Spatiotemporal MCA Approach for the Motion Coordination of Heterogeneous MRS

Fabio M. Marchese
Università degli Studi di Milano-Bicocca
Italy

1. Introduction

In this work, a collision-free motion-planning technique for multiple robots based on Multilayered Cellular Automata (MCA) has been studied. What we want to realize is a Coordinator for multi-robot systems which decides the motions of a team of robots while they interact with the environment and react as fast as possible to the dynamical events.
Many authors have proposed different solutions for the Path/Motion Planning problem during the last twenty-five years for single and multiple robots. For example, a solution based on a geometrical description of the environment had been proposed since 1979 (e.g., Lozano-Pérez & Wesley, 1979, Lozano-Pérez, 1983). The motion-planners (path-planners) working on these types of models generate very precise optimal trajectories and they can solve really difficult problems, also taking into account non-holonomic constraints, but they are also very time consuming. To face a real dynamical world with many events, a robot must constantly sense the world and re-plan as fast as possible, according to the newly acquired information. Other authors have developed alternative approaches less precise but more efficient: the Artificial Potential Fields Methods.
In the eighties, Khatib first proposed this method for the real-time collision avoidance problem of a manipulator in a continuous space (Khatib, 1986). Jahanbin and Fallside first introduced a wave propagation algorithm in the Configuration Space C-Space on discrete maps (*Distance Transform*, Jahanbin & Fallside, 1988). In (Barraquand et al., 1992), the authors used the Numerical Potential Field Technique on the C-Space to build a generalized Voronoi Diagram. Zelinsky extended the *Distance Transform* to the *Path Transform* (Zelinsky, 1994). Tzionas et al. in (Tzionas et al., 1997) described an algorithm for a diamond-shaped holonomic robot in a static environment where they let a CA build a Voronoi Diagram. In (Warren, 1990) the coordination of robots is proposed using a discretized 3D C-Space-Time (2D workspace plus Time) for robots with the same shape (only square and circle) and a quite simple kinematics (translation only). LaValle in (LaValle, 1998) applies the concepts of the Game Theory and multi-objective optimization to the centralized and decoupled planning. A solution in the C-Space-Time is proposed in (Bennewitz, 2001), where the authors use a decoupled and prioritized path planning in which they repeatedly reorder the robots to try to find a solution. It can be proven that these approaches are not complete.
In this work, we want to design a motion coordinator for a set of heterogeneous mobile robot (different shapes and kinematics), able to determine the motions of the mobile agents

Source: Recent Advances in Multi-Robot Systems, Book edited by: Aleksandar Lazinica, ISBN 978-3-902613-24-0, pp. 326, May 2008, I-Tech Education and Publishing, Vienna, Austria

in order to avoid collisions with obstacles and with other robots. We have adopted Cellular Automata as formalism for merging a grid model of the world (Occupancy Grid) with the C-Space-Time of multiple robots and Numerical (Artificial) Potential Field Methods, with the purpose to give a simple and fast solution for the motion-planning problem for multiple mobile robots, in particular for robots with different shapes and kinematics. This method uses a directional (anisotropic) propagation of distance values between adjacent automata to build a potential hypersurface embedded in a 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 poses for each robot C-Space-Time.

2. Context: a M^{n}RS self-similar layered architecture

The area of the cooperative/distributed robotics moved its first steps in the 80's, and since then it has received ever increasing attention, especially in the last decade, involving many application domains. A MultiRobot System is characterized by a set of robots working in the same environment, interacting between them inside the system and toward the outside of the system with the external environment, and last but not least, sharing their resources to achieve a general common task. With respect to an equivalent single robot achieving the same task, a MRS improves the robustness and the reliability thanks to its modularization. Many are the application domains where MRSs are involved: from service robotics to planets exploration. In (Dudek et al., 1996) an analysis and a classification of the typical tasks for MRS have been proposed. Taxonomy of robots collectives is based on the following main dimensions: size, composition, communication, re-configurability and computation. In the editorial (Arai et al., 2002) the authors identify seven important topics of the MRS research area: biological inspirations, communication, architectures and control, localization/mapping/exploration, object transport and manipulation, motion coordination, and reconfigurable robots. A survey of the area has been proposed more recently in (Farinelli et al., 2004). In this paper, an interesting classification about MRS architectures is described. It is mainly based on two primary types of features: coordination dimensions and system dimensions. The first is a layered taxonomy structured on four levels: Cooperation level, Knowledge level, Coordination level and Organization level. The system dimensions are subdivided into four groups: Communication, Team composition, System architecture, Team size.
On the basis of this taxonomy, we can give a classification for our coordination architecture: cooperative (Cooperation level); unaware (Knowledge level); strongly coordinated (Coordination level); strongly centralized (Organization level). On the standpoint of the system dimensions: direct communication, throughout a central dispatcher; heterogeneous (Team composition); deliberative (System architecture); small-medium Team size. The framework can be intended as *cooperative* because the robots reaching their goals accomplish a single global task specified at a higher level of abstraction. At the *Organization level*, it is strongly centralized: there is a leader robot (or an external supervisor, but it is only a deployment consideration) that is omniscient with respect to its team mates, and which organizes, synchronizes and controls the other robots (*strongly coordinate*). The other team mates do not have any knowledge about each other (*unaware*). In a strongly centralized architecture, the central unit is responsible for taking any decision, and the peripheral units operate consequently. This must not be intended as a severe restriction to the autonomy of

the single unit: every robot can decide how to realize the goal/command that the leader provides for it. Besides this, the autonomy restriction will also depend on the type of commands: for example, the central unit can provide a gross trajectory, which will be refined by the single agent. *Heterogeneity*: the task is to design an architecture operating independently from the robot characteristics.

