Proceedings of the IEEE
International Conference on Automation and Logistics
Qingdao, China September 2008

The Motion Coordination in the C-Space-Time with a Multi-layered MRS Architecture

Fabio M. Marchese
Dipartimento di Informatica, Sistemistica e Comunicazione
Università degli Studi di Milano-Bicocca
I-20126, Milano, Italy
fabio.marchese@disco.unimib.it

Abstract—**In this paper we present a layered architecture for multirobot motion coordination. The purpose is to control and coordinate autonomous mobile robot with generic shapes and kinematics in a priori known environment. It is a centralized framework, where a leader robot (or a supervisor) plans the motion of all the robots and makes them moving synchronously. The architecture is layered, self-similar and modularized, where each module is realized with concurrent threads. The underlying motion planner is based on an Artificial Potential Fields method applied on a discretized C-Space-Time.**

Index Terms—**Multirobot Systems, Motion Planning, Motion Coordination, Software Architecture**

I. INTRODUCTION

The area of the cooperative/distributed robotics moved its first steps in the 80's, and since then it has received an always 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 with the environment and sharing their resources to achieve a general common task. In the editorial [1] 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 in [3]. 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 in four groups: Communication, Team composition, System architecture, Team size.

II. A Self-similar Layered Architecture for Motion Coordination

In this paper a software architecture for a MRS composed by heterogeneous mobile robots is proposed. The aim of this architecture is the coordination of the motion and the navigation control. In particular, we want that our framework exposes those functionalities that permit the MRS to navigate without any conflict, i.e. without collisions with stationary obstacles and with each other, but also without conflicts at the coordination level. The goal is to make all the robots to move coordinately. In [5] the authors suggest a multirobot three-layered architecture to explicitly coordinate their actions at the different levels. The task allocation is performed using a market-based architecture, in which the robots exchanges the tasks throughout a public auction.

On the base of the taxonomy in [3], we can classify our system as (coordination dimensions): *Cooperation level*: cooperative; *Knowledge level*: unaware; *Coordination level*: strongly coordinated; *Organization level*: strongly centralized. On the standpoint of the system dimensions: *Communication*: direct, throughout a central dispatcher; *Team composition*: heterogeneous; *System architecture*: deliberative; *Team size*: small-medium. 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) that is omniscient with respect to its team mates, and which organizes, synchronizes and controls the other robots (*strongly coordinated*). The other team mates do not have any knowledge about each other (*unaware*). In a strongly centralized architecture, the central unit is responsible to take any decision and the peripheral units operates consequently. This must not to 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 to 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. 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.

