Adel Akbarimajd, Akbar Hasan Zadeh/ International Journal of Engineering Research and Applications (IJERA) ISSN: 2248-9622 www.ijera.com
Vol. 1, Issue 4, pp.1262-1267

A Novel Cellular Automata Based Real Time Path Planning Method for Mobile Robots

Adel Akbarimajd*, Akbar Hasan Zadeh**

*(Electrical and Computer Engineering Group, Department of Engineering
University of Mohaghegh Ardabili, Ardabil, Iran
** (School of Electrical and Computer, College of Engineering,
Shiraz University, Shiraz, Iran)

ABSTRACT

Path planning method for mobile robots based on two dimensional cellular automata is proposed. The method is appropriate for multi-robot problems as well as dynamic environments. In order to develop the planning method, environment of the robot discretized by a rectangular grid and the automata with four states is defined including Robot cell, Free cell and Obstacle cell and Best goal directing cell. Evolution rule of automata are proposed in such a way that at each step time the robot cell is exchanged with best goal directing cell.

Keywords - Cellular automata, Computational Intelligence, Mobile robot, Path Planning

I. INTRODUCTION

Path-planning problem for a mobile robot means finding a free path from an initial position to a goal position in the configuration space [1]. The robot must move around obstacles without colliding them and reach into its goal. There are plenty of techniques for path planning of mobile robots if the map of the environment and obstacles is completely known and off-line planning is acceptable. Under these assumptions, existing path planning methods can fall into three main categories [2]: cell decomposition, road map and potential field methods. In cell decomposition methods, configuration space is decomposed into some cells where by connecting neighboring cells a graph is constructed. This connectivity graph should be searched in order to find appropriate path from initial cell to goal cell. In roadmap methods a network of 1-D curves is made to illustrate connectivity of the free spaces. The network is afterwards considered as a set of standard paths in which the path of robot must be found. In the potential field methods, robot's environment is modeled as an artificial potential field where attractive magnetic potentials are assigned for robots and goal and repulsive potentials are assigned to robots and obstacles.

In all of above approaches a comprehensive and precise representation of configuration space is required which by itself yields large computational cost in offline planning. In general, the complexity rises exponentially with the number of degrees of freedom of a robot and the dimensions of configuration space [4]. Moreover dynamic environments would impose much more complicated problem. Multi-robot multi-goal assumption would also increase the complexity of problem.

In summary, finding a real time path planning method with satisfactory performance in dynamic multi-robot multi-goal environments has been an extremely difficult task. None of existing works in this area (see the next section) are applicable for dynamic multi-robot multi-goal environments as well as being real time and using only local representations of the environment. The aim of this paper is to develop cellular automata (CA) based path planning algorithm in order to address this challenge.

The rest of this paper is organized as the sequel. The next section includes related research to this paper. In Section III main concept of cellular automata is introduced. In forth section the proposed path planning method is devised. Section V and Section

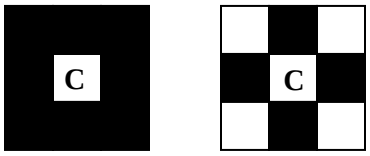


Fig.1 Two definitions of neighborhood of central cell (**C**) in Cellular Automata: Neumann neighborhood (left) and Moor neighborhood (right)

VI include simulation results and conclusions respectively.

II. Related Works

Because of highly nonlinear and computationally expensive properties of path planning problem of mobile robots, developing efficient planning methods has been on focus of interest. Varity of techniques based on computational intelligent algorithms have been used. Genetic algorithms [5], fuzzy systems [6], ant colonies [7], particle swarm optimization [9] and artificial bee colonies [10] are examples.

Von Neumann [11] developed Cellular Automata as simple models of spatially distributed processes. The idea is then extended and outspreaded by Burks[12][13]. A cellular automaton is a grid composed of set of cells with discrete possible values where each cell evolve as a function of time according to states of neighboring cells and based on a set of rules. Cells in CA are simple components with local links; however, CA as a multi-agent system has the ability of doing complicated computations in a highly robust and efficient manner. Consequently, CA has become a popular modeling system and computational tool in mathematics, natural science, computer science and technologies. Training CA to perform image processing [14], design of reconfigurable robots [15], prediction of protein sub-cellular location [16] modeling phenomena of urban growth [17] and modeling earthquake activity features [18] are some examples. CA have also been used for computational tasks and high speed simulations in scientific models [19].