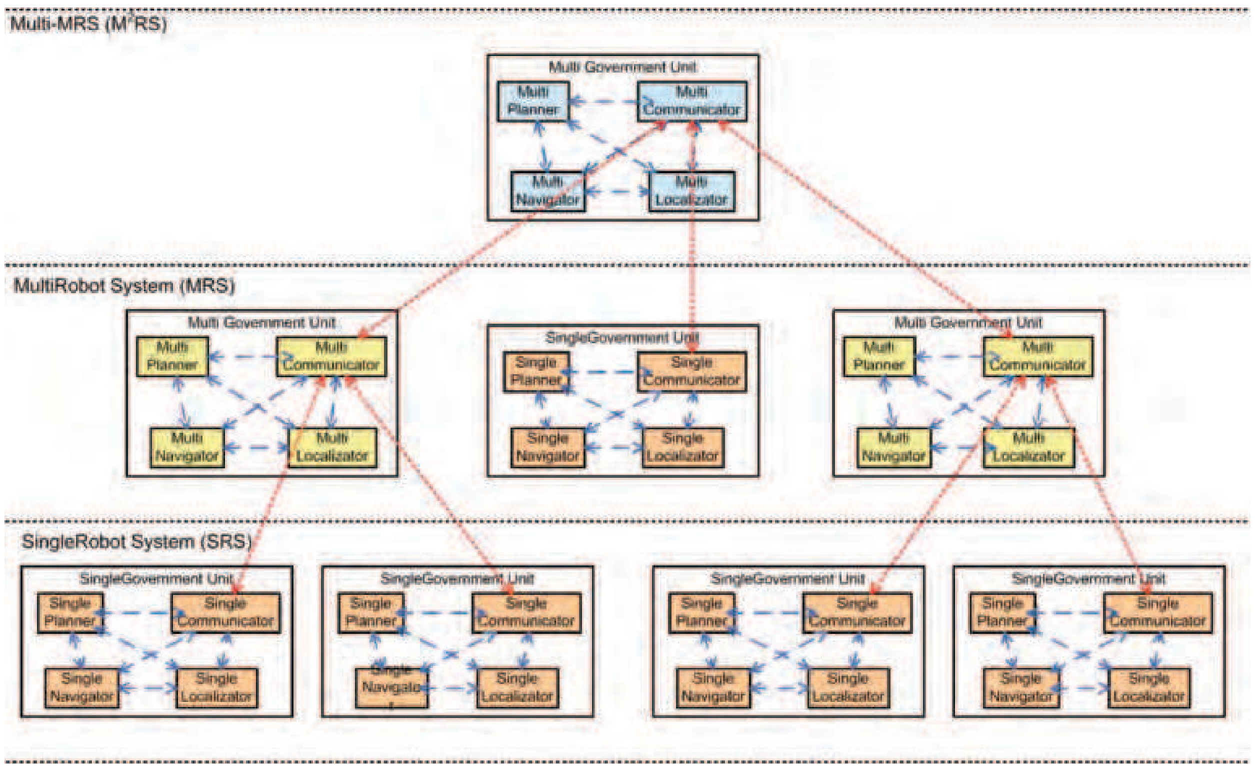


Figure 1. An example of Multi-MRS at the second level

The proposed approach is coherent with a MRS architecture (briefly described in this paragraph), where the coordination system is just a subpart of it. It is basically organized into self-similar objects, where essentially the MRS is reified as an object similar to the single robot composing it. The architecture can be further extended to the coordination of groups of teams (let's say Multi-MRS or M^{2}RS) extending the number of layers, but not changing the significance: each M^{n}RS is an object similar to all the other objects at the lower levels and to the single robots. Each level is an aggregation of entities of a lower order, till a single robot system {M^{0}RS, MRS, M^{2}RS, … M^{n-1}RS}, where M^{0}RS = SRS (Single-Robot System).
The framework has to be able to control a group of robots with quite different hardware and software devices. In particular, from the point of view of the motion control, the robots are quite different in sizes, shapes and kinematics.
Our task is to define a framework able to work independently from the single robot hardware and software characteristics. We have designed a layered architecture (an example in Fig. 1), called Multi-Robot Layered architecture (MRL architecture). The two lower layers concerns to the multi-robot level (MRS) and the single-robot level (SRS). It is interesting to note that the entities inside the layers have the same structure and expose similar functionalities/modules. In this context, we are considering the multirobot as a single (abstract) separated entity which performs its own functionalities and to which the single robots are interfaced throughout a communication system. From the conceptual point

of view, the multirobot has similar functionalities as the single robot, for example it navigates, communicates, and so on, as the single robot does. The differences arise at the implementation/deployment level.

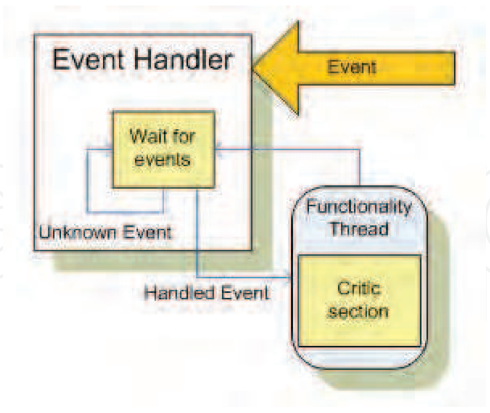


Figure 2. The core of a module: the event-handler

The concept can be abstracted further to higher level where a M^{n}RS is seen as an entity with the same role of the others, thus realizing a self-similar structure, with similar modules which can interact with each other even at different levels, as shown in Fig. 1. The architecture has been thought as a concurrent event driven system. The modules inside each entity (single or multi) are synchronized with the other modules using events. The basic structure of a module is essentially a thread controlling an event-handler (Fig. 2). The main task of the thread is to wait for a set of events. When one of the events is signaled, it triggers a separate sub-thread that realizes the corresponding functionality and immediately returns to wait for other events. Unexpected events are simply ignored. In this way, we realize a simple ad efficient mechanism that it is not blocked during the execution of a functionality, and thus it is independent to how much it is complex and how long it takes to be completed. Therefore, every event receives an immediate response: an important characteristic for a real-time system that permits to handle fast dynamics of the environment.
Besides architectural issues, which are mainly concerned with the design and then the handling of multi-threads/concurrent systems (a more detailed description in (Marchese, 2007)), the real-time coordination of the motion of n bodies is the major problem.

3. The Multirobot Motion-Planner Coordination System

3.1 Problem statement: from Motion-Planning to Spatiotemporal MCA

The Multirobot Motion Planner is "just" a module (*Multi-Planner*) inside the entities at the MRS level above described.
To solve the Motion-Planning Problem we need some essential features: a world representation, one or more motion models of the robots, a substrate on which the two previous entities interact to generate a (multi-)plan.
There is a very large variety of world models in literature able to describe the interaction between autonomous agents and their environment. A very well-known model and probably the most important one is the Configuration Space (Lozano-Pérez, 1983; Latombe, 1991). Let us consider a simple rigid body $\mathcal{R}$ with a generic shape, a finite extension and an orientation (a preferential direction of movement). The C-Space $\mathcal{C}$ of a rigid body is the set of all its configurations $\mathbf{q}$ (i.e. all its poses). $\mathcal{R}$ moves in a workspace $\mathcal{W}$

