

An Algorithm for Robot Path Planning with Cellular Automata

C. Behring¹, M. Bracho², M. Castro and J. A. Moreno³.

1. Lehrstuhl für Systemanalyse, Universität Dortmund, Fachbereich Informatik,
D-44221, Dortmund, Germany
e-mail: carsten@alfred2000.ping.de

2. Universidad Centroccidental “Lisandro Alvarado”, Decanato de Ciencias,
Av. Las Industrias, Núcleo Obelisco. Barquisimeto, Venezuela
e-mail: mbracho@delfos.ucla.edu.ve

Laboratorio de Computacion Emergente,
3. Universidad Central de Venezuela, Facultades de Ciencias e Ingeniería,
Caracas, Venezuela
e-mail: {mcastro, jose}@neurona.ciens.ucv.ve

Abstract

The application of Cellular Automata to the problem of robot path planning is presented. It is shown that a Cellular Automata allows the efficient computation of an optimal collision free 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 for a mobile robot. The path computation is performed by the successive application of two simple local transition functions. The method was experimentally tested for a small robot on a real environment, in all cases very good paths were obtained with negligible computer effort. These results indicate that the Cellular Automata approach is a very promising method for real time robot path planning

1 Introduction.

The ultimate goal of robotics is to build artificial agents capable of displaying rational and complex behaviors in the accomplishment of a specific task. The tasks under consideration are characterized by the need of meaningful interactions of the agent with a real dynamic world through a physical body [1]. The achievement of this goal requires the development of a series of innovative technologies and an extensive effort on research in areas such as artificial intelligence and emergent computing, [2]. In this context the present paper deals with a very important aspect: the real time planning of a free-collision route for the robot to move from an initial to a goal position. This problem is well known in robotics and is called the basic motion planning problem [3],[4],[5]. Such problems involve searching the system configuration space for a collision-free path that connects a given start and goal configurations, while satisfying constraints imposed by a complicated obstacle distribution. This definition of the problem simplifies some of the aspects of robot

motion planning. The dynamic properties of the robot are ignored and the problem is transformed to a purely geometrical path planning problem.

In the literature diverse algorithms have been proposed to tackle this problem: Some of them, such as the randomized potential field methods [6], [7] 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. Others are based on roadmaps [8],[9] 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 subpaths, 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 [3], [10]. The path is a sequence of cells with the following properties: (1) The first cells contains the initial configuration. (2) The final cell contains the goal positions. (3) Neighboring cells in the sequence are cells in free space with a common boundary.

In spite of the diversity of available path planning methods their computational complexity, is an important limiting factor in their practical applicability. In contrast the path planning algorithm proposed in this work is based on an intrinsically parallel methodology and hence its complexity allows real time planning.

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.

2 Configuration Space

Robot path planning can generally be considered as a search in a configuration space defined in what is known as the configuration space formulation [3] [6]: Let A be a single rigid object, moving in a Euclidean space $W = \mathfrak{R}^N$, $N = 2$ or 3 . Let B_1, \dots, B_n be fixed rigid objects distributed in W . The B_i 's are called obstacles. If A is described as a compact subset of W , and the obstacles B_1, \dots, B_n are 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 $\mathbf{t} : [0,1] \rightarrow C$ with $\mathbf{t}(0) = q_{init}$ and $\mathbf{t}(1) = q_{goal}$.

The workspace contains a finite number of obstacles denoted by B_i with $i = 1, \dots, n$. Each obstacle B_i , maps in C to a region $C(B_i) = \{q \in C \mid A(q) \cap B_i \neq \emptyset\}$ which is

called $C_{obstacle}$. The union of all the $C_{obstacle}$ is the $C_{obstacle}$ region, $\bigcup_{i=1}^n C(B_i)$ and the set

$$C_{free} = C \setminus \bigcup_{i=1}^n C(B_i) = \left\{ q \in C \setminus A(q) \cap \left(\bigcup_{i=1}^n B_i \right) = 0 \right\}$$

is called the free space. A *free collision path* between two configurations is any continuous path $\mathbf{t} : [0,1] \rightarrow C_{free}$.

The configuration space is a powerful conceptual tool because it seems to be the natural space where the path planning problem “lives”. 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 [11][12][13][14][15]. A 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. In the method a square 2 D. lattice. (2) A *set of states* \mathbf{S} with cardinality $k = |\Sigma|$ 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 . (3) The *neighborhood* \mathbf{h}_i^t

transition rule $s_i^{(t+1)} = \mathbf{f}(\mathbf{h}_i^t)$, 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.

