Path-Planning for Multiple Generic-Shaped Mobile Robots with MCA

Fabio M. Marchese and Marco Dal Negro

Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano - Bicocca
Via Bicocca degli Arcimboldi 8, I-20126, Milano, Italy
fabio.marchese@disco.unimib.it

Abstract. In this paper is described a fast Path-Planner for Multi-robot composed by mobile robots having generic shapes and sizes (user defined) and different kinematics. We have developed an algorithm that computes the shortest collision-free path for each robot, from the starting pose to the goal pose, while considering their real shapes, avoiding the collisions with the static obstacles and the other robots. It is based on a directional (anisotropic) propagation of attracting potential values in a 4D Space-Time, using a Multilayered Cellular Automata (MCA) architecture. This algorithm searches for all the optimal collision-free trajectories following the minimum valley of a potential hypersurface embedded in a 5D Time-Space.

1 Introduction

In this paper we describe a safe path-planning technique for multiple robots based on Multilayered Cellular Automata. The aim is to design a Coordinator for multi-robot systems that interacting with the environment and reacting as fast as possible to its dynamical events, decides the motions of a team of robots. Many authors have proposed different solutions during the last twenty-five years, based, for example, on a geometrical description of the environment (e.g. [10, 11]). The path-planners working on these types of models generate very precise optimal trajectories and can solve really difficult problems, also taking into account non-holonomic constraints, but they are very time consuming, too. In our opinion, to face a real dynamical world, a robot must constantly sense the world and re-plan accordingly to the new acquired information. Other authors have developed alternative approaches less precise, but more efficient: the Artificial Potential Fields Methods. In the eighties, Khatib [7] first proposed this method for the real-time collision avoidance problem in a continuous space. Jahanbin and Fallside first introduced a wave propagation algorithm in the Configuration Space ($\mathcal{C}$-*Space*) on discrete maps (*Distance Transform* [6]). In [2], the authors used the Numerical Potential Field Technique on the $\mathcal{C}$-*Space* to build a generalized Voronoi Diagram. Zelinsky extended the *Distance Transform* to the *Path Transform* [15]. Tzionas et al. in [13] described an algorithm for a diamond-shaped holonomic robot in a static environment, where they let a CA to build a

V.N. Alexandrov et al. (Eds.): ICCS 2006, Part III, LNCS 3993, pp. 264–271, 2006.

Voronoi Diagram. In [14] propose the coordination of robots using a discretized 3D $\mathcal{C}$-*Space-Time* (2D workspace and time) for robots with the same shape (only square and circle) and a quite simple kinematics (translation only). La Valle in [8] applies the concepts of the Game Theory and multiobjective optimization to the centralized and decoupled planning. A solution in the $\mathcal{C}$-*Space-Time* is proposed in [4], where the authors use a decoupled and prioritized path planning in which they repeatedly reorder the robots priorities to try to find out a solution. It can be proofed that these approaches are not complete. In this paper, we introduce a development of previous works of the authors on single robot path-planning: the multi-robot path-planning. To face with multiple robots coordination it is necessary to introduce the Time to handle precisely the space occupation of each robot in every instant. We have used CA as a formalism for merging a Grid Model of the world (Occupancy Grid) with the $\mathcal{C}$-*Space-Time* of multiple robots and Numerical (Artificial) Potential Field Methods, with the purpose to give a simple and fast solution for the path-planning problem for multiple mobile robots, with generic shapes and kinematics. This method uses a directional (anisotropic) propagation of distance values between adjacent automata to build a potential hypersurface embedded in 5D space. Applying a constrained version of the descending gradient on the hypersurface, it is possible to find out all the admissible, equivalent and shortest (for a given metric of the discretized space) trajectories connecting two positions for each robot $\mathcal{C}$-*Space-Time*.

2 Problem Statements

A wide variety of world models can be used to describe the interaction between an autonomous agent and its environment. One of the most important is the Configuration Space [9, 11]. The $\mathcal{C}$-*Space* $\mathcal{C}$ of a rigid body is the set of all its configurations $\mathbf{q}$ (i.e. poses). If the robot can freely translate and rotate on a 2D surface, the $\mathcal{C}$-*Space* is a 3D manifold $\mathbb{R}^2 \times \mathbf{SO}(2)$. It can be modelled using a 3D Bitmap $\mathcal{GC}$ ($\mathcal{C}$-*Space Binary Bitmap*), a regular decomposition in cells of the $\mathcal{C}$-*Space*, represented by the application $\mathcal{GC} : \mathcal{C} \rightarrow \{0, 1\}$, where 0s represent non admissible configurations. The $\mathcal{C}$-*Potential* is a function $\mathbf{U}(\mathbf{q})$ defined over the $\mathcal{C}$-*Space* that "drives" the robot through the sequence of configuration points to reach the goal pose [2]. Let us introduce some other assumptions: 1) space topology is finite and planar; 2) the robot has a lower bound on the steering radius (non-holonomic vehicle). The latter assumption introduces important restrictions on the types of trajectories to be found.