embedded in a physical n-dimensional space (we do not necessarily consider the Euclidean Space $\mathcal{R}^n$: also n-D manifold are admissible (Latombe, 1991)). In the workspace, a finite number of obstacles $\mathcal{O}_i$ are placed. A configuration **q** of $\mathcal{R}$ is a compact representation of all the points of its shape. Because the robot is a rigid body, any configuration **q** corresponds to a set of n independent parameters, corresponding to the n DOF of the robot. If the robot can freely translate and rotate on a 2D surface, then its C-Space is a 3D manifold $\mathbb{R}^2 \times \mathbb{S}^1$, where $\mathbb{S}^1$ is the unit circle **SO**(2). $\mathcal{R}(\mathbf{q})$ is the set of points occupied by the robot in $\mathcal{W}$ at the configuration **q**. Every obstacle $\mathcal{O}_i$ is mapped in the C-Space as a set of configuration $\mathcal{CO}_i$ (*C-Obstacle*): $\mathcal{CO}_i = \{\mathbf{q} \in \mathcal{C} \,/\, \mathcal{R}(\mathbf{q}) \cap \mathcal{O}_i \neq 0\}$. In other words, a C-Obstacle is the set of configuration **q** at which the robot collides with the obstacle. The free space is the subset of points of $\mathcal{C}$ where the robot and the obstacles do not have any intersection: $\mathcal{C}_{\text{free}} = \mathcal{C} - \cup_i \mathcal{CO}_i$. A collision-free path between two configuration $\mathbf{q}_S$ (starting pose) and $\mathbf{q}_G$ (goal pose) is any continuous map $\tau : [0, 1] \rightarrow \mathcal{C}_{\text{free}}$ with $t(0) = \mathbf{q}_S$ and $t(1) = \mathbf{q}_G$.
Using a regular decomposition in (hyper-)cells, the C-Space can be easily represented using a n-D bitmap $\mathcal{GC}$ (*C-Space bitmap*). The *C-Potential* is a function $\mathbf{U}(\mathbf{q})$ defined over the C-Space that *drives* the robot through the sequence of configuration points to reach the goal pose (Barraquand et al., 1992). These definitions can be extended to an arbitrary number of robots with an arbitrary number of DOF working in the same space (LaValle & Hutchinson, 1998).
Another feature needed to plan is the substrate where to make the environment and the robots models to interact. We use Cellular Automata (CA) for this purpose. CA are automata distributed over the cells of a Cellular Space $\mathbb{Z}^n$ (a regular lattice) with transition functions invariant under translation (Goles & Martinez, 1990): $f_c(\cdot) = f(\cdot)$, $\forall\, c \in \mathbb{Z}^n$, where $f(\cdot)$: $\mathbf{S}^{|\mathbf{A}_0|} \rightarrow \mathbf{S}$, where c is the coordinate vector identifying a cell, **S** is the set of states of a FSM, $\mathbf{A}_0$ is the set of arcs outgoing from a cell towards the neighbors and $f(\cdot)$ is the transition function. This definition introduces an n-D grid of hyper-cells connected to the surrounding cells with the arcs $\mathbf{A}_0$. In each cell it is embedded an automaton, and the automaton of a cell is equal to the automaton of any other cell. The state $s \in \mathbf{S}$ of the automaton evolves (function $f(\cdot)$) on the basis of the states of the neighbors (arcs $\mathbf{A}_0$). The global behavior of the whole cellular automata is influenced by the local rules (transition function) and the updating rules, i.e., it is governed by the chronological sequence of cells states updating (e.g. Synchronous, Asynchronous, Block-Sequential, etc.).
It is possible to organize the CA in layers of homogenous automata, but with different transition functions across the layers, thus having a Multilayered Cellular Automata (MCA). The mapping between the Robot Motion-Planning Problem and MCA is quite simple: every "pixel" of the C-Space bitmap $\mathcal{GC}$, corresponding to a configuration **q,** is a hyper-cell of an n-D CA. The state inside the cell represents a potential value and it contributes to build the *C-Potential* $\mathbf{U}(\mathbf{q})$ through a diffusion mechanism over the neighbors. The *C*-Potential is a potential hyper-surface defined on the n-D space (it is embedded in an n+1-D space). The trajectories are found following the minimum valley of this hyper-surface.
Extending the concept, we associate a vector of attributes (a state vector instead of a single state) to every cell. Each state vector depends on the state vectors of the cells in the neighborhood. An alternative interpretation is the following: this is a Multilayered Cellular Automaton, where each layer deals with a subset of components of the state vector. Each subset is evaluated in a single layer and it depends on the same attribute of the neighbor

cells in the same layer, but it also depends on the values of the corresponding cells in other layers. In the following sections, we are going to describe the layers and their behaviors.

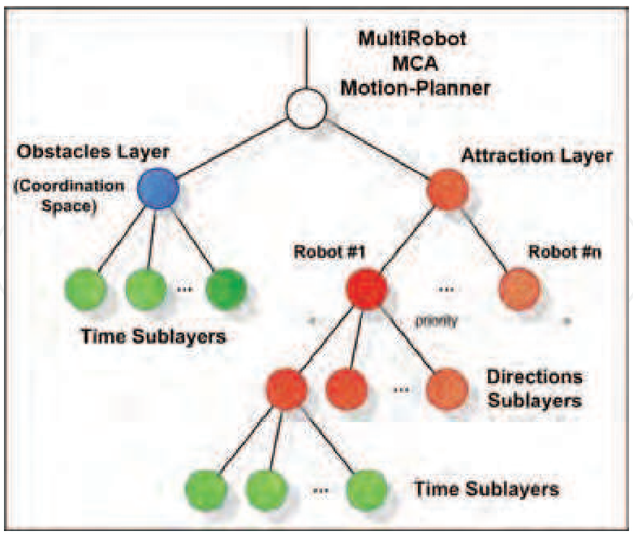


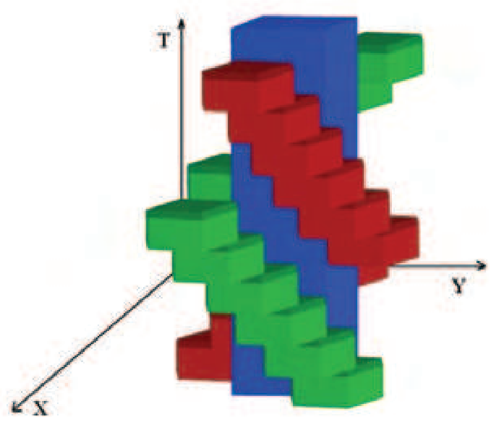
Figure 3. MultiLayered Motion-Planner Architecture

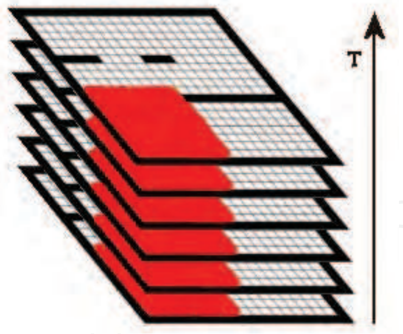
3.2 Multilayered/Multidimensional Architecture

All the planning system is based on a layered architecture, as shown in Fig. 3, where the layers and sub-layers structure and their dependencies are defined.
There are two main layers: the *Obstacles Layer* and the *Attraction Layer*. Each layer is subdivided into more sub-layers: the *Obstacles L.* has 3 dimensions (2 for the workspace (X, Y) and 1 for the time), while the *Attraction L.* has up to 5 dimensions (1 for the robots, 3 for the robots C-Spaces (X, Y, θ) and 1 for the time). The *Obstacles L.* conceptually depends on the outside environment by means of an obstacle detector or a world modeler. Its sub-layers should 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 re-plan as needed.

3.3 The Obstacles Layer

The main role of the *Obstacles L.* is to map the obstacles of the environment and to dynamically create a repulsive force that keeps the robots away from them.
Up to now, only static obstacles are considered (e.g. walls). For the single robot, the other robots are seen as moving obstacles, with unknown and unpredictable motions. We are considering a centralized motion-planner/coordinator, which can decide the timed trajectories of all the supervised robots. Thanks to this information, the planner considers the silhouette of a robot as an obstacle for the other robots. In this work, we introduce a discretized version of the C-Space-Time (Erdmann & Lozano-Pérez, 1986; Warren, 1990).
With the introduction of the Time axis, a static obstacle becomes a hyper-obstacle extended along the time direction in the Spacetime 4D space. In Fig. 4.a a discretized Spacetime is shown, where the blue object is a static obstacle extended vertically along the time. The robots moving in the environment become "static" hyper-obstacles in the Spacetime, having different poses in each time slice (red and green objects in Fig. 4.a).





a) b)

Figure 4. Discretized C-Space-Time and C-Obstacles: a) both dynamical (red & green) and static (blue) objects becomes static hyper-objects in the Spacetime; b) robot silhouette at different time slice

We can still call them as C-Obstacles, remembering that they have been extended in the time dimension. In the case of the *Obstacles Layer*, we have 3D Spacetime: $\mathbb{R}^2 \times \mathbb{R}$, where $\mathbb{R}^2$ is the planar workspace. The overall "static" spacetime obstacle, representing one robot, is the composition of all the robot silhouettes in each time slice (Fig. 4.b). If we consider all the robots moving at the same speed, the time step (slice) is the time needed by a robot to move to an adjacent cell (the robots motions are synchronized). This layer is also called Coordination Layer (/Space) because the motions of all the robots are coordinated in this layer (see § 3.5).