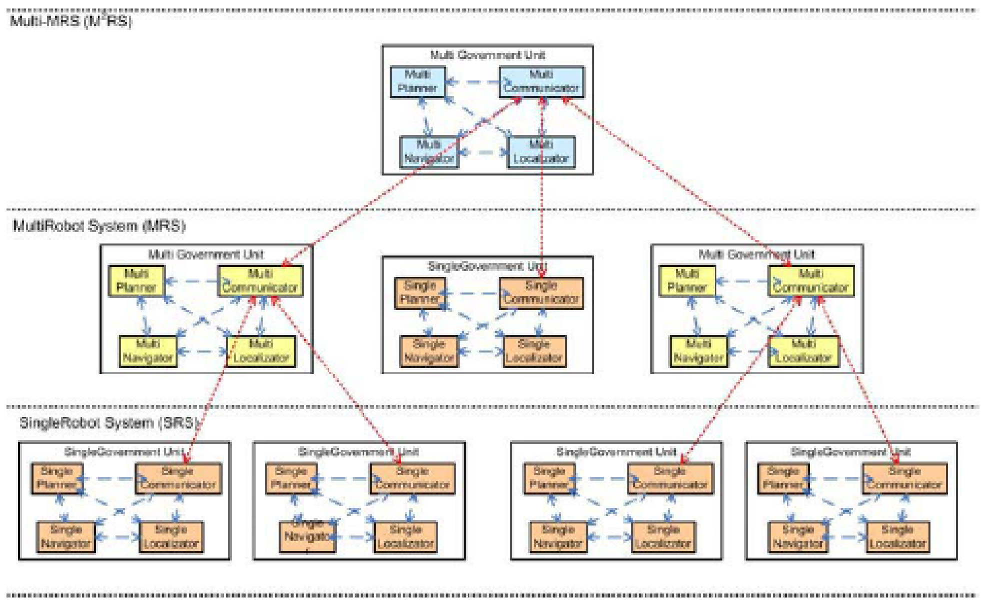

Fig. 1. Two-layers Architecture

A. The MRL Architecture

Our task is to define a framework able to work independently from the single robot hardware and software characteristics. The proposed approach is coherent with a MRS architecture, where the coordination system is just a subpart of it. It is 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^2RS) extending the number of layers, but not changing the significance: each M^nRS 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 $\{M^0RS, MRS, M^2RS, , M^{n-1}RS\}$, where $M^0RS = SRS$ (SingleRobot System). The concept can be abstracted further to higher level where a M^nRS 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

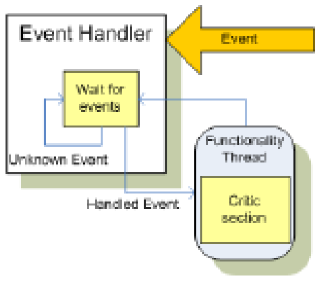

Fig. 2. General module structure

immediate response: an important characteristic for a real-time system that permits to handle fast dynamics of the environment. The delicate aspect of this schema is during the design phase, where it is important to define carefully the sets of events and the corresponding functions (event-handlers) to be activated. Besides architectural issues, which are mainly concerned with the design and then the handling of multi-threads/concurrent systems, the real-time coordination of the motion of n bodies is the major problem. We have identified four basic functionalities sufficient to make a multirobot to navigate in a priori known environment: self-localization, motion planning, navigation and communication.

B. Multi-Localizer functionality

To plan a trajectory or to navigate it is important to know the (multi)robot position and orientation with respect to the environment reference frame. In the case of the multirobot, this function is provided by the Multi-Localizer module (Fig. 3). It collects the poses of every single robots creating a *Multi-Pose*, which is the vector of all the robots poses. This module is only passive, because it is the single robot that sends its pose information whenever it has terminated to update it. To be able to trace the "freshness" of the information, a timestamp is also associated.

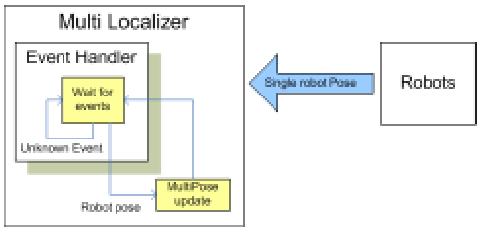

Fig. 3. Multi-Localizer module

C. Multi-Planner functionality

This module is responsible for the planning of the coordinated synchronous motions of all the robots (let's call it *multiplan*). Every module needing a new plan, can make the request signaling the corresponding event, which starts a new thread (Fig. 4). This thread effectively computes the multiplan, first acquiring the most updated multipose from the Multi-Localizer, and the desired goal. If during the computation arrives a new request there are two possible behavior: if the request has been done because there has been a relevant change in the environment (an object is detected in e new position) the planning is aborted and a new computation starts (replanning) with an updated map; in the other cases, the request is ignored leaving the thread to terminate its computation. At the end, the caller is signaled if the new trajectories are ready or if the planning is failed.

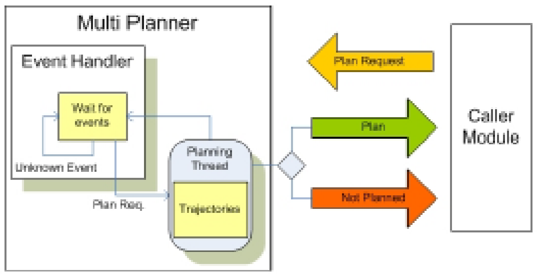

Fig. 4. Multi-Planner module

D. Multi-Navigator functionality

The Multi-Navigator is responsible to drive every robots along the trajectories provided by the Multi-Planner. When a caller decide to start the navigation, it signals the corresponding event (Fig. 5). In response to the event, the Multi-Navigator acquires the multiplan (if available) from the Multi-Planner. The navigation is performed sending a temporized sequence of motion commands to all the robots synchronously. In other words, the navigator sends periodically to all the robots the next pose that each one has to reach from its current pose, realizing a synchronized motion. The current multipose must be received and updated by the Multi-Localizer with a higher frequency (more than double) than the frequency the motion commands are sent and executed.