Because of capabilities in distributed computation and their local computation characteristics, CA is a good candidate to act as a fast and reliable path planning tool. Shu and Buxton introduced a basic CA based path planning algorithm for mobile robots in simple environments [20]. A collision-free planning algorithm for a diamond-shaped robot was developed by Tzionas et al [21]. In their work, free space is mapped onto the Voronoi diagram which is constructed through the time evolution of cellular automata. In [22] it was illustrated that an efficient computation method can be developed by cellular automata in order to find a collision free path from initial to goal configuration on a physical space occupied by obstacles in arbitrary locations. Marchese presented a reactive path-planning method for a non-holonomic mobile robot by multilayered cellular automata [23].

III. Cellular Automata

Cellular automata is a *d*-dimensional colored grid composed of cells where each cell can be in one of a finite number of states (colors) [11]. States could be some integer numbers or some symbols. Cells are finite-state machines those can be specified by index *n* where $n=1,\ldots,N$. In a two dimensional $I\times J$ automata (d=2), for example, we have $N=I\times J$. Each cell may interact with adjacent cells in order to update its state. CA is updated based on some local rule that is identical for all cells [11][12]. S_n^t denotes the state of a cell *n* at time step *t*. In *CA*'s rule, current states of a cell and the states of its neighborhood is delivered as inputs and the next state of central cell is returned as output. Rules may be demonstrated by a function or by a lookup table. Here, definition of neighborhood is a very important concept. This definition is straightforward in one-dimensional grids but in two dimensional automata, different definitions of neighborhood exist. For example *Neumann neighborhood* considers all adjacent cells of a central cell as neighbors and *Moore neighborhood* only considers the top, bottom, left and right cells as neighbors (See Fig.1). In this paper we will use Neumann neighborhoods.

Fig. 2 shows a one-dimensional (d=1) nearest neighbor (with radius r=1) cellular automata with two possible states (binary) and with grid length N=8. The grid and evolution rule table are illustrated.

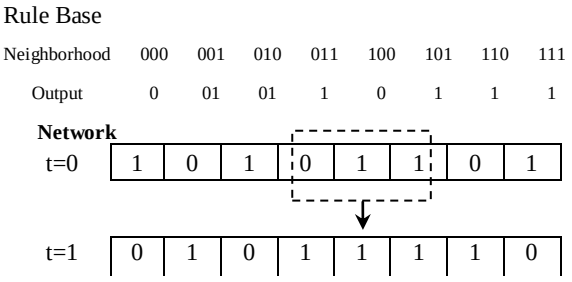


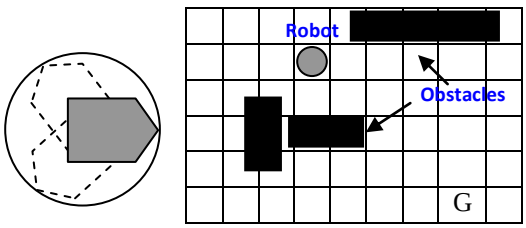
Fig. 2. Binary one-dimensional cellular automata

The progress of evolution is shown at two time steps.

IV. PATH PLANNING ALGORITHM

A. *Set up of Cellular Automata*

In this section we describe our path planning problem of mobile robot and establish a cellular automaton in order in order to appropriately modeling of the problem. We assumed that the robot can recognize its own position vector $\vec{P}_r = [x_r, y_r]^T$ at each step time. It is also assumed that the robot knows goal position vector $\vec{P}_g = [x_g, y_g]^T$ and by a mechanism with short sensing depth it can identify if the adjacent cells are free cells or not. The sensory system can be one rotary sonar sensor or 8 fixed sensors mounted on the robot. To handle probable non-homonymic characteristics, the robot is encircled into a circular robot as shown in Fig. 3.a and it is assumed to be a free flying object. The work-space is divided into a rectangular grid of cells (see Fig. 3.b) so that the robot can be entirely placed inside a cell. We have so far built two-dimensional cellular automata with



a. The robot (considering different orientations) can be encircled

b. Robot cell, obstacle cells and free cells in the work-space

Fig 3. Building up path planner cellular automata

F	F	F	F	O	O	O	O	O
F	F	F	R	F	F	F	F	F
F	O	O	F	B	F	F	F	F
F	O	O	O	O	F	F	F	F
F	F	O	O	F	F	F	F	F
F	F	F	F	F	F	F	F	F
F	F	F	F	F	F	F	F	F

Goal

Fig 4. Cellular automata corresponding to the robot workspace of Fig 2.b with four states: *{F, O, R, B}*

three possible states including *free cell*, *obstacle cell* and *robot cell* represented by *F*, *O*, *R*, respectively. Now we should find a way to incorporate relative position of the robot and goal in our automata. To this end we define the best goal directing cell as the following:

Definition: The *best goal directing cell* is the closest free cell to the goal in the neighborhood of the robot cell.