3.4 The Attraction Layer

The *Attraction 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). To pass from one cell to another one, the robot can follow different paths, combining different atomic moves, such as *straight_forward* move, *diagonal* move, *rotation*, and so on. Each move has its own cost (always positive); the entire set of admissible movements defines the robot kinematics. The kinematics together with the silhouette is the motion model of the robot (the third "ingredient" needed to make planning). This planner is able to handle different types of robots at the same time, with different shapes, sizes and kinematics (e.g. car-like kinematics, holonomic, Dubins' car, etc.), providing the proper trajectory for each one. The moves are space-temporal, i.e. every time the robot moves in the Space, it also moves in the Time. Even when the robot stands in the same place, it moves (vertically) in the Spacetime (also this particular move has its own cost). 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 Spacetime. The hyper-surface is defined on a 4D space: $\mathbb{R}^2 \times \mathbf{SO}(2) \times \mathbb{R}$, where $\mathbb{R}^2 \times \mathbf{SO}(2)$ is the C-Space. 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 computation is performed by the automata in the spacetime cell, which compute the potential value depending on the potential values of the surrounding spacetime cells. Iterating this process for all the free cells (avoiding the cells occupied by a hyper obstacle), the potential surface is built leaving the goal cell with the minimum potential. The potential value of a cell has a particular mean: it is the total (minimal) cost needed to reach the goal from the current cell. Since the costs are positive, the bowl always grows with increasing values and 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. Every robot has its own goal, kinematics and shape, hence a potential bowl has to be generated separately for each one. Since the potential bowl is built only in the free space, the robot *Attraction L.* depends on the contents of the *Obstacles L.*

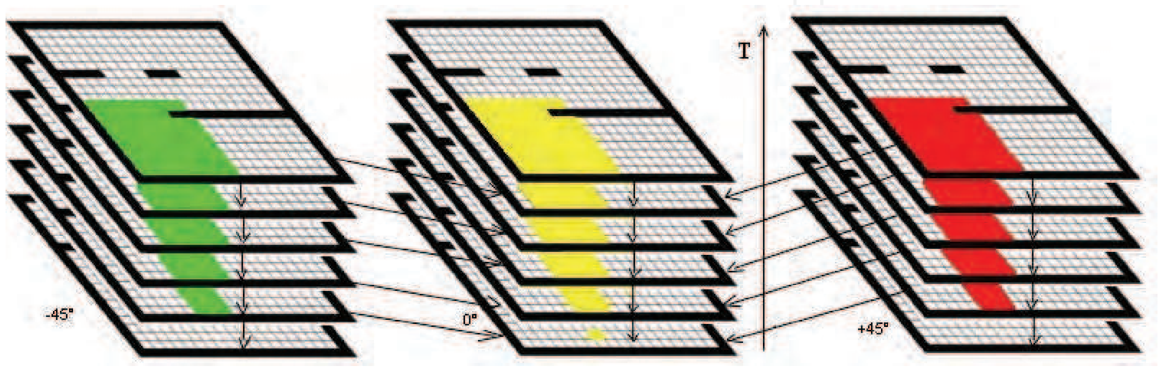


Figure 5. Dependencies between Attraction Layers at different orientations

The last one includes the Time, thus the potential bowl varies from time to time, generating different potential surfaces at different starting time and consequently, different trajectories depending on the time of arrival. In Fig. 5 is shown the dependencies between Layers and Sub-layers of the overall *Attraction Layer*. Each sub-layer, at a given robot orientation, depends on the adjacent sub-layers at different orientations (aside in the figure), and depends also on the below temporal sub-layer.

3.5 The Planning Algorithm

Entailing the layer structure previously described, it is now possible to introduce the algorithm that computes the robots movements.

To avoid the burden of the complexity of the problem using a centralized solution (LaValle & Hutchinson, 1998), we have adopted a Priority Planning approach. Therefore, the first step is to establish the priority level of every robot. Up to now, there is no evident heuristic to be used in this phase and the assignment is defined by the user or casually. The following step is to initialize every temporal sub-layer with the obstacle distribution known from the map of the workspace (a simple occupancy grid). Then, the algorithm computes the *Attraction L.* (potential bowl) of the robot with the highest priority level, also considering the information of the *Obstacles L.* Setting the minimum potential value to the goal cell makes the potential bowl grow while surrounding the obstacles (green cells in Fig. 6).

Once the potential surface is known, the shortest path is extracted, following the negated gradient from the starting cell. For each passing point, then it adds the robot silhouettes, properly oriented, in temporal sub-layer of the *Obstacles L.* In this way, the first robot becomes a spacetime obstacle for the successive robots having lower priorities. The *Obstacles Layer* plays an important role in this phase: it ensures that the robots do not collide, even taking into account their real extensions. For this reason, in this context it is also called the Coordination Space. The next phase is to repeat the procedure for the second robot, starting to build its *Attraction L.* on the base of the modified *Obstacles L.*, and then adding to it the sequence of its silhouettes along the spacetime trajectory. The algorithm terminates when it has computed all the robots trajectories.

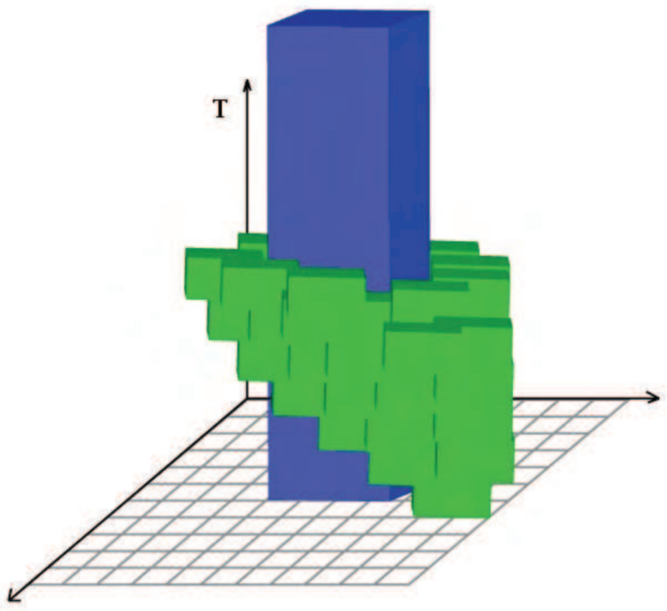


Figure 6. Spacetime Potential bowl growing around a static obstacle

4. Simulation Results

The first consideration is that the priority planning is not complete: there are situations for which there is a solution, but this type of algorithm is not able to find it. In Fig. 7 a simple counterexample is shown (a similar one in (Bennewitz, 2001)) of this type of problems with a simple solution. Adopting a specific priority order (e.g., the green robot moves before the red one) there is no solution. Because this problem is symmetric (the world is symmetric with respect to a horizontal axis, the two robots have the same shape and kinematics), even swapping the priority order (e.g. the red one passes before the green robot) there is still not a solution with a priority schema. Fortunately, for most of the situations we do not have this type of problem and the algorithm can easily find a solution.
In the example of Fig. 8, the red robot has to free the way to permit the green one to get out of the room, and then it goes back to the original position. This is an interesting situation, where a robot, which should stand without moving, is enforced to move by the coordinator to free the passage. The triangles represent the kinematic center and the robot orientation.
The following example (Fig. 9) shows a classical problem where the robots trajectories have to cross to exchange their positions in the four corners. In the middle of the environment there is a conflict zone (interference zone) where all the robots have to pass. All the robots start contemporarily, but the red and yellow robots have to stop and wait for the blue and the green ones (with higher priority values) to pass and exit from the conflict zone. Then, also the red and yellow ones can complete their motions.

