CELLULAR AUTOMATA BASED PATH-PLANNING ALGORITHM FOR AUTONOMOUS MOBILE ROBOTS

Rami Al-Hmouz, Tauseef Gulrez & Adel Al-Jumaily

Information and Communications Group
ARC Centre of Excellence in Autonomous Systems
University of Technology Sydney, Australia
ralhmouz | tgulrez | adel @eng.uts.edu.au

Abstract: This paper presents the application of Cellular Automata (CA) model in solving the problem of path planning. It is shown that a CA allows the efficient computation of a path from an initial to a goal configuration on a physical space cluttered with obstacles. The cellular space represents a discrete version of the workspace. The method was experimentally tested on an autonomous mobile robot on real-time software player/stage, in all cases very good paths were obtained with negligible computer effort. These simulation results indicate that the Cellular Automata approach is a very promising method for real time path planning.

Keywords: Cellular Automata, path planning.

1. INTRODUCTION

Path planning is essential problem need to be solved in autonomous mobile robot. Automatic motion planning has application in many areas such as robotics, virtual reality systems, and computer-aided design. Although many different motion planning methods have been proposed, most are not used in practice since they are computationally infeasible except for some restricted cases like the case of mobile robots Many researchers have proposed different solutions during the last 20 years. Since (1979, Lozano-Pérez et.al) first proposed a path-planning algorithm among polyhedral obstacles based on the visibility graph, then extended to the Configuration Space approach (T. Lozano-Pérez *et.al* 1983). These types of approach require a geometrical description of the environment. The reconstruction of geometrical primitives from sensorial data is usually difficult and time-consuming when in the real life the environment is changing.

Therefore the updating phase of the world model becomes a relevant part of the navigation algorithm. The path-planners working on these models generate very precise optimal trajectories and can solve really difficult problems, especially in cluttered worlds, also taking into account non-holonomic constraints, but they are very time consuming, too. In the literature diverse algorithms have been proposed to tackle this problem: Some of them, such as the randomized potential field methods (H. Chang et.al 1995) represent the robot as a particle moving under the influence of an artificial potential field produced by the sum of a repulsive potential, generated by the obstacles, and an attractive potential, generated by the goal configuration. The path is obtained by a descent along the negative gradient of the total potential. Other researcher's work is based on

roadmaps (D. Hsu, 2000) and (L. Kavraki, *et.al* 1995) in which a network of one-dimensional curves, called the roadmap, lying in the free space of the workspace is constructed. The final path results from the concatenation of three sub-paths, one connecting the initial configuration to the roadmap, another belonging to the roadmap and a final one from the roadmap to the goal configuration. Another general approach is based on a division of the free space into a set of exact or approximate cells (E. Anshelevich, *et.al* 2000) and (H. Gutowitz, *et.al* 1990).

This paper describes a very fast, safe and complete routing method for a cluttered environment based on the very simple rules of cellular automata. The task is to build a robot navigation system that works in real-time, while interacting with the environment and reacting as fast as possible to its dynamical events.

The organization of the paper is as follows: In the next two sections introductions to the configuration space formalism and Cellular Automata are presented. In the fourth section the proposed path planning algorithm is described. In the fifth section the complexity of the algorithm is examined and in the last section some experimental results are discussed, with the concluded remarks and the future work.

2. CONFIGURATION SPACE

Path Planning problem can generally be considered as a search in a configuration space: Let *A* be a single rigid object, moving in a Euclidean space W= R^N, *N* = 2 or 3 , Let $B_1....B_n$ be fixed rigid bodies distributed in *W*. The $B_i's$ are called as obstacles in workspace *W,* and the obstacles $B_1....B_n$ are the closed subsets of *W*, a configuration of *A* is a specification of the position of every point in *A* with respect to F_W, where F_W is a Cartesian coordinate system. The configuration space of *A* is the space denoted by *C*, with all possible configurations of *A*. The subset of *W* occupied by *A* at configuration *q* is denoted by *A(q)*. A path from an initial configuration q_{init} to a goal configuration q_{goal} is a continuous map $\tau:[0,1]\to C$ with $\tau(0)=q_{init}$ and $\tau(1)=q_{goal}$. The workspace contains a finite number of obstacles denoted by B_i with i = 1…N. Each obstacle B_i maps in *C* to a region $C(B_i)=\{q\in C \mid A(q)\cap B_i\neq 0\}$ which is called as $C_{obstacle}$. The Union of all $C_{obstacle}$ is the $C_{obstacle,region}$ $\bigcup_{i=1}^{n} C(B_i)$ and the set $C_{free}=C-\bigcup_{i=1}^{n} C(B_i)$. A *free collision path* between two configurations is any continuous path $\tau:[0,1]\to C_{free}$. The configuration space is a powerful conceptual tool because it seems to be the natural space where the path-planning problem lies. This is mainly because any transformation of a rigid or articulated body becomes a point in the configuration space.