Cellular Automata are automata distributed on the cells of a Cellular Space $\mathbb{Z}^n$ (a regular lattice) with transition functions invariant under translation [5]: $\mathbf{f}_c(\cdot) = \mathbf{f}(\cdot), \forall c \in \mathbb{Z}^n$, $\mathbf{f}(\cdot) : \mathbf{Q}^{|A_0|} \rightarrow \mathbf{Q}$, where $\mathbf{c}$ is the coordinate vector identifying a cell, $\mathbf{Q}$ is the set of states of an automaton and $\mathbf{A}_0$ is the set of arcs outgoing from a cell to the neighbors. The mapping between the Robot Path-Planning Problem and CA is quite simple: every cell of the $\mathcal{C}$-*Space Bitmap* $\mathcal{GC}$ is an automaton of a CA. The state of every cell contributes to build the $\mathcal{C}$-*Potential* $\mathbf{U}(\mathbf{q})$ through a diffusion mechanism between neighbors. The trajectories are found following the minimum valley of the surface $\mathbf{U}(\mathbf{q})$. In

this work, we use a simple extension of the CA model: we associate a vector of attributes (state vector) to every cell. Each state vector depends on the state vectors of the cells in the neighborhood. There is a second interpretation: this is a Multilayered Cellular Automaton [1], where each layer corresponds to a subset of the state vector components. Each subset is evaluated in a single layer and depends on the same attribute of the neighbor cells in the same layer and depends also on the states of the corresponding cell and its neighbors in other layers. In the following sections, we describe each layer.

3 Multilayered Architecture

In Fig. 1 is shown the layers structure and their dependencies. There are two main layers: *Obstacles Layer* and the *Attraction Layer*. Each layer is subdivided in more sublayers: the *Obstacles L.* has 3 dimensions (2 for the workspace (X, Y) and 1 more for the time), while the *Attraction L.* has up to 5 dimensions (1 for the robots, 2 for the robots workspaces + 1 for their orientations (X, Y, Θ) and 1 for the time). The *Obstacles L.* conceptually depends on the

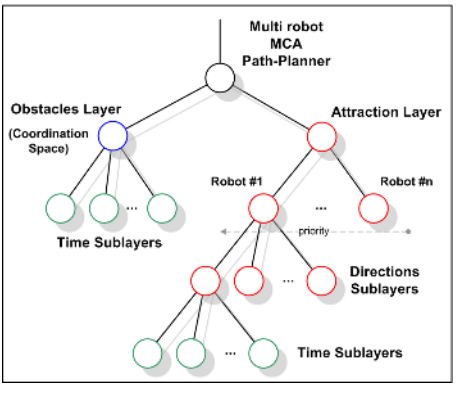

Fig. 1. MultiLayers Architecture

outside environment. Its sublayers have to react to the "external" changes: the changes of the environment, i.e. the movements of the obstacles in a dynamical world. Through a sensorial system (not described here), these changes are detected and the information is stored in *Obstacles L.* permitting the planner to replan as needed.

3.1 The Obstacles Layer

The main role of the *Obstacles Layer* is to generate a repulsive force in the obstacles to keep the robots away from them. In the present work, only static obstacles are considered (e.g. walls). For the single robot, the other robots are seen as moving obstacles, with unknown and unpredictable trajectories. We are