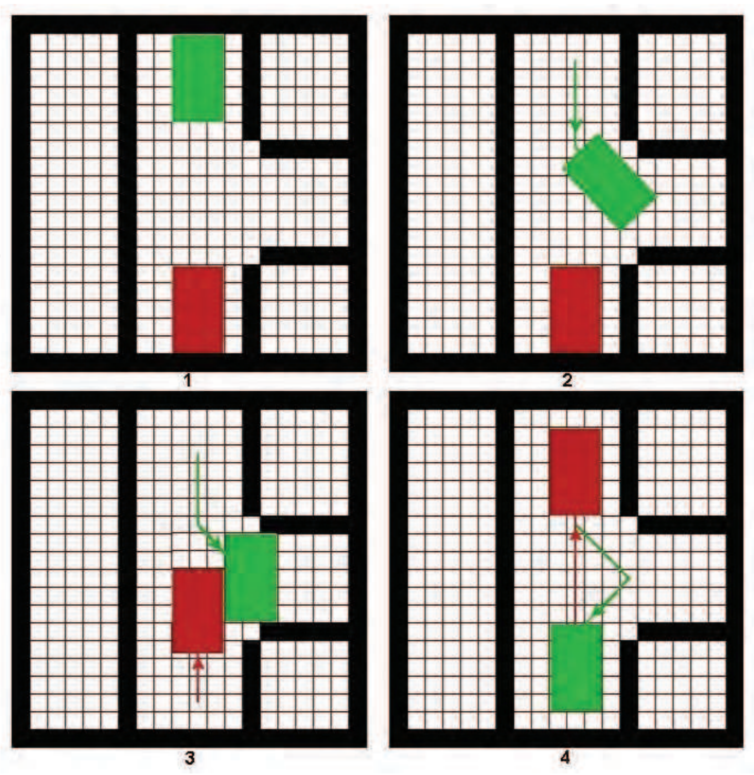


Figure 7. Counterexample: an obvious solution of a problem for which the priority planning fails

In the example of Fig. 10, four robots have to permutate (right rotation) their docking poses. Because the green robot has the highest priority, it enforces the other robots to wait before entering in the new stations.
The example of Fig. 11 concerns an O-ring shaped robot that closes the way to a second robot. The O-ring robot has its kinematics center over a static obstacle, thus it can only rotate to free the passage to the other robot.
In Fig. 12, two robots with unusual shapes and different kinematics (holonomic the green one, car-like the red one) swap their positions between two rooms passing through a narrow door. The silhouettes of the two "robots" are the projections on the plane of objects like a "Truck" and a "Portainer crane" moving on wheels. The task is to make passing the truck under (between the uprights) the crane. This is a very unusual example of planning for a vehicle with a silhouette composed by two disjoined parts and where a second object passes over the kinematic center of the first.
All the times reported have been measured on a PC Intel Core 2 Duo CPU 2.20 GHz. In parenthesis it is also indicated the number of hyper-cells composing the entire C-Space-Time.

5. Conclusions

In this work we have proposed a new version of a decoupled and prioritized motion-planning algorithm for the coordination of a Multi-Robot composed by mobile robots. This Motion-Planner is a single module of a wider multilayered architecture for the coordination of MRS. Using a Multilayered Cellular Automata as paradigm for a joined space representation and planning technique, we face contemporarily mobile robots having generic shapes and sizes. It is based on the Priority Planning approach (decoupled 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 situations, finding all the collision-free equivalent paths for each robot, while simplifying the complexity of the problem. Differently from most of the other planners, the algorithm is also able to handle the robots orientations and robots with different kinematics. This property avoids wasting a lot of space during the motion, and permits to find paths in cluttered workspaces. The trajectories found are smoothed and respect the kinematics constraints of each robot.

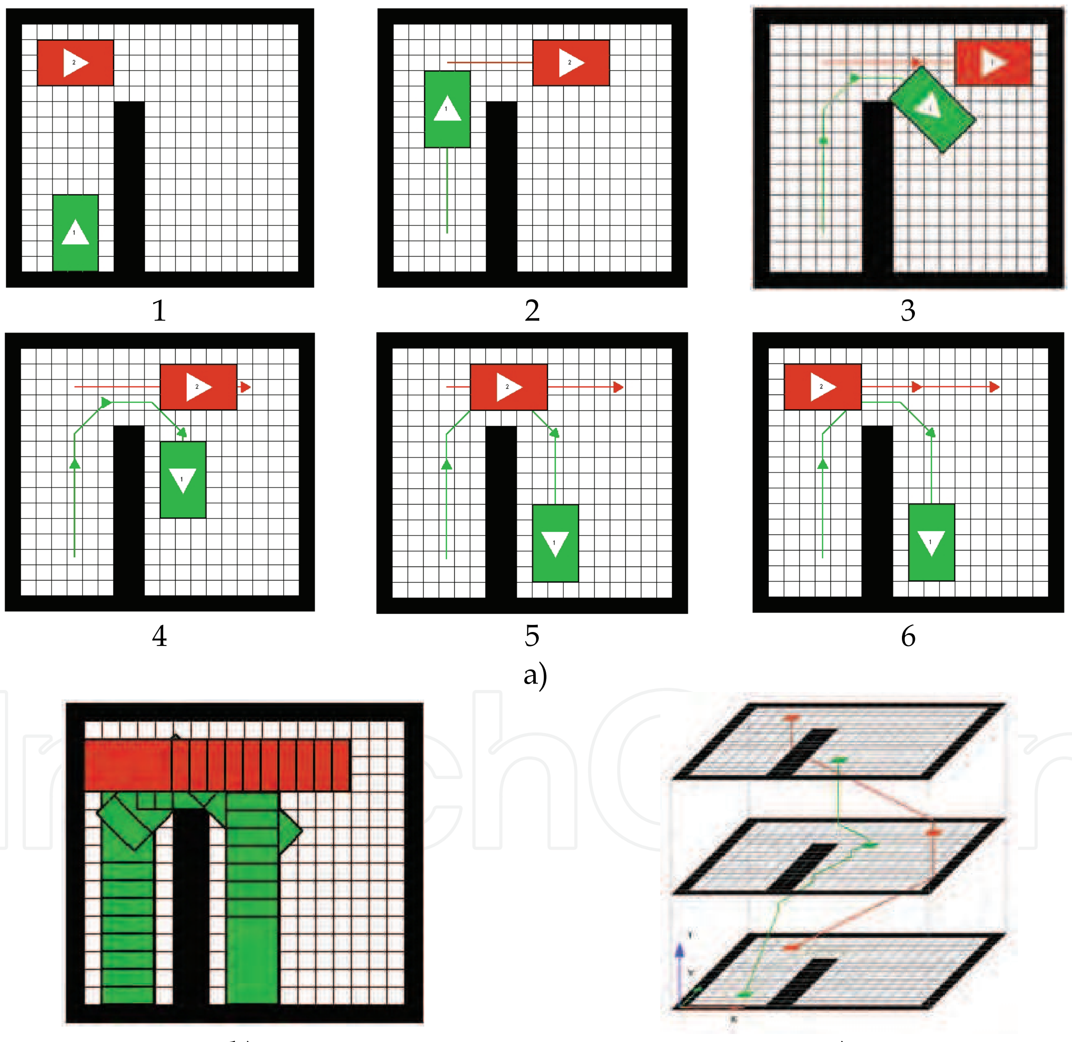


Figure 8. Robot clearing the way: a) snapshots sequence; b) overall movements; c) C-Space-Time movements (planning time: 0.19 s, 259’200 cells)