3. CELLULAR AUTOMATA

The proposed path planning method, consists on the successive application of two simple local Cellular Automata (*CA)* transition rules (H. Gutowitz, *et.al* 1990). CA is a decentralized extended system consisting of a large number of identical entities with local connectivity arranged on a regular array. It consists of the following components: (1) A *cellular space*: a regular lattice of cells each with identical finite state machines and the same local pattern of connectivity along with definite boundary conditions. Here a square 2 D bitmap lattice. (2) A *set of states* Σ with cardinality state k = $|\Sigma|$ over which the finite-state machines takes values. Each cell of the lattice is denoted by an index *i* and its state at time *t* is represented by S_i^t. The neighborhood η_i^t, *transition rule* $S_i^{t+1}=\phi(\eta_i^t)$ which establishes the way in which each cell of the automata is to be updated. The transition rule is applied synchronously to each cell in the CA, defining an intrinsic parallel dynamic.

$$S_i^{t+1}=\begin{Bmatrix} 1 if S_i^t=0 \wedge \exists x\in\eta_i^t \,|\, S_x^t=1 \\ S_i^t otherwise \end{Bmatrix} \qquad (1)$$

with S_i^t as the state of the automata *i* at time *t* and η_i^t Moore neighborhood of cell *i*. The objective of this initial dynamics is the growth of each obstacle a number of cells to account for the physical size of the path point. The number of iterations will depend on the size of the path gap (safest path's conditions) this number is up to four. The schematic diagram in figure 1 shows this process.

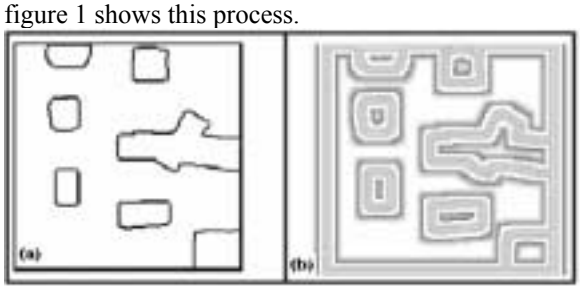

Fig. 1.Cellular Automata Process. (a) The abstract of environment showing free space and bounded space shows obstacle region, (b) After applying

the transition rules on the environment the obstacles grow.

In the third phase, the final configuration resulting from phase 2 is used as initial state for a second CA dynamics that computes the shortest distance between the initial and goal positions. In this second cellular automata the possible states for the cells are the following: (0) free, (1) obstacle, (2) initial position, (3) goal position, (4) distance *l* to the goal... (3+*l*) distance *l* to the goal. The transition rule applied to evolve this cellular automata is:

$$S_i^{t+1} = \begin{Bmatrix} S_x^t + 1 if S_i^t = 0 \wedge \exists x \in \eta_i^t \middle| S_x^t \geq 3 \\ S_i^t otherwise \end{Bmatrix} \quad (2)$$

This process that resembles a flood from the goal to the initial position is shown in figure 2. The flood dynamics is stopped either when the cell with the initial position is reached or when all cells in the cellular space are different from 0 in which case no path is possible. In the former case a path is calculated by going backwards from the goal to the start in a descending manner. Due the existence of saddle points in the navigation function, the shortest path between an initial and goal position is not well defined.

