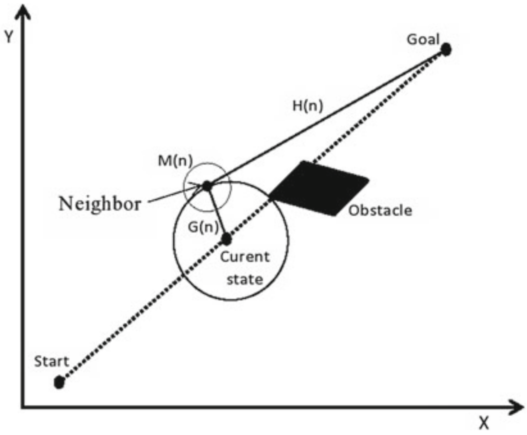
The next Fig. 4 presents the parameter of cost function.

4 Simulation Results

In this section, we give examples to illustrate the new path planning algorithm. The working space is of 36 × 36 grids, and all obstacles in the working space are described such that the mobile robot can be viewed as a point in a grid.

The path planning algorithm is programmed in LABVIEW platform on a computer. We use two matrices, one for the map occupancy O, and the other for map trajectory T [9]. The value of the element of the matrix O represents the occupancy of nodes (free, occupied, unknown); for example, if the n node is an obstacle, then the value of the element of matrix O(n) is infinite (in a program,

Fig. 4 Cost function parameters



assigned 100); if the m node is a free space, then the value of the element of matrix O(m) is 0. The value of the element of matrix O that represents mobile robot start or goal is 1, and the value of the element of matrix T that expresses the mobile robots trajectory grid is increment by 1 each time robot pass through the node (the values of these elements of the matrices were 0 before mobile robot passes).

Example 1: A mobile robot is passing through a simple environment with static obstacles, as shown in Fig. 5a. According to the path planning algorithm, the mobile robot will use initial path planning until an obstacle passes. Then, the mobile robot will program new path based on OA* algorithm. Figure 5b gives the path planning result. Where the red square is the position of the goal, the black squares are the positions of the obstacles, and the blue squares show the path planning. The algorithm find an optimal path to the goal through the unknown environment.

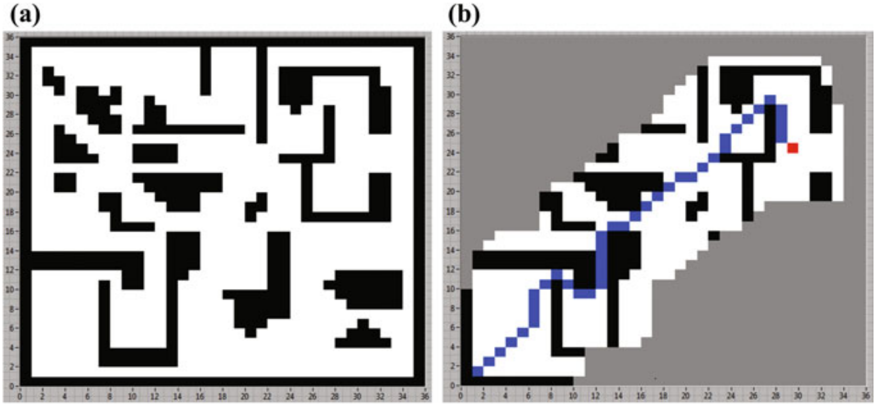


Fig. 5 Simple environment path planning

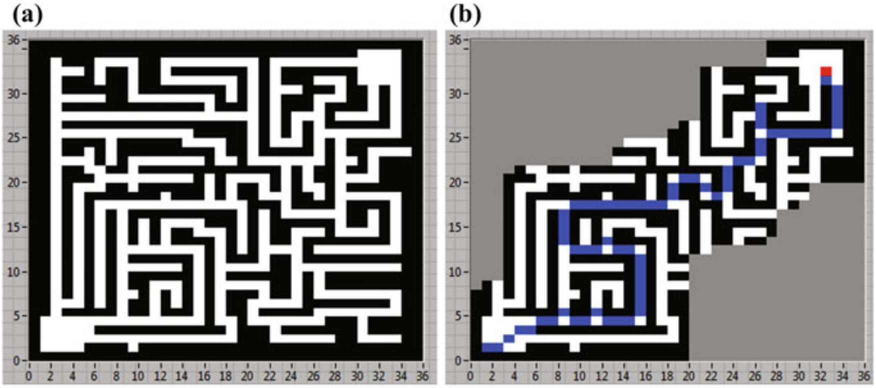


Fig. 6 Maze path planning

Example 2: for improving the feasibility of the algorithm, we use the robot in a small maze Fig. 6a, the result of simulation improve the feasibility of the algorithm in complexes environments Fig. 6b.

5 Conclusion

This paper presented an algorithm for path planning in an unknown environment based on the metric map. An improvement method for mobile robots path planning was explored. The simulation results have shown that the algorithm is efficient. The path planning algorithm has the following features:

(1) Easy to program
(2) Applicability to different static environments
(3) Convergence

References

1. Meryer, J.A., Filliat, D.: Map-based navigation in mobile robots: II. A review of map-learning and path planning strategies. Cogn. Syst. Res. **4**, 283–317 (2003)
2. Ferguson, D., et al.: A guide to heuristic-based path planning. In: Proceedings of the Workshop on Planning Under Uncertainty, for Automated Planning and Scheduling (ICAPS-05), Monterey, CA (2005)
3. Howard, A., Matari, M.J., Sukhatme, G.S.: An incremental self-deployment algorithm for mobile sensor networks, autonomous robots. Intell. Embedd. Syst. **13**(2), pp. 113–126 (2002)
4. Florczyk, S.: Robot Vision Video-based Indoor Exploration with Autonomous and Mobile Robots. WILEY-VCH Verlag GmbH & Co. KGaA, Weinheim (2005)

5. Janglova, D: Neural networks in mobile robot motion. Int. J. Adv. Robot. Syst. **1**(1), 15–22 (2004). ISSN 1729-8806
6. Al Marzouqi, M., Jarvis, R.A.: Robotic covert path planning: a survey. In: 2011 IEEE Conference on Robotics, Automation and Mechatronics (RAM), pp. 77, 82, 17–19 Sept 2011
7. Ersson, T., Hu, X.: Path planning and navigation of mobile robots in unknown environments. In: Proceedings of the IEEE International Conference on Intelligent Robots and Systems (IROS) (2001)
8. Bellman, R.: Dynamic Programming. Princeton University Press (1957)
9. Emharraf, M., Rahmoun, M., Saber, M., Azizi, M.: Mobile robot unknown indoor environment exploration using self-localization and grid map building. In: 2014 9th International Conference on Intelligent Systems: Theories and Applications (SITA-14), pp. 1, 5, 7–8 May 2014