4 The Algorithm.

In this approach, a free flying robot without dynamic and kinematic constrains is considered. It consists therefore of a single point without any consideration about its orientation. The workspace space is an Euclidean space in \mathbb{R}^2 , 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

$$C = \{ (x, y) \mid x \in \{0, \dots, x_{max}\}, y \in \{0, \dots, y_{max}\} \}. \quad (1)$$

Every configuration q_i corresponds to a point (x, y) . Each obstacle is a set

$B_i = \bigcup_{j=1}^{q_i} (x_j, y_j)$. If each of the n obstacles of size r , is decomposed into a set of

$r B_i$ of size 1 each, then $C_{obstacle} = \bigcup_{j=1}^{n \times r} (x_j, y_j)$. The free space is defined by

$C_{free} = C \setminus \bigcup_{j=1}^{n \times r} (x_j, y_j)$ The following Figure shows a typical discrete configuration

space.

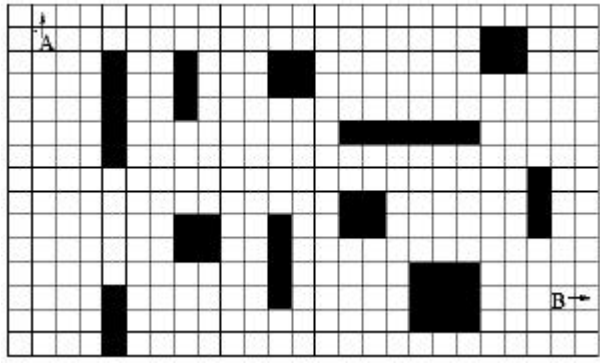


Figure 1: Discrete Configuration Space. The workspace is decomposed in a set of square cells. The obstacles occupy a certain number of cells. The initial and goal positions are shown as points A and B respectively.

The robot, considered as a point, occupies exactly one cell of size 1. Therefore, the path is a finite discrete sequence of configurations $(q_{init}, q_1, \dots, q_{goal})$.

The input of the algorithm is the image taken by the camera, containing information about the position of each object in the workspace. In phase one of the algorithm, the image information is transformed to the discrete cellular space with four possible initial states: (0) free, (1) obstacle, (2) initial position, (3) goal position. Then in phase two, using a Moore neighborhood a first transition rule is applied:

$$s_i^{t+1} = \begin{cases} 1 & \text{if } s_i^t = 0 \wedge \exists x \in \mathbf{h}_i^t \mid s_x^t = 1 \\ s_i^t & \text{otherwise} \end{cases} \quad (2)$$

with s_i^t as the state of the automata i at time t and \mathbf{h}_i^t the 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 robot. The number of iterations will

depend on the size of the robot and on the size of the cell, for typical resolutions this number is up to four. The schematic diagram in Figure 2 shows this process:

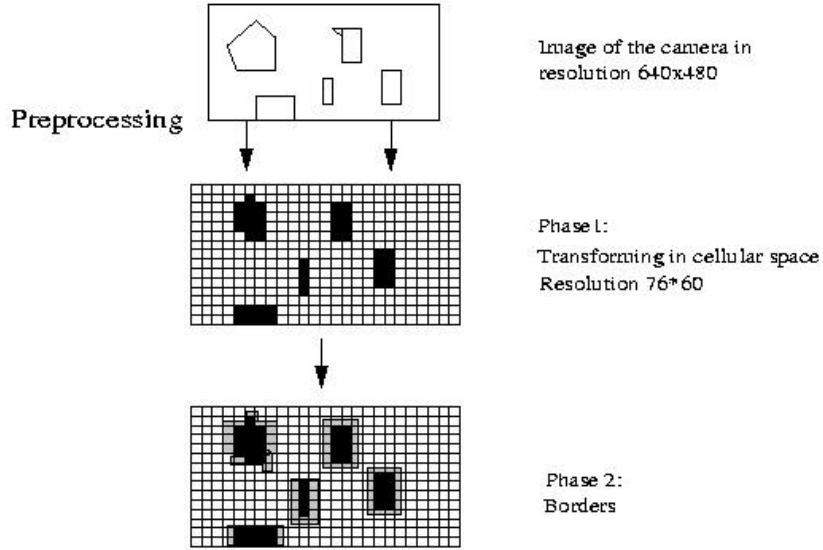


Figure. 1: Phase 1 and 2 of the Algorithm. Process of transforming the image information into the cellular space and growth dynamics of the obstacles.

In the third phase, the final configuration resulting in phase 2 is used as initial state for a second CA dynamics that computes the Manhattan distance [16] 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) Manhattan distance 1 to the goal,....., (3+l) Manhattan distance l to the goal. The transition rule applied to evolve this cellular automata is

$$s_i^{t+1} = \begin{cases} s_x^t + 1 & \text{if } s_i^t = 0 \wedge \exists x \in \mathbf{h}_i^t \mid s_x^t \geq 3 \\ s_i^t & \end{cases} \quad (3)$$

This process that resembles a flood from the goal to the initial position is shown in figure 3. 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.