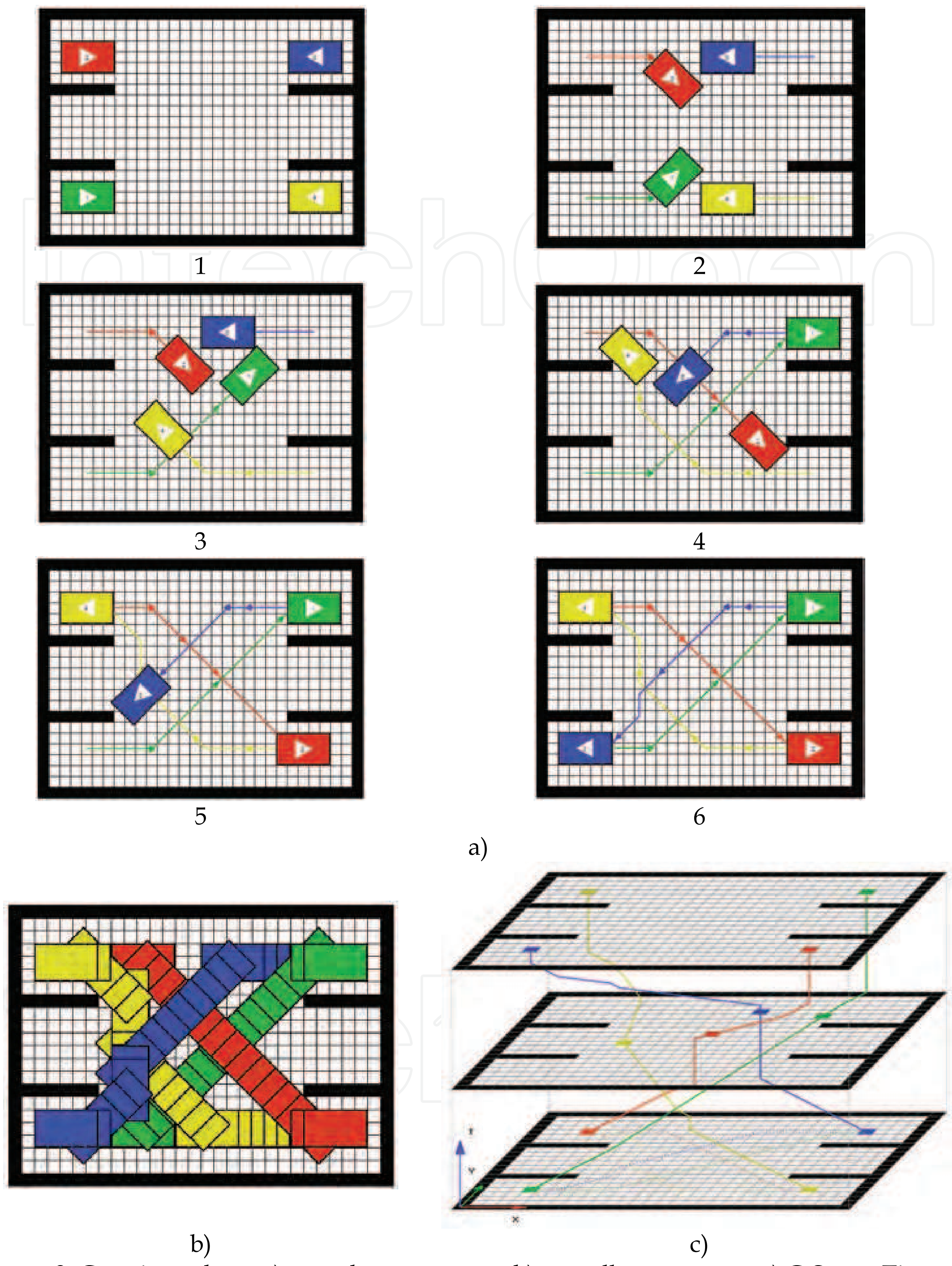
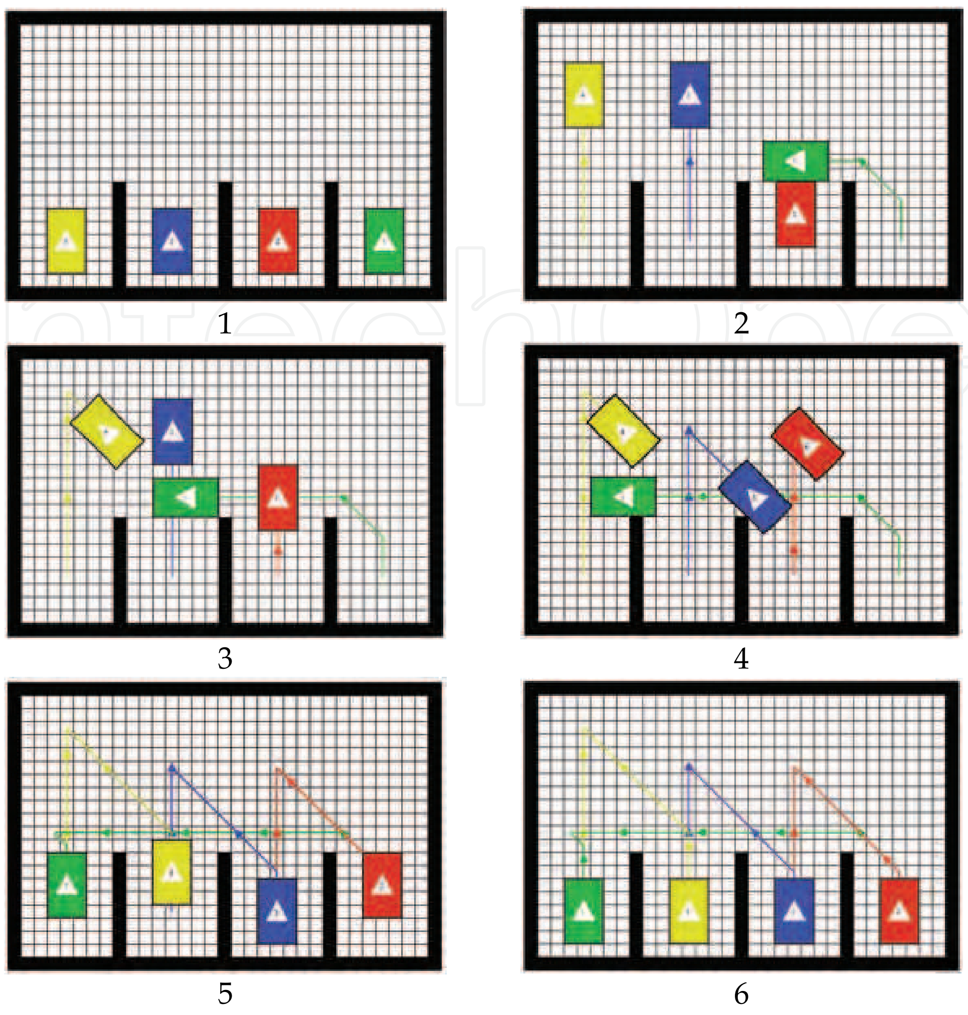
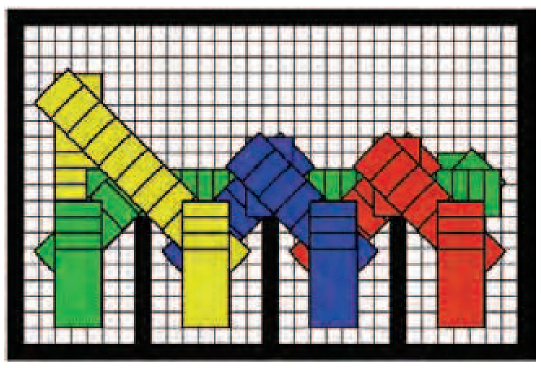
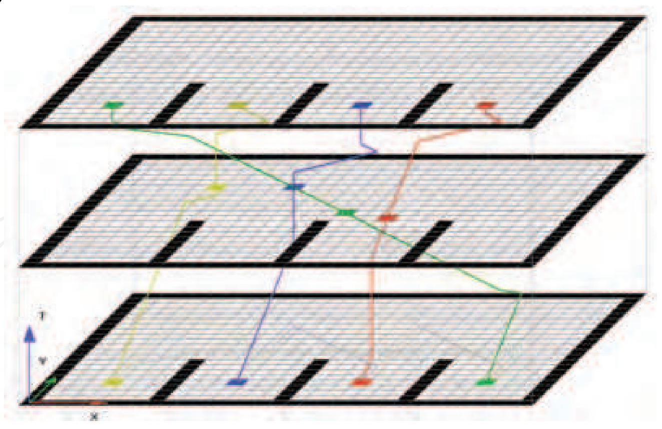