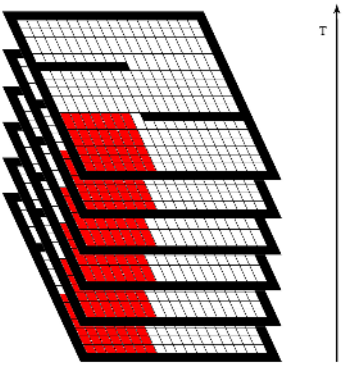

Fig. 2. Discretized *C-Space-Time* and *C-Obstacles*

considering a centralized planner/coordinator, that can decide (and, of course, it knows) the trajectories of all the supervised robots. Thanks to this knowledge, the planner considers the silhouette of the robot as an obstacle for the other robots. In this work, we introduce a discretized version of the *C-Space-Time* as in [14]. With the introduction of the Time axis, the robots moving in the environment become "static" obstacles in the Space-Time 4D space, having a pose in each time slice (Fig. 2). We can still call them as *C-Obstacles*, remembering that they are extended also in the Time dimension. In our case, we have a 3D Space-Time: $R^2 \times R$, where R^2 is the planar workspace. The time step (slice) is the time needed by a robot to move to an adjacent cell, if we consider all the robots moving at the same speed (the robots motions are synchronized).

3.2 The Attractive Layer

The *Attractive Layer* is the core of the algorithm. It computes the shortest collision-free path for a single robot, from the starting pose to the goal pose, while considering its real occupation (shape and size = silhouette). The planner is able to handle at the same time different type of robots, with different shapes, sizes and kinematics (e.g. car-like kinematics, omnidirectional, etc.). To pass from one cell of *C-Space-Time* to an other one, the robot can follow different paths, combining different atomic moves, such as *strict forward move*, *diagonal move*, *rotation*, and so on. Each move has its own cost (always positive); the entire set of admissible movements define the robot kinematics. The moves costs are used to build incrementally a potential surface starting from the goal cell, placed at a high time slice (the desired time of arrival), and expanding it in all the free space-time. In our case, it is a hypersurface embedded in a 4D space: $R^2 \times SO(1) \times R$, where R^2 is the workspace, enlarged with the robot orientation dimension ($SO(1)$) and the time. The goal cell receives the first value (a seed), from which a potential bowl is built adding the cost of the movement that would bring the robot from a surrounding cell to it. The calculus is performed by the automata in the space-time cell, which computes the potential value depending

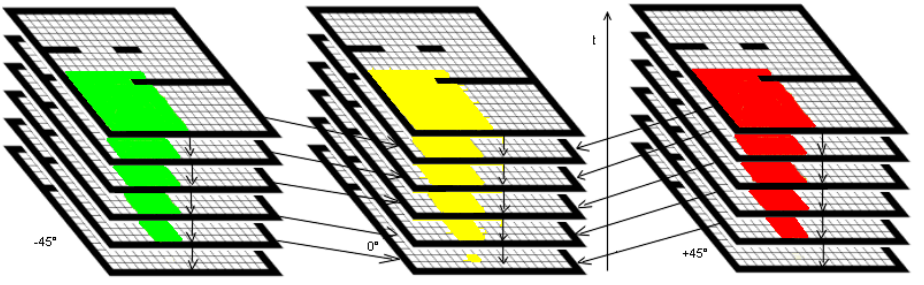

Fig. 3. Dependencies between Attractive Layers at different orientations

on the potential values of the surrounding space-time cells. Iterating this process for all the free cells, the potential surface is built leaving the goal cell with the minimum potential. The potential value has a particular mean: it is the total (minimal) cost needed to reach the goal from the current cell. Because the costs are positive, no other minimum is generated, thus avoiding the well-known problem of the "local minima". The entire trajectory is computed just following the direction of the negated gradient of the surface from the starting point. The path results to be at the minimum valleys of the surface. The robots have different goals and kinematics, hence a potential bowl for each one has to be generated separately. The potential bowl is built only in the free space, avoiding to enter in the $\mathcal{C}$-*Obstacles* areas. Therefore, the robot *Attractive Layer* depends on the *Obstacles Layer*. The last one embeds the Time, thus the potential bowl varies with the time, generating different potential surfaces at different starting time and, consequently, different trajectories. In Fig. 3 is shown the dependencies between Layers and Sublayers of the overall *Attractive Layer*. Each sublayer, at a given robot orientation, depends on the adjacent sublayers at different orientations (aside in the figure), and depends also on the below temporal sublayer.