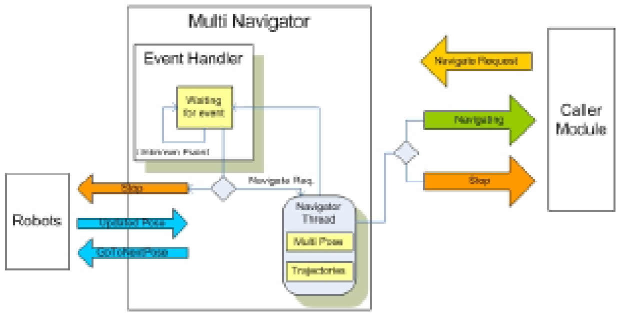

Fig. 5. Multi-Navigator module

E. Multi-Communicator functionality

Without facing problem of the communication protocol definitions, this module tackles the communication at the application level. The Multi-Communicator module is a double event-handler (Fig. 6). There is an "event-handler" that dispatches the messages received from all the robots to the corresponding module, and a second event-handler that, on the base of events signaled by the modules, provides to translate them in messages to be sent to the appropriate robot.

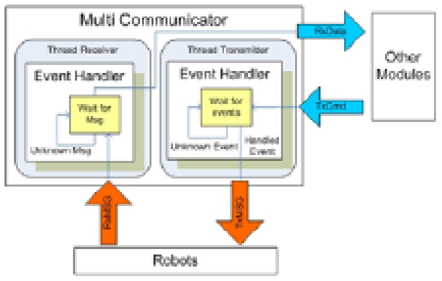

Fig. 6. Multi-Communicator module

F. Multi-Government Unit functionality

The Multi-Government Unit module is the most important functionality, because it schedules all the multirobot activities and coordinates all the robots. First it makes every other module to start and initialize. In Fig. 7 it is shown a simplified activity diagram, in which it is shown the Multi-Government Unit module startup, with a fork starting the threads of the other modules and a corresponding join at the end of the section. A detailed activity diagram has been designed, but it is very complex and so large that it is not possible to include it in the paper. If the task is to make the multirobot to navigate from one multipose to another one, the Multi-Government Unit then activates the Multi-Planner signaling the *plan event*, and, if a plan is available, it signals the Multi-Navigator to start the navigation. Another important feature is the error handling: this module manages all the error conditions occurring in any other module (Fig. 8) in a dedicated thread. It also manages the errors occurring in the robots, errors sent to the attention of the multirobot by means of the communication link.