Figure 9. Crossing robots: a) snapshots sequence; b) overall movements; c) C-Space-Time movements (planning time: 1.26 s, 897’600 cells)



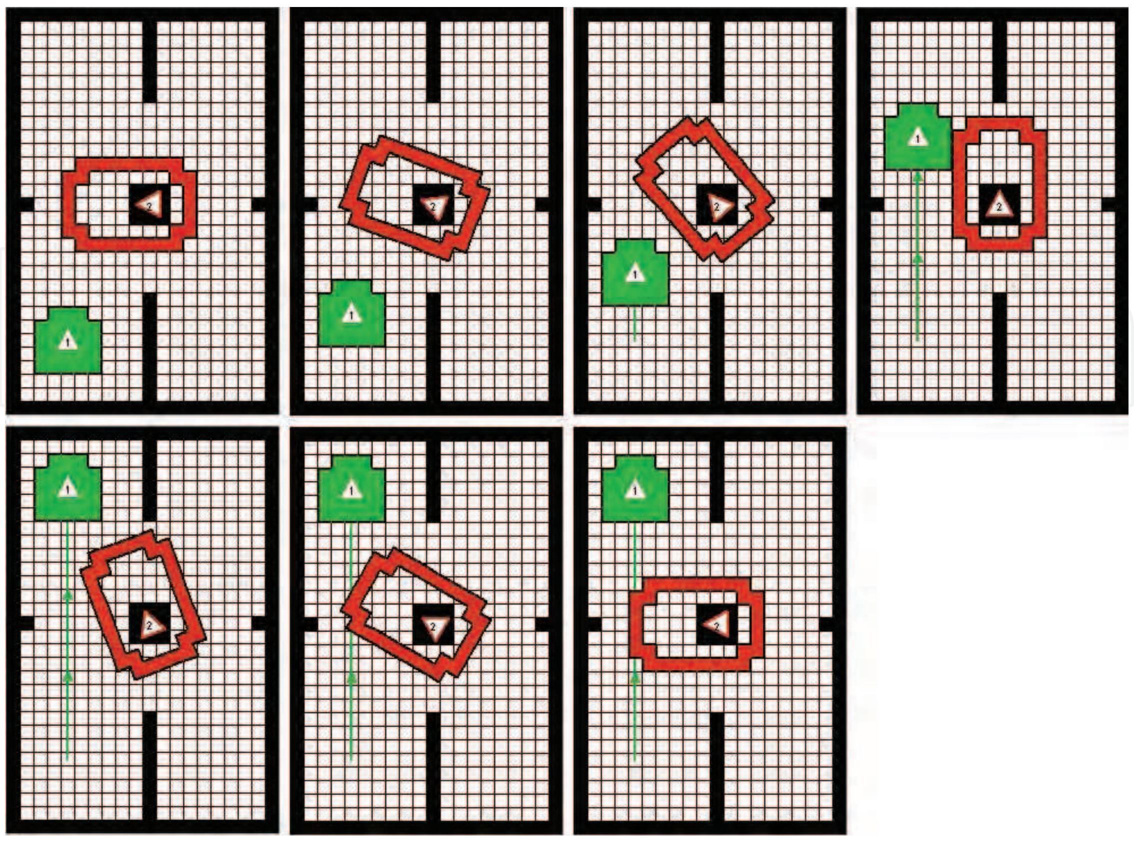
a)



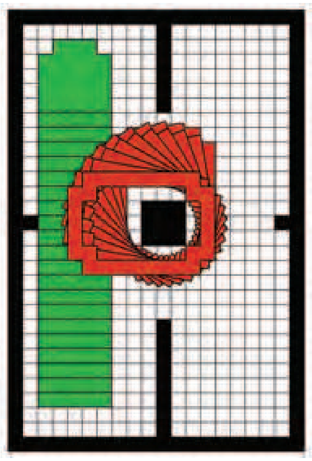


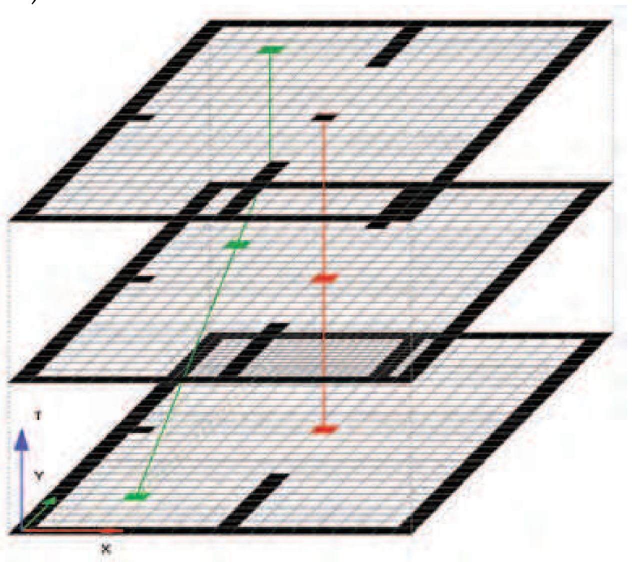
b) c)

Figure 10. Circular permutation: a) snapshots sequence; b) overall movements; c) C-Space-Time movements (planning time: 3.03 s, 987’360 cells)



a)





b) c)

Figure 11. O-Ring example: a) snapshots sequence; b) overall movements; c) C-Space-Time movements (planning time: 0.16 s, 1’554’000 cells)