If we show the best goal directing cell by *B*, our automata would have four possible states for each cell: *{F, O, R, B}*. Fig. 4 shows a schematic view of CA constructed in accordance to workspace of Fig. 3.b.

The robot has to determine the best goal directing cell in each step of its evolution. In order to develop a routine of finding the best directing cell, we assign direction vectors to adjacent cells as Fig. 5. Now, it is easy to discover the best goal directing cell based on angle between those direction vectors and the vector

$\vec{D}_1 = [-1,-1]^T$	$\vec{D}_2 = [-1,0]^T$	$\vec{D}_3 = [-1,1]^T$
$\vec{D}_4 = [0,-1]^T$	**Robot Cell**	$\vec{D}_5 = [0,1]^T$
$\vec{D}_6 = [1,-1]^T$	$\vec{D}_7 = [1,0]^T$	$\vec{D}_8 = [1,1]^T$

Fig. 5. Direction vectors correspond to adjacent cells of the robot cell.

$$\mathbf{K}=\left\{k\in\{1,...,8\}\;\middle|\;\frac{\left\langle \vec{D}_k,\vec{P}_g-\vec{P}_r\right\rangle}{\left|\vec{D}_k\right|}=\max\left(\frac{\left\langle \vec{D}_k,\vec{P}_g-\vec{P}_r\right\rangle}{\left|\vec{D}_k\right|}\right)\right\}$$

if $\overline{\overline{\mathbf{K}}}>1$

$\mathbf{K}=\left\{k\in\mathbf{K}\;\middle|\;\vec{D}_k(2)=\min\left(\vec{D}_k(2)\right)\right\}$

if $\overline{\overline{\mathbf{K}}}>1$

$\mathbf{K}=\left\{k\in\mathbf{K}\;\middle|\;\vec{D}_k(1)=\max\left(\vec{D}_k(1)\right)\right\}$

Number best goal dircting cell $=k\in\mathrm{K}$

Fig. 6. Pseudo code of subroutine of finding the best goal directing cell. $\overline{\overline{\mathbf{K}}}$ denotes cardinality of set **K**.

$\vec{P}_g-\vec{P}_r$. This angle can be calculated using dot product approach. Among eight neighboring cells (k=1,..,8), the cell corresponding to the maximum value of the dot product would be marked as *B*. Fig. 6 illustrates the routine of finding *B* cell.

B. Evolution Rules

After establishing cellular automata for robot environment, appropriate updating rules have to be devised to direct the robot toward its goal without colliding with obstacles. It is preferred that the evolution rules use only local information and do not need a complete representation of the configuration space.

Evolution rules presented in this section are based on a simple sense: "robot cell *R* moves into the best goal directing cell *B* in each step and leaves a free cell *F* in its location". This can be implemented by the pseudo code represented in Fig. 7. For example, if we apply this rule to the CA of Fig. 4 the block of cells marked by gray is updated as the sequel:

while Robot cell (R) $\neq$ Goal Cell (G) ***do***
  for n=1:N
    if $S_n^t==R$
      find best goal directing cell: $S_j^t=B$
      $S_n^t=R$;
      $S_n^t=F$;
    elseif $S_n^t\neq R$
      $S_n^{t+1}=S_n^t$

Fig. 7. Pseudo code of evolutionary rule of CA

$$\begin{bmatrix}F & F & O\\ F & R & F\\ O & F & B\end{bmatrix}\Rightarrow\begin{bmatrix}F & F & O\\ F & F & F\\ O & F & R\end{bmatrix}$$

and the state of other cells remain unchanged. It is noteworthy to mention that in a single robot problem only the states of two cells are changed in each step time.

If the automaton is evolved according to rules of Fig. 7, the robot cell would become more and more close to goal cell at each step time. Since the number of cells of the grid is finite there are no concave obstacles, after a finite time steps the robot cell would meet the goal cell.