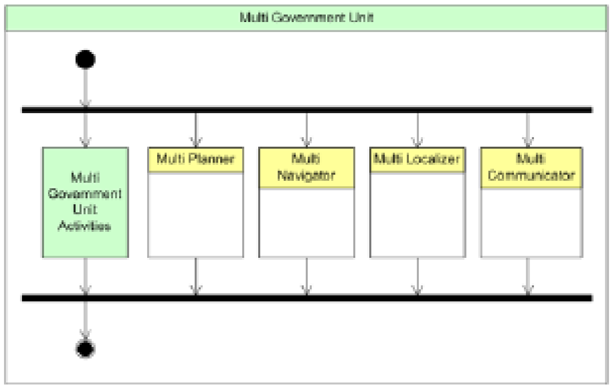

Fig. 7. Multirobot threads startup

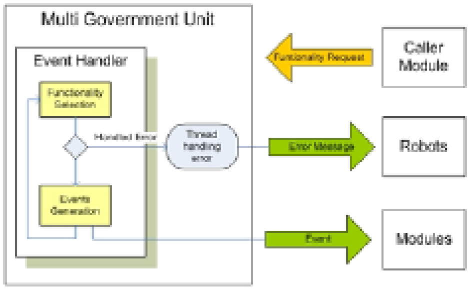

Fig. 8. Multi-Government Unit module

III. The Multirobot Motion Planning algorithm

Many solutions have been proposed during the last twenty-five years for single and multiple robots path or motion planning, based, for example, on a geometrical description of the environment (e.g. [6], [7]). In the eighties, Khatib [8] first proposed this method for the real-time collision avoidance problem of a manipulator in a continuous space. Jahanbin and Fallside introduced a wave propagation algorithm in the Configuration Space (*C-Space*) on discrete maps (*Distance Transform* [9]). In [10], the Numerical Potential Field Technique on the *C-Space* has been introduced. Zelinsky extended the *Distance Transform* to the *Path Transform* [11]. In [12] the author uses a discretized 3D *C-Space-Time* first defined in [13], to coordinate the motion of robots using with the same shape (only square and circle) and a quite simple kinematics (translation only). A solution in the *C-Space-Time* is proposed in [16], 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. We have adopted a decoupled prioritized solution, using Cellular Automata as a formalism on a grid model of the world (Occupancy Grid) with the *C-Space-Time* of multiple robots and Numerical (Artificial) Potential Field Methods. The purpose is to give a simple and fast solution for the motion-planning problem for multiple mobile robots with different kinematics and complex shapes. This method uses a directional (anisotropic) propagation of distance values between adjacent automata to build a potential hypersurface embedded in 5D space. It is possible to find out all the admissible, equivalent and shortest trajectories for each robot following the valleys of the potential surface.

A. Environment and Robots models

The environment is represented using an Occupancy Grid of rectangular cells. Because the obstacle are stationary, it is static (frozen) during the planning phase, but it can changes between two planning sections. A robot is a solid object moving in an environment. To give a description of the robot we need two components: the robot shape and the robot kinematics. In many works the robot has a convex shape and/or it is shrunk to a point of the C-Space. To take care of its real extension, the technique is to enhance the objects with the maximum robot radius. To have a more

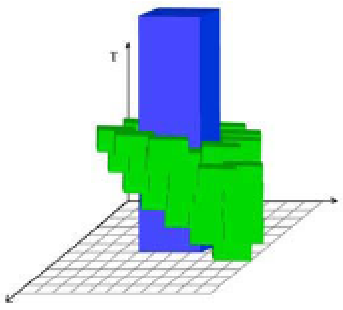

Fig. 9. The potential bowl (green) growing in the C-Space-Time around a stationary obstacle (blue)

precise planning and to face a wider spectrum of problems, we want to represent the complete robot shape, even if it is irregular, with concavities, holes, etc., and with its real extension. This is achieved with a small square occupancy grid, where the cells have the same size of the environment cells, and where the robot kinematic center is placed in the middle. The robot kinematics is defined as a finite set of basic motions (moves), where for each move it is specified the next relative spatiotemporal pose (referred to the current pose at the current time) in term of cells, and its cost. In this way it is possible to define any type of kinematics: car-like, omnidirectional, etc., but it is also possible to specify the motion speed.

B. *The Multirobot Motion Planner module*

The Multirobot Motion Planner module is itself based on a layered architecture. There are two main layers: *Obstacles Layer* and the *Attraction Layer*. Each layer is subdivided in more sub-layers: 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 plus 1 for their orientations (X, Y, Θ) and 1 for the time T). The *Obstacles L.* conceptually depends on the outside environment. Its sub-layers have to react to the "external" changes: the changes of the environment, i.e. the movements of the obstacles in a dynamical world. The *Obstacles L.* is also a *Coordination Space*, in which are progressively mapped the trajectories of all the robots. The *Attractive Layer* 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 moves costs are used to build incrementally a potential bowl starting from the goal cell, placed at a high time slice (the expected time of arrival), and expanding it in all the free space-time. In our case, it is a hypersurface embedded in a 4D space: $\mathbb{R}^2 \times \mathbf{SO(2)} \times \mathbb{R}$, where $\mathbb{R}^2$ is the workspace, enlarged with the robot orientation dimension $(\mathbf{SO}(2))$ and the time T. 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 C-Space-Time cells, which compute the potential values depending on the potential values of the surrounding space-time cells. Iterating this process for all the free cells, the potential hypersurface is built, leaving the goal cell at the minimum potential. The potential value has a geometrical mean: it is the total (minimal) cost needed to reach the goal from the current cell. The entire trajectory is computed just following the direction of the negated gradient of the surface from the starting point. Every robot has different goals and kinematics, hence a potential bowl for each one has to be generated in separate *Attractive Layers*. The potential bowl is built only in the free $\mathcal{C}$*-space*, surrounding the $\mathcal{C}$*-Obstacles* regions (Fig. 9). Therefore, the robot *Attractive Layer* has to depend 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. Entailing the layer structure previously introduced, it is now possible to describe the algorithm that computes the multirobot trajectories. The first phase is to establish the robots priorities (user defined). The