3.3 The Planning Algorithm

Entailing the layer structure previously described, is now possible to describe the algorithm to compute the robots trajectories. First of all, we have to assign the robots priorities. Up to now, no particular heuristic has been found, thus we assign them casually. The following step is to initialize the cells of the *Obstacles Layer* (full if it belongs to an obstacle, empty otherwise) in each temporal sublayer, with the obstacles distribution known from the Occupancy Map of the workspace. Then, the algorithm computes the *Attractive Layer*, based on the *Obstacles Layer*, of the robot with the highest priority level. Setting the minimum value to the goal cell, it makes the potential bowl to grow while surrounding the obstacles. With the potential surface, it extracts the shortest path, following the negated gradient from the starting cell. For each passing points of the path, then it adds the robot silhouettes, properly oriented, in each temporal sublayer of the *Obstacles Layer*. In this way, the first robot becomes a space-time "obstacle" for the lower priorities robots. The *Obstacles Layer* has a central role in this

phase, because it ensures that the robots avoid each others and the movements do not intersect, even taking into account their real extensions. For this reason it is also called Coordination Layer. The next phase is to repeat the procedure for all the other robots in order of priority. The algorithm terminates when it has computed all the robot trajectories.

4 Experimental Results

The priority planning is not complete: there are problems for which there is a "natural" simple solution, but this type of algorithm is not able to find it. In Fig. 4 (a similar one in [4]) is shown a symmetric example where adopting a priority order (e.g. the green robot passes before the red one) it does not found any solution. Fortunately, for most of the problems we do not have this kind of problems and the algorithm finds a correct solution. In the example of Fig. 5, the