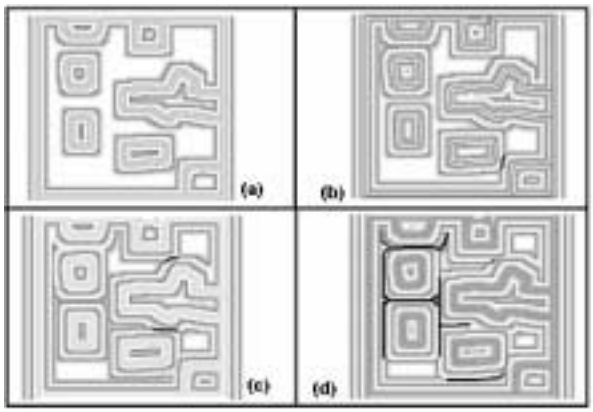

Fig.2. Flood Dynamics through CA. (a) First CA iteration (expansion starts), (b) The sides collide shown in black, (c) The result from third iteration and more collisions shown in black, (d) The fourth and last iteration and more collisions recorded.

Some times a cell has more than one neighbor with the same shortest distance to the goal. To follow always the steepest descend of the function may not work because there could be cases where the gradient may have the same value in several directions. In order to select a reasonable good path two heuristics are applied among the neighbors of a cell: (1) The cells that fulfill the condition of being in direction of the steepest descend and allows the conservation of the direction of movement for the route are selected in the path with the highest priority or (2) the next priority is for those cells in direction North, South, West and East and have the smallest angle to the moving direction of the robot are selected. In general it is found experimentally that these two heuristics efficiently produces reasonable good paths with few minimal changes of the direction of movement (commands for the route). The knowledge of the sequence of cells that constitute a path together with the initial direction of movement of the route allows in a straightforward manner the production of a list of commands which guides the route along the path to its goal.

4. THE ALGORITHM

The workspace space is an Euclidean space in 2D, and it is decomposed in a finite collection of convex polygons, squares in this case, called cells such that their interiors do not intercept. These constitute the cellular space of the CA. The configuration space is hence represented by a set of these discrete square cells, with the obstacles occupying a certain number of cells. Assuming a grid of given size, the discrete configuration space can be defined by:

$$C = \{(x,y) | x \in \{0,...,x_{\max}\}, y \in \{0,...,y_{\max}\}\} \quad (3)$$

Every configuration q_i corresponds to a point *(x,y)*, each obstacle is a set :

$$B_i = \bigcup_{j=1}^{q_i} (x_j, y_j) \quad (4)$$

If each of the n obstacles of size *r*, is decomposed into a set of rB_i of size one each, then

$$C_{obstacle} = \bigcup_{j=1}^{nxr} (x_j, y_j) \quad (5)$$

The free space is defined by $C_{free} = C\text{-}C_{obstacle}$.

The computational cost for cellular automata, with a simple transition rule, is proportional to the number of updates executed on the cells. Hence for a two dimensional automata with a cellular space of $(x_{\max} \times y_{\max} \times i)$ where *i* is the number of update iterations. In the application of the transition rule each cell executes two operations: reading and writing, of the state of the cell itself and of those cells in its neighborhood. In consequence, in a complete evolution, the number accessed cell is:

$$(x_{\max} \times y_{\max} \times i) \times (2+|\eta| \quad (6)$$

where $|\eta|$ is the size of neighborhood. The first CA dynamics in the proposed algorithm produces a growth of the congestions (obstacles) in ω cells, therefore each cell will be visited:

$$(x_{\max} \times y_{\max} \times i) \times (2+|\eta|) \times \omega \quad (7)$$

times. The upper bound for the order of this CA process is : $O\ (x_{\max} \times y_{\max})$.