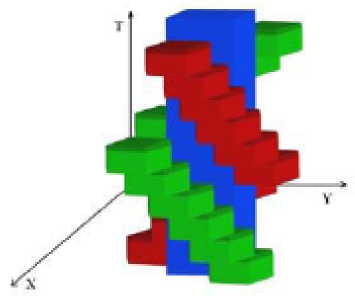

Fig. 10. The sequences of C-Space-Time cells generated for two robots (green and red) around a stationary obstacle (blue)

following step is to initialize the *Obstacles Layer* with the obstacles distribution known from the Occupancy Map of the workspace. Each temporal sublayer is set to the Occupancy Map (*full* if the cell has an obstacle inside, *empty* otherwise). Then, the algorithm computes the potential bowl in the *Attractive Layer* of the robot with the highest priority level based on the *Obstacles Layer*. When the potential surface is determined, it extracts the shortest path, following the negated gradient from the starting cell. For each passing point, then it adds the robot silhouette, in each temporal sublayer of the *Obstacles Layer*. In this way, the first robot becomes a space-time "obstacle" for the successive robots, with lower priorities. The *Obstacles Layer* has a central role in this phase, because it ensures that the robots avoid each other and the trajectories do not intersect each other, even taking into account of their real extensions. For this reason, in this context it is also a *Coordination Space*. The procedure is repeated for the successive robot with a lower priority level, starting to build its *Attractive Layer* on the base of the previously modified *Obstacles Layer*. The algorithm terminates when it has computed all the robots trajectories. In then following, few examples of simulation of motion coordination are shown. The first example regards two robots, the second one with a special disjoined shape (Fig. 11). Their silhouettes are the projections on the plane of objects like a "Truck" and a "Portainer crane" moving on wheels. The triangles represent the kinematic center and the robot orientation. The task is to make passing the truck under (between the uprights) the crane. This is an example of a vehicle with a silhouette projected on the plane which is composed by two disjoined parts. In the second simulation,

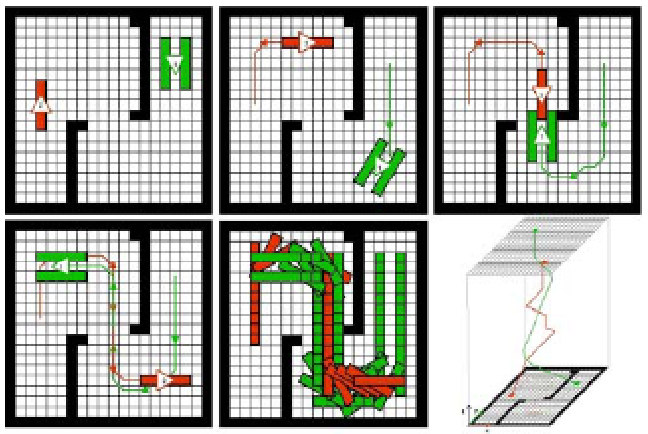

Fig. 11. Truck and Portainer crane example

we have a robot the have to pass from one room to another one, but the passage is closed by a second O-ring-shaped robot (Fig. 12). It is noteworthy that the kinematic center of the latter is on a stationary obstacle inside the ring. The only solution that allows the first robot to transit is that the o-ring robot rotates synchronously with the first one. The planning times on a Intel PIV 2.26 GHz are respectively: 185 ms (overall number of cells to be updated: 416Kcells) and 230 ms (1.55M cells).

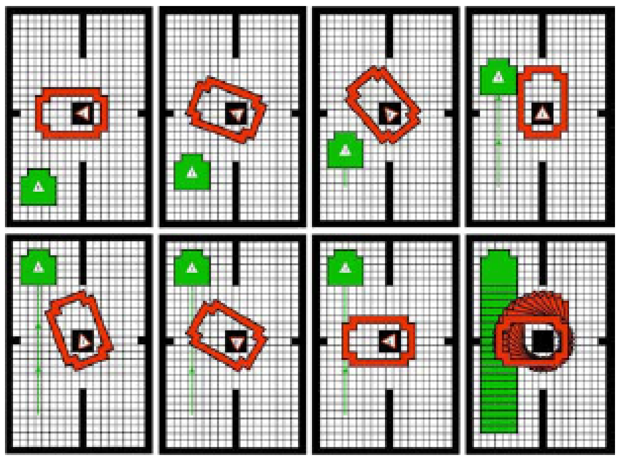

Fig. 12. O-Ring robot closing a passage

IV. Conclusions

In this paper we have described a software architecture for multirobot system motion coordination. Our approach is centralized: there is a robot leader which plans the motion in a known environment for all the robots and then it coordinates synchronously the overall motion. The architecture is layered at different level of resolution (there are layers with sublayers inside them) and self-similar. It allows to coordinate robots with different software and hardware characteristics, modelling them with the same structures that expose the same functionalities, thus reducing the design and implementation phase at the application level. The multirobot framework is event driven and realized with concurrent threads, permitting to tackle also asynchronous situations. Its behavior and properties are still under evaluation from the theoretical point of view. The algorithm for the motion planner module is decoupled and prioritized and it is based on an Artificial Potential Fields method applied on a discretized $\mathcal{C}$-*Space-Time*. It is able to plan in a fast and simple way the movements of heterogeneous robots, with complex shapes and different kinematics.