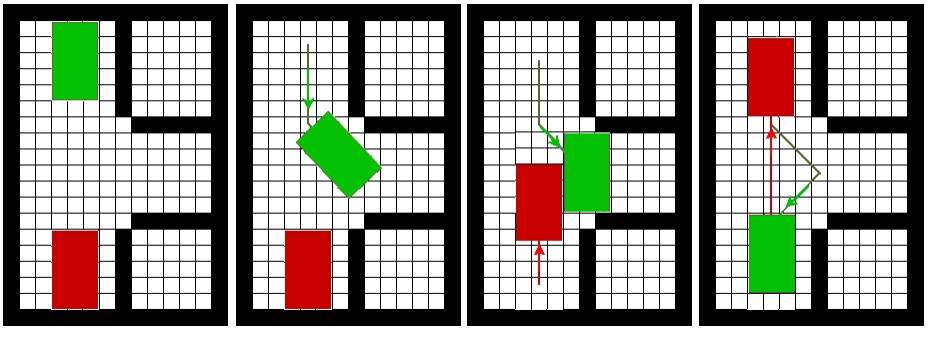

Fig. 4. Counter-example: situation for which the priority planning does not work

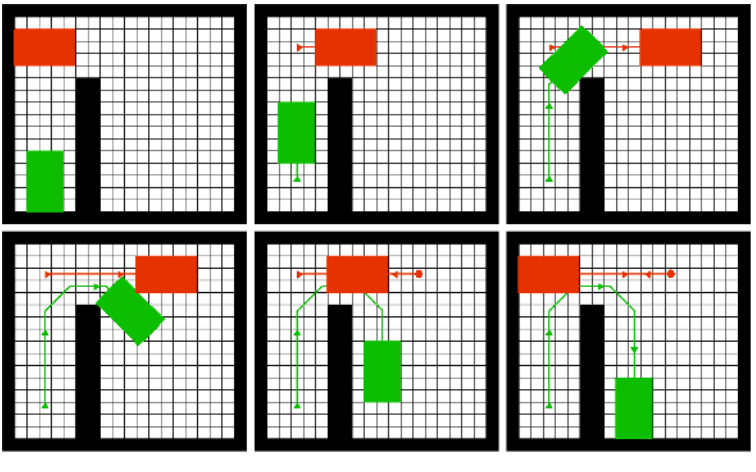

Fig. 5. Clear the way

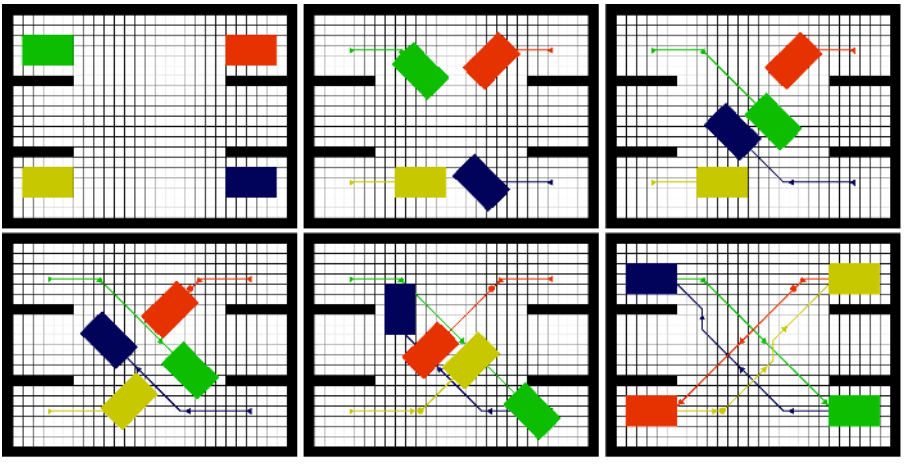

Fig. 6. Crossing robots

red robot has to clear the way to permit the green one to get out of the room, then it gets back to the original position. The following example (Fig. 6) shows a classical problem where the robots trajectories have to cross to exchange their positions in the four corners. All the robots start contemporarily, but the red and yellow robots have to stop to permit the blue and the green robots (with higher priority levels) to pass before them. Then also the red and yellow ones complete their paths.

5 Conclusions

In this paper we have proposed a decoupled and prioritized path planning algorithm for coordinating a Multi-Robot composed by mobile robots using a Multilayered Cellular Automata. One of the main topics is that we face contemporarily with mobile robots have generic shapes and sizes (user defined) and different kinematics. It is based on the Priority Planning approach in the *C-Space-Time* where the Time has been added to the normal *C-Space*.

The Priority Planning is not complete, but it works very well for most of the problems, finding all the collision-free equivalent paths for each robot. The algorithm is also able to manage the robots orientations, avoiding to waste a lot of space during the motion, and permitting to find paths in cluttered workspaces. The trajectories found are smoothed and respect the kinematics constraints and space occupancies of the robots.

References

1. Bandini S., Mauri G., Multilayered cellular automata, *Theoretical Computer Science*, 217 (1999), 99-113
2. Barraquand J., Langlois B., Latombe J. C., Numerical Potential Field Techniques for Robot Path Planning, *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 22, No. 2 (Mar 1992), 224-241

3. Bennewitz M., Burgard W. Coordinating the Motions of Multiple Mobile Robots using a Probabilistic Model, *Proc. of the Int. Symp. on Intelligent Robotic Systems (SIRS)*, (2000)
4. Bennewitz M., Burgard W., Thrun S., Optimizing schedules for prioritized path planning of multi-robot systems, *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Seoul (Korea), (2001)
5. Goles E., Martinez S., Neural and Automata Networks: dynamical behavior and applications, Kluwer Academic Publishers (1990)
6. Jahanbin M. R., Fallside F., Path Planning Using a Wave Simulation Technique in the Configuration Space in Gero J. S., Artificial Intelligence in Engineering: Robotics and Processes, Computational Mechanics Publications (Southampton 1988)
7. Kathib O. Real-time Obstacle Avoidance for Manipulator and Mobile Robots, *Proc. of Int. Conf. on Robotics and Automation* (1985)
8. LaValle S. M., Hutchinson S. A., Optimal Motion Planning for Multiple Robots Having Independent Goals *IEEE Trans. on Robotics and Automation*, Vol. 14, No. 6 (Dec 1998), 912-925
9. Latombe J. C., Robot Motion Planning, Kluwer Academic Publishers, Boston, MA (1991)
10. Lozano-Pérez T., Wesley M. A., An Algorithm for Planning Collision-Free Paths Among Polyhedral Obstacles, *Comm. of the ACM*, Vol. **22**, No. 10, (Oct 1979), 560-570
11. Lozano-Pérez T., Spatial Planning: A Configuration Space Approach, *IEEE Trans. on Computers*, Vol. C-32, No. 2 (Feb 1983), 108-120
12. Sipper M., Evolution of Parallel Cellular Machines - The Cellular Programming Approach, LNCS 1194, Springer (1997)
13. Tzionas P. G., Thanailakis A., Tsalides P. G., Collision-Free Path Planning for a Diamond-Shaped Robot Using Two-Dimensional Cellular Automata, *IEEE Trans. on Robotics and Automation*, Vol. 13, No. 2 (1997), 237-250
14. Warren C., Multiple robot path coordination using artificial potential fields, *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, (1990), 500-505
15. Zelinsky A., Using Path Transforms to Guide the Search for Findpath in 2D, *Int. J. of Robotics Research*, Vol. 13, No. 4 (Aug 1994), 315-325