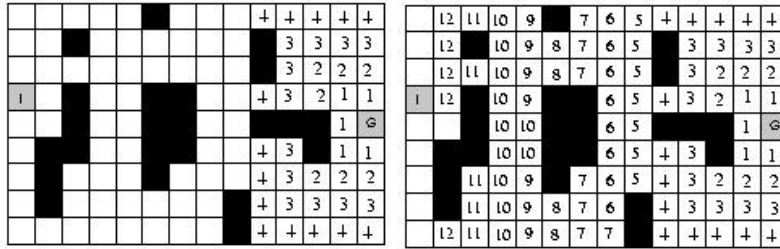


Figure 2: The Flood Dynamics. The second cellular automata dynamics computes the Manhattan distance (numbers in the cells) between each cell and the goal position, - dark point at the right size of each figure -. a) 4th iteration. b) 13th iteration.

Some times a cell has more than one neighbor with the same Manhattan 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 robot 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 robot).

The knowledge of the sequence of cells that constitute a path together with the initial direction of movement of the robot allows in a straightforward manner the production of a list of commands which guides the robot along the path to its goal.

5 Complexity of the Algorithm

The computational cost for a 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 $(x_{max} \times y_{max})$ cells the cost is proportional to $(x_{max} * y_{max} * i)$, 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 cells is $(x_{max} * y_{max} * i) * (2 + |h|)$, where $|h|$ is the size of neighborhood.

The first CA dynamics in the proposed algorithm produces a growth of the obstacles in w cells, therefore each cell will be visited $(x_{max} * y_{max}) * (2 + |h|) * w$ times. The upper bound for the order of this CA process is $O(x_{max} * y_{max})$.

The second CA dynamics computes the Manhattan distance and the basic number of cell accessed is $(x_{max} * y_{max}) * (2 + |h|) * l$, where l is the Manhattan 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

$(x_{max} * y_{max}) * (2 + |h|) * l * (\frac{x_{max} * y_{max}}{2})$. The order of this process is $O(x_{max}^2 * y_{max}^2)$. In the best case the initial and goal positions are neighbors, with a number of accesses equal to $(x_{max} * y_{max}) * (2 + |h|)$. The process for this case is of order $O(x_{max} * y_{max})$. The final phase of the algorithm deals with the calculation of the optimal path: each neighbor of a cell on the path has to be visited in order to select the best heuristic path. Then $(|h| + 2) * l$ visits have to be performed and depending on one of the following cases: worst case $(2 + |h|) * (\frac{x_{max} * y_{max}}{2})$, and best case $(2 + |h|)$.

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

In conclusion, the overall complexity of the path planning algorithm is of the order of $O(x_{max} * y_{max})$ in the best case and $O(x_{max}^2 * y_{max}^2)$ in the worst case. These results indicate that it is possible to do real time motion planning using the proposed CA based algorithm. In experimental trials with a Pentium Pro-S 200 MHz. 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.

6 Experiments

The experiments were carried out on a real life testbed consisting of a small mobile robot chasing a ball on a table tennis sized football field according to RoboCup regulations [17]. The following pictures in Figure 4 and 5 show a sequence of instances in the execution of the proposed algorithm. The images are taken by a Connectix Color QuickCam 2 digital camera placed above the workspace and working in a resolution of 320x240 and 24bit mode with a frame rate of 5.6. The scene consists of an arbitrary obstacle distribution:

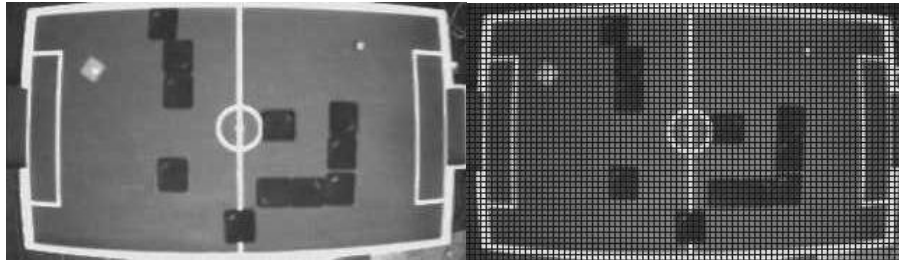


Figure 3: (a) Original image of the scene with the robot, ball and obstacles. (b) Image with a superimposed square grid of 8x8 pixels.