References

[1] T. Arai, E. Pagello, and L. E. Parker, "Editorial: Advances in multi-robot systems," *IEEE Trans. on Robotics and Automation*, vol. 18, no. 5, pp. 655–661, 2002.

[2] G. Dudek, M. Jenkin, E. Milios, and D. Wilkes, "A taxonomy for multiagent robotics," *Autonomous Robots*, no. 3, pp. 375–397, 1996.

[3] A. Farinelli, L. Iocchi, and D. Nardi, "Multirobot systems: A classification focused on coordination," *IEEE Trans. on System, Man, and Cybernetics-part B: Cybernetics*, vol. 34, no. 5, pp. 2015–2028, 2004.

[4] L. E. Parker, "Cooperative robotics for multi-target observation," *Intelligent Automation and Soft Computing*, vol. 5, no. 19, pp. 1–17, 1999.

[5] R. Simmons, T. Smith, and M. B. Dias, "A layered architecture for coordination of mobile robots," in *Multi-Robot Systems: From Swarms to Intelligent Automata (A. Schultz and L. Parker eds.)*. Kluwer, 2002.

[6] T. Lozano-Pérez and M. A. Wesley, "An algorithm for planning collision-free paths among polyhedral obstacles," *Comm. of the ACM*, vol. 22, no. 10, pp. 560–570, October 1979.

[7] T. Lozano-Pérez, "Spatial planning: A configuration space approach," *IEEE Trans. on Computers*, vol. C-32, no. 2, pp. 108–120, February 1983.

[8] O. Kathib, "Real-time obstacle avoidance for manipulator and mobile robots," in *Int. Conf. on Robotics and Automation*, 1985.

[9] M. R. Jahanbin and F. Fallside, "Path planning using a wave simulation technique in the configuration space," in *Artificial Intelligence in Engineering: Robotics and Processes (J. S. Gero ed.)*. Southampton: Computational Mechanics Publications, 1988.

[10] J. Barraquand, B. Langlois, and J. C. Latombe, "Numerical potential field techniques for robot path planning," *IEEE Trans. on Systems, Man and Cybernetics*, vol. 22, no. 2, pp. 224–241, 1992.

[11] A. Zelinsky, "Using path transforms to guide the search for findpath in 2d," *Int. J. of Robotics Research*, vol. 13, no. 4, pp. 315–325, August 1994.

[12] C. Warren, "Multiple robot path coordination using artificial potential fields," in *Proc. of the IEEE Int. Conf. on Robotics and Automation (ICRA)*, 1990, pp. 500–505.

[13] M. Erdmann and T. Lozano-Pérez, "On multiple moving obstacles," in *Proc. IEEE Int. Conf. on Robotics and Automation*, San Francisco (USA), 1986, pp. 1419–1424.

[14] T. Simeon, S. Leroy, and J. P. Laumond, "Path coordination for multiple mobile robots: a resolution-complete algorithm," *IEEE Trans. on Robotics and Automation*, vol. 18, no. 1, pp. 42–49, February 2002.

[15] S. M. LaValle and S. A. Hutchinson, "Optimal motion planning for multiple robots having independent goals," *IEEE Trans. on Robotics and Automation*, vol. 14, no. 6, pp. 912–925, December 1998.

[16] M. Bennewitz, W. Burgard, and S. Thrun, "Optimizing schedules for prioritized path planning of multi-robot systems," in *IEEE Int. Conf. on Robotics and Automation (ICRA)*, Seoul (Korea), 2001.

[17] J. E. Hopcroft, J. T. Schwartz, and M. Sharir, "On the complexity of motion planning for multiple independent objects: Pspace-hardness of the warehouseman's problem," *Int. J. of Robotics Research*, vol. 3, no. 4, pp. 76–88, 1984.