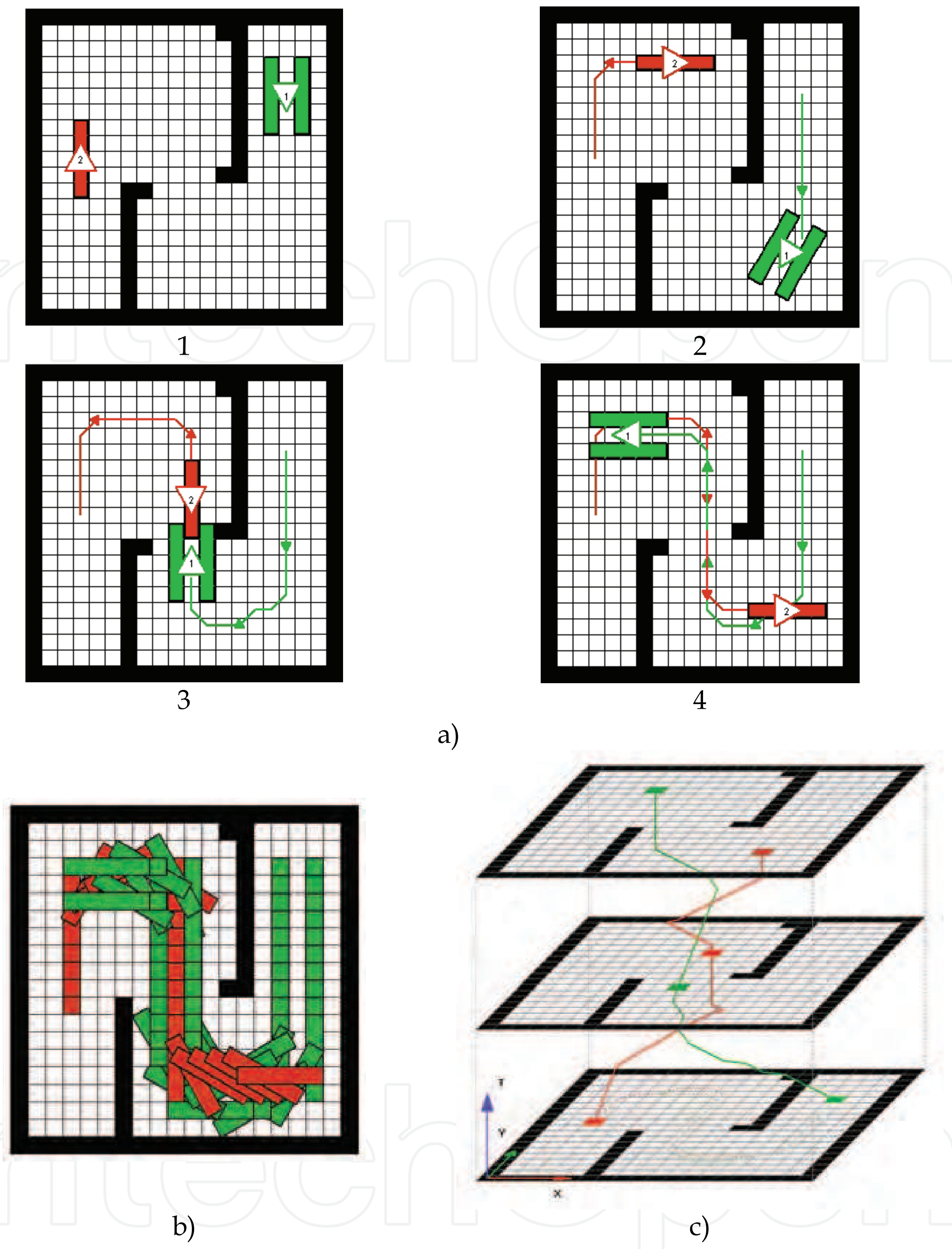


Figure 12. Truck & Portainer Crane example: a) snapshots sequence (crane in green, truck in red); b) overall movements; c) C-Space-Time movements (planning time: 0.33 s, 416’000 cells)

6. References

Arai, T.; Pagello, E. & Parker, L.E. (2002). Editorial: Advances in Multi-Robot Systems, *IEEE Trans. on Robotics and Automation*, Vol. 18, No. 5, Oct. 2002, 655-661, ISSN=1042-296X

Barraquand, J.; Langlois, B. & Latombe, J. C. (1992). Numerical potential field techniques for robot path planning, *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 22, No. 2, 224-241, ISSN:0018-9472

Bennewitz, M.; Burgard, W. & Thrun, S. (2001). Optimizing schedules for prioritized path planning of multi-robot systems, *Proceedings of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 271-276, ISSN:1050-4729, Seoul (Korea), May 2001

Dudek, G.; Jenkin, M.; Milios, E. & Wilkes D. (1996). A taxonomy for multiagent robotics, *Autonomous Robots*, Vol. 3, 375-397

Erdmann M. & Lozano-Pérez, T. (1986). On multiple moving obstacles, Proceedings of IEEE International Conference on Robotics and Automation, pp. 1419-1424, San Francisco, April, 1986

Farinelli, A.; Iocchi, L. & Nardi, D. (2004). Multirobot systems: a classification focused on coordination, *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, Vol. 34, No. 5, Oct. 2004, 2015-2028, ISSN: 1083-4419

Goles, E. & Martinez, S. (1990). *Neural and Automata Networks: dynamical behavior and applications*, Kluwer Academic Publishers, ISBN:0-7923-0632-5, Norwell, MA, USA

Jahanbin, M. R. & Fallside, F. (1988). Path planning using a wave simulation technique in the configuration space, In: *Artificial Intelligence in Engineering: Robotics and Processes*, J. Gero S. (Ed.), Computational Mechanics Inc., ISBN:0-931215-98-6, Billerica, MA, USA

Kathib, O. (1986). Real-time obstacle avoidance for manipulators and mobile robots, *Int. J. of Robotics Research*, Vol. 5, No. 1, 90-98

Latombe, J. C. (1991). *Robot Motion Planning*, Kluwer Academic Publishers, ISBN: 978-0792392064, Boston, MA

LaValle, S. M. & Hutchinson, S. A. (1998). Optimal motion planning for multiple robots having independent goals, *IEEE Trans. on Robotics and Automation*, Vol. 14, No. 6, 912-925, ISSN:1042-296X

Lozano-Pérez, T. (1983). Spatial planning: A configuration space approach, *IEEE Trans. on Computers*, Vol. C-32, No. 2, 108-120, ISSN:0018-9340

Lozano-Pérez, T. & Wesley, M. A. (1979). An algorithm for planning collision-free paths among polyhedral obstacles, *Comm. of the ACM*, Vol. 22, No. 10, 560-570, ISSN:0001-0782

Marchese, F. M. (2007). An Architecture for MultiRobot Motion Coordination, *Proceeding of Robotics and Applications and Telematics (RA 2007)*, 352-357, ISBN: 978-0-88986-685-0, Würzburg, Germany, August 2007

Tzionas, P. G.; Thanailakis, A. & Tsalides, P. G. (1997). Collision-free path planning for a diamond-shaped robot using two-dimensional cellular automata, *IEEE Trans. on Robotics and Automation*, Vol. 13, No. 2, 237-250, ISSN:1042-296X

Warren, C. (1990). Multiple robot path coordination using artificial potential fields, *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 500-505, May 1990

Zelinsky, A. (1994). Using path transforms to guide the search for findpath in 2d, Int. J. of Robotics Research, Vol. 13, No. 4, 315-325