Identifying the best goal directing cell is a crucial requirement in the updating rule of Fig. 7. Routine of Fig. 5 is used as a subroutine to handle this requirement.

C. Multi Robot Environments

The path planning method introduced in previous subsections can be easily generalized for multi-robot multi-goal problems. To this end, for example in a two-robot problem, possible states of cells should be extended to *{F, O, B, R_1, R_2}*. The rule of Fig. 7 would be sufficient to achieve a successful path planning automata if it is sequentially applied to robots. For approaching each robot to its own goal, the same inference of single-robot problem still holds. To verify that the robots do not collide, without loss of generality, consider two robots in Fig. 8. If we apply the rules to robot R_1 in Fig. 8.a, it moves into cell B which is assumed to be the central cell in our case. Now in Fig. 8.b. that old B cell is not a free cell any longer and will not be treated as new B cell for R_2.

R_2	R_1	F
F	F	O
F	O	F

a. Before applying the rules

R_2	F	F
F	R_1	O
F	O	F

b. After applying the rules for

Fig. 8. The robots do not collide with each other in multi-robot problem

V. SIMULATION RESULTS

In the first stage of simulations, we considered a robot environment shown in Fig. 9. A model is constructed in MATLAB. The proposed CA based planning method is applied for two different initial-goal sets. Paths resulted by the algorithm are illustrated Fig. 9 and imply that the algorithm has been able to find appropriate path witch is the shortest path in approximation.

In the second stage of simulations, two robots assumed to be located at two different initial points where their goals are different. Fig. 10.a and Fig. 10.b depict paths of the robots obtained by applying the algorithm to the problem. To verify that the robots do not collide with each other, positions Robot cells in different time steps are shown in Fig. 10.c. It is obvious that the robots are not in the same cell at the same time, while each travels along an acceptable path.

VI. CONCLUSIONS

Because of capabilities of CA in local evolution and high-speed parallel computation, it was employed to be basis of an online and real-time path planning technique for mobile robots. The work-space of robot was modeled as two dimensional cellular automata with four possible states: Robot cell, Free cell, Obstacle cell and the Best goal directing cell. It is assumed that the robot has a short sensing and only is able to identify states of only adjacent cells. Based on a dot product of goal cell vector and some direction vectors, updating rules of the automata was

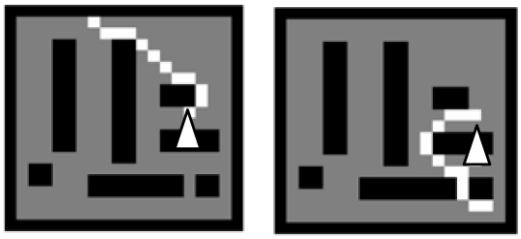
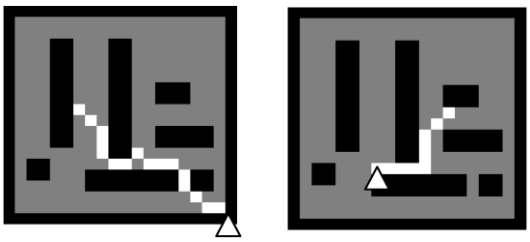


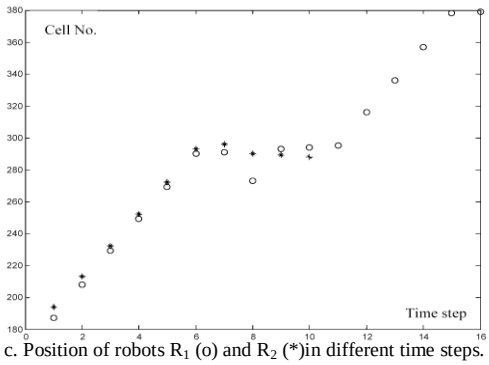
Fig. 9. Simulation of single robot problem for two set of initial-goal points. Paths created using proposed CA based algorithm. (Δ shows goal points)

constructed to direct the robot into a cell which is nearest to the goal. The method is a real-time method applicable for both single-robot and multi-robot problems. Significance of the proposed algorithms was verified by simulations in MATLAB.

This research can be completed in future by extending the proposed method for concave obstacles and finding a method in order to efficiently implement it.



a. Path of robot R1 b. Path of Robot R2



c. Position of robots R_1 (o) and R_2 (*)in different time steps. The robots are not in the same cell at the same time.