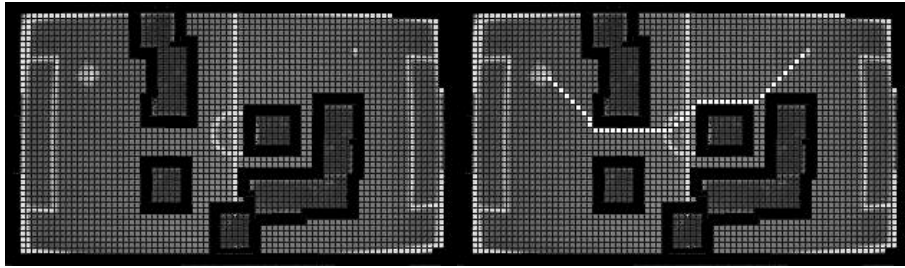


Figure 4: (c) Scene after the execution of the first CA dynamics. (d) Scene after the execution of the second CA dynamics with the selected collision free path.

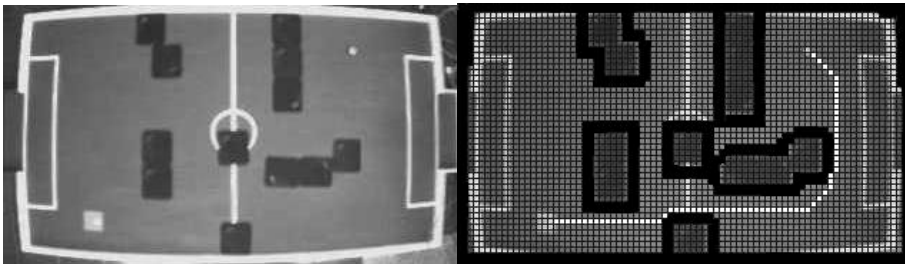


Figure 5: (a) Original images of the scene with the robot, ball and obstacles. (b) After the execution of the second CA dynamics with the selected collision free path.

It is observed that the actual execution time of the algorithm depends on several factors, the most important of which are the frame rate of the camera and the data transfer rate by the parallel port that limits the transfer velocity. In fact, the executing program waits for the camera, in its process of acquiring the image and transferring the data, in order to compute the path.

7 Conclusions

A cellular automata approach for solving the robot path planning problem that yields very efficient experimental performance on real life path planning situations is proposed. 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.

References

1. Asada M., Stone P., Kitano H., et al.: The RoboCup Physical Agent Challenge: Goals and Protocols for Phase I. In: Kitano Hiroaki (ed). RoboCup -97: Robot Soccer World Cup I.: Springer-Verlag, Berlin, 1997, pp 42-61 (Lectures Notes in Artificial Intelligence.; no. 1935)
2. Kitano H., Asada M., Kuniyoshi Y., et al.: RoboCup: A Challenge Problem for AI and Robotics. In: Kitano Hiroaki (ed). RoboCup -97: Robot Soccer World Cup I.: Springer-Verlag, Berlin, 1997, pp 1-19 (Lectures Notes in Artificial Intelligence.; no. 1935)
3. Latombe J. C. Robot Motion Planning. Kluwer Academic Publishers, Boston, 1991
4. Canny J. F. The Complexity of Robot Motion Planning. MIT Press, Cambridge, 1988
5. LaValle S. M. Robot Motion Planning: A Game Theoretic Foundation. PhD thesis, University of Illinois. 1995 <http://janowiec.cs.iastate.edu/~lavalle/>. Iowa State University.
6. Barraquand J., Langlois B. and Latombe J. C. Numerical Potential Field Techniques for Robot Path Planning. IEEE, Transactions on System, Man and Cybernetics. 1992; 22(2):224--241.,
7. Hwang, Y. and Ahuja, N. A Potential Field Approach to Path Planning. IEEE, Transactions on Robotics and Automation. 1992; 8(1).
8. Amato, N. and Wu, Y. A Randomized Roadmap for Path Manipulation Planning. IEEE, International Conference on Robotics and Automation. 1996; pp 113--120.
9. Kavraki, L. E. and Latombe, J. C. Randomized Preprocessing of Configurations Space for Path Planning. IEEE International Conference on Robotics and Automation. 1994; pp 2138--2139.
10. Lozano Pérez, T. An Algorithm for Planning Collision Free Path among Polyhedral Obstacles. Communication of the ACM 1979; 22(10): 560--70.
11. Gutowitz, H. Cellular Automata. The Mit. Press., Cambridge, 1990
12. Mitchell M. Computation in Cellular Automata: A Selected View. In: Gramms, T. and Shuster H. G. (eds). Nonstandard Computation. VCH Verlagsgesellschaft. 1996
13. Smith A. Simple Computation Universal Cellular Spaces. Journal of the ACM 1971; 18(3): 339-353.
14. Toffoli T. and Margolus M. Cellular Automata Machine. The Mit. Press, Cambridge, 1987
15. Wolfram, S. Theory and Applications of Cellular Automata. World Scientific, Singapore, 1986
16. Berlekamp E., Conway J. and Guy R. Gewinnen. Strategien für Mathematische Spiele. Vieweg, Braunschweig, 1985.
17. <http://RoboCup.org/regulations/4.htm>