Start

1. Preprocessing of the Environment

2. Compute the configuration space
Free, obstacle ,initial-position & Goal-position
3. Update the computed configuration space with CA
No of update iterations ($x_{\max} \times y_{\max} \times i$)
Transition rule each (Blob) cell executes two operations: reading and writing
($x_{\max} \times y_{\max} \times i$) × (2+ $|\eta|$), *where* $|\eta|$ *is the size of neighborhood.*
4. Compute the Shortest distance
The expansion till the edges meet with each other.
Read and write
If *Not Goto 3*
Else if *the map fully cellularly automated*
Compute the shortest path from initial to goal
Finish

The second CA dynamics computes the distance and the basic number of cell accessed is: ($x_{\max} \times y_{\max} \times i$)× (2+ $|\eta|$)× *l,* where *l* is the Shortest distance from initial to goal position. The worst case occurs when the path contains nearly half of the cells, so the number of cell accesses is given by ($x_{\max} \times y_{\max} \times i$)× (2+ $|\eta|$) × *l* × ($\frac{x_{\max} \times y_{\max}}{2}$). The order of this process is $O(x^2_{\max} \times y^2_{\max})$.In the best case the initial and goal positions are neighbors, with a number of accesses equal to ($x_{\max} \times y_{\max}$)× (2+ $|\eta|$). The final phase of the algorithm deals with the calculation of the optimal route path: each neighbor of a cell on the path has to be visited in order to select the best heuristic path. Then (2+ $|\eta|$) × *l* visits have to be performed and depending on one of the following cases: worst case (2+ $|\eta|$) × *l* × ($\frac{x_{\max} \times y_{\max}}{2}$), and best case (2+ $|\eta|$).

Finally when the algorithm determines the path, commands for each cell of the path has to be examined along the entire path length, therefore the upper bound is of order *O(l)*.

5. SIMULATION AND RESULTS

After running the program, the occupied space obstacles might expand and collide with each other; the collision lines produced by the expansion of obstacles could be joined to make a path. This path is so far the safest path as far as the present scenario is concerned. Afterward, the collision lines are connected to each other as shown in figure 3.

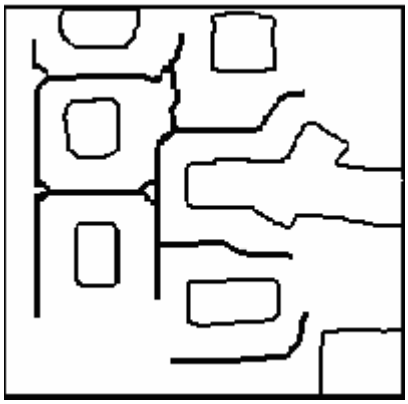

Fig. 3. The final environmental extraction with CA, and many paths has been created which are maintaining the safe distance from the obstacles.

Once the lines generated are joined, this will create many paths within the same map, the selection of path starts when the 'start' and 'goal' destination are added to the environment as shown in figure 4. These points will determine the real safest and shortest path selection for the robot. Based upon the start and goal points finally a real path is obtained as in figure 4.

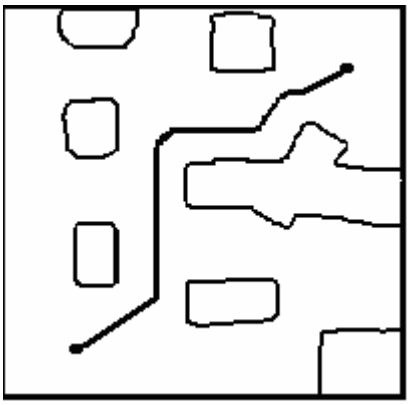

Fig. 4. The shortest path has been selected from the predefined start and goal position, resulting in the distance *l*.

The algorithm was tested using real-time software player/stage see (R. Vaughan *et.al* 2004) on autonomous robot. The results were first tested on the real-time simulation environment, where the map was created with arbitrary obstacles, computer generates a very fine path for the robot and robot traveled very safely as in figure 5,6.

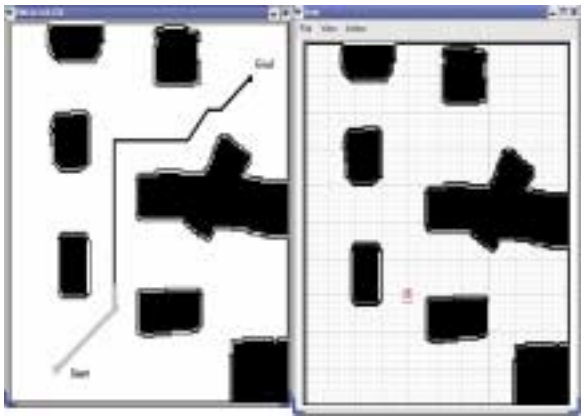

Fig. 5. The map sent to the software for robot to move, right hand side robot (red) is moving and on the left hand side its movement can be seen on the window in grey movement.