Fig. 10. Two robot problem

REFERENCES

[1] J.Xiao, and L.Zhang, Adaptive evolutionary planner/navigator for mobile robots, *IEEE transactions on Evolutionary Computation, Vol. 1, no. 1, pages. 18-28, April 1997.*

[2] J. C. Latombe, *Robot Motion Planning* Kluwer Academic Publishers, Eighth print, 2004,

[3] S.M. Lavalle, Motion planning: the essentials, *IEEE robotics and automation magazine, Vol 18, No. 1, March 2011.*

[4] J. H. Reif., Complexity of the mover's problem and generalizations, *In Proceedings of the IEEE Symposium on Foundations of Computer Science, pages 421-427, 1979.*

[5] M. Kapanoglu, M. Ozkan, A. Yazıcı and Osman Parlaktuna, Pattern-based genetic algorithm approach to coverage path planning for mobile robots, *Lecture Notes in Computer Science, Volume 5544, 2009*

[6] M.A. Porta Garcia, O. Montiel, O. Castillo, R. Sepúlveda, and Patricia Melin, Path planning for autonomous mobile robot navigation with ant colony optimization and fuzzy cost function evaluation, *Journal of Applied Soft Computing Volume 9, Issue 3, Pages 1102-1110, June 2009*

[7] S.H. Chia, K. L. Su, J. H. Guo, C.Y. Chung, Ant colony system based mobile robot path planning,

[8] *Fourth International Conference on Genetic and Evolutionary Computing, Shenzhen, China, December 2010*

[9] E. Masehian, D. Sedighzadeh,, Multi-objective PSO- and NPSO-based algorithms for robot path planning, *Advances in Electrical and Computer Engineering, Volume 10, Issue 4, On page(s): 69 – 76, 2010*

[10] Q. Ma and X. Lei, Dynamic path planning of mobile robots based on ABC algorithm, A*rtificial Intelligence and Computational Intelligence, Lecture Notes in Computer Science, Volume 6320/2010, 267-274, 2010.*

[11] J von Neumann, *Theory of Self-reproducing Automata (edited and completed by A. W. Burks)* Urbana, IL: University of Illinois Press. 1966

[12] A. W. Burks, *Von Neumann's Self-reproducing Automata* In Burks 1970.

[13] A W. Burks, *Essays on Cellular Automata*, IL: University of Illinois Press, 1970

[14] Paul L. Rosin, Training cellular automata for image processing, *IEEE Transaction on Image Processing, Vol. 15 Issue: 7, July 2006*

[15] S Murata, H Kurokawa, Self-reconfigurable robots, *IEEE Robotics and Automation Magazine, Vol. 14, Issue 1, March 2007*

[16] X. Xiao, S. Shao, Y. Ding, Z. Huang and K.-C. Chou, Using cellular automata images and pseudo amino acid composition to predict protein sub cellular location, *Journal of Amino Acids, Vol. 30, No 1, pages 49-54 2006*

[17] J. Han, Y. Hayashi, X. Cao, H. Imura, Application of an integrated system dynamics and cellular automata model for urban growth assessment: A case study of Shanghai, China, *Journal of Landscape and Urban Planning, Vol. 91, Issue 3, pp 133-141, 2009.*

[18] IG. Georgoudas, GC. Sirakoulis, I. Andreadis, Modeling earthquake activity features using cellular automata, *Journal of Mathematical and Computer Modelling, Vol. 46, Issues 1-2, pp 124-137, 2007*

[19] J. Maddox, The Universe as a fractal structure. *Nature, 329(17) 195, 1987*

[20] C. Shu and H. Buxton, Parallel path planning on the distributed array processor, *Parallel Computing Vol. 21, Issue 11, pp 1749-767, 1995*

[21] P.G. Tzionas, A. Thanailakis, P.G. Tsalides, Collision-free path planning for a diamond-shaped robot usingtwo-dimensional cellular automata, *IEEE Transactions on Robotics and Automation, Vol. 13, Issue 2, pp 237-250, 1997,*

[22] C. Behring, M. Bracho, M. Castro, J. A. Moreno, and Emergente, An algorithm for robot path planning with cellular automata. *in Proceedings of the Fourth Int. Conference on Cellular Automata for Research and Industry, 2000,*

[23] F. M. Marchese, A directional diffusion algorithm on cellular automata for robot path-planning, *Future Generation Computer Systems, Vol.18, Issue 7, pp 983 – 994, 2002*