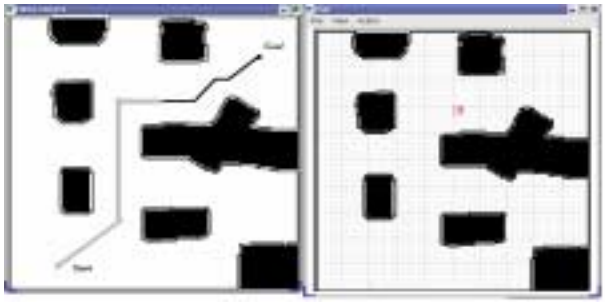

Fig. 6. The robot successfully reaches its destination.

The results show the possibility in a real time path palnning scenario using the proposed CA based algorithm. In experimental trials with a Pentium III 1.6 GHz. machine, and using an input image with 160*120 pixels resolution, an average response of 399 ms per path was obtained. When using images of 80*60 pixels resolution, a 112 ms average response per path was measured. The results are a true picture of real time Pioneer 2DX autonomous robot (fig 7), while using CA based planner; hence robot can be moved fast among the cluttered obstacle environment.

Fig. 7. Pioneer 2DX Robot.

6. CONCLUSIONS

A cellular automata approach for solving the path planning problem yields very efficient experimental performance in real time situations. The cellular automata algorithms were tested with different workspace configurations and cellular space sizes over real time images from a digital camera. The computer effort depends on the size of the cellular space and the length of the resulting path. This reduced time complexity together with the simplicity of the cellular automata simulation allows the algorithm to perform quite well on serial machines. Some of the key practical advantages of the method are: it does not require parameter tuning, real time performance for any kind of workspace scene, consistent behavior over repeated experiments; it can handle path-planning in dynamic environment. In conclusion the practical performance is very satisfactory however further study involving many more benchmarking examples are necessary.

7. FUTURE WORK

The next step is to use CA in a cluttered environment, while using probabilistic roadmaps methods as a shortest path method, while trying to implement the same theory on the distributed wireless networks project as well, where many unforeseen problems arise.

REFERENCES

Anshelevich. E, S. Owens, F. Lamiraux, and L. Kavraki, “Deformablevolumes in path planning applications,” in Proc. IEEE Int. Conf. Robotics and Automation, 2000, pp. 2290–2295.

Chang. H and T. Y. Li, “Assembly maintainability study with motionplanning,” in Proc. IEEE Int. Conf. Robotics and Automation, 1995, pp.1012–1019.

Donald. B, K. Lynch, and D. Rus, New Directions in Algorithmic and Computational Robotics . Natick, MA: A. K. Peters, 2000.

Finn P.W and L. E. Kavraki, “Computational approaches to drug design,” Algorithmica, vol. 25, pp. 347–371, 1999.

Gutowitz H, Cellular Automata. The Mit. Press,. Cambridge, 1990.

Hsu D, “Randomized single-query motion planning in expansive spaces,” Ph.D. dissertation, Dept. Comput. Sci., Stanford Univ., Stanford, CA, 2000.

Kavraki L, J. Latombe, R. Motwani, and P. Raghavan, “Randomized query processing in robot motion planning,” in Proc. ACMSymp. Theory of Computing, 1995, pp. 353–362.

Lozano-Pérez T, M.A. Wesley, An algorithm for planning collision-free paths among polyhedral obstacles, Commun. ACM 22 (10) (1979) 560–570.

Lozano-Pérez T, Spatial planning: a configuration space approach, IEEE Trans. Comput. C 32 (2) (1983) 108–120.

Vaughan R and B. Gerkey, “Player Version 1.4rc2Manual”, http://playerstage.sourceforge.net/doc/Player-manual-1.4-html/node1.html. (2004).