J Intell Robot Syst
DOI 10.1007/s10846-012-9783-5

# A Decentralized Architecture for Multi-Robot Systems Based on the Null-Space-Behavioral Control with Application to Multi-Robot Border Patrolling

**Alessandro Marino · Lynne E. Parker · Gianluca Antonelli · Fabrizio Caccavale**

Received: 20 April 2012 / Accepted: 11 September 2012


**Abstract** This paper presents a control architecture for multi-robot systems. The proposed architecture has been developed in the framework of the Null-Space-based-Behavioral (NSB) control, a competitive-collaborative behavior-based control approach. The standard NSB statically determines a set of suitably defined elementary tasks (behaviors) and their priorities, i.e., they cannot be dynamically changed according to mission requirements and environmental constraints. In this paper, a three layer architecture has been designed in order to avoid such a drawback. The single robotic unit (agent) performing the mission is placed on the lower layer. In the middle layer, suitably defined elementary *behaviors* are defined; these elementary *behaviors* are then combined, via the NSB approach, in more complex *actions*. The upper layer is a Supervisor in charge of dynamically selecting the proper action to be executed. As further contribution, the architecture has been applied to the multi-robot border patrolling mission to generate a decentralized, deterministic and non-communicative solution that is robust to faults, and prevents collisions, even in the case of high robot density. Finally, the simulations on a team composed by a large number of robots, and experiments on a real setup, composed by three Pioneer-3DX robots, are provided.

**Keywords** Behavioral control · Null-space based behavioral approach · Multi-robot systems · Swarm robotics · Border patrol

The manuscript is based on three conference papers of the same authors, namely, Marino et al. [27–29].

A. Marino (✉)
Dipartimento di Ingegneria Elettronica e Ingegneria Informatica, Università degli Studi di Salerno,
Via Ponte Don Melillo, 84084, Fisciano, SA, Italy
e-mail: almarino@unisa.it

L. E. Parker
Department of Electrical Engineering and Computer Science, The University of Tennessee,
1122 Volunteer Blvd, Knoxville, TN 37996-3450, USA
e-mail: leparker@utk.edu

G. Antonelli
DIEI - Dipartimento di Ingegneria Elettrica e dell'Informazione, Università di Cassino e del Lazio Meridionale, Via G. Di Biasio 43,
03043, Cassino, FR, Italy
e-mail: antonelli@unicas.it

F. Caccavale
Scuola di Ingegneria, Università degli Studi della Basilicata, Viale dell'Ateneo Lucano 10,
85100, Potenza, Italy
e-mail: fabrizio.caccavale@unibas.it

## 1 Introduction

Collaborative multi-robot teams have great potential to add capabilities and minimize risks,

Published online: 03 October 2012

especially in the security domain. In fact, robots are increasingly used in performing patrolling, surveillance and military tasks. Today, such robots are not able to carry out a mission on their own, due to limited autonomy; hence, most of them are remotely controlled by human operators. This is due also to the risks connected to robots' failures in highly dynamic scenarios.

Of course, multi-robot systems require a coordination mechanism to accomplish the mission. Although several paradigms have been developed in the last decades, behavior-based robotics has been proven to be a successful technique [9].

Arbitration mechanisms are usually adopted for selection of active behaviors on the basis of mission and system requirements. An example is the subsumption architecture, in which a priority is assigned to each behavior, and behaviors with higher priorities are allowed to override the output of behaviors with lower priority [9]. However, multiple non-conflicting behaviors cannot be activated simultaneously.

Command fusion mechanisms try to establish a sort of cooperation between behaviors. For example, the motor schema allows the output of multiple behaviors to be combined [7]. However, it suffers from the problem of local minima and command averaging without explicit handling of conflicting behaviors.

The traditional NSB approach requires the definition of a set of elementary behaviors [4]. The mission achievement is obtained by properly combining the elementary behaviors via the null-space projection mechanism.

Thus, the NSB can be defined as a competitive-cooperative approach, that overcomes the disadvantages of the above approaches, allows the activation of several behaviors at once and properly handles the conflicts among them. A comparison of the NSB approach with the other paradigms cited above can be found in Antonelli et al. [5]. However, the behaviors and their priorities are statically determined, i.e., they cannot be changed dynamically according to mission requirements and environment constraints. Hence, the standard NSB approach can be effectively applied only to relatively simple missions. Thus, it is worth further extending this mechanism to handle dynamic changes of behaviors and their priorities.

In this paper, the NSB approach for multi-robot systems has been extended with respect to previous published papers. Namely, a mission whose complexity cannot be effectively handled by the standard NSB approach is first decomposed into several sub-missions (or sub-goals). Each sub-mission requires the activation of multiple behaviors, each characterized by its priority, and their combination according to the Null-Space-Behavioral approach. Such a combination of behaviors is called *action*. Finally, a higher level (i.e., the Supervisor) is needed to dynamically select the action to be activated. In addition, the classic NSB has a centralized structure, i.e., a central computing unit or a leader agent is in charge of collecting the system's state and calculating the agents' motion commands. Here, a different perspective is used. Namely, a completely decentralized architecture is proposed to overcome this limitation.

The proposed architecture has been applied to the multi-robot patrolling mission. The overall patrolling architecture in the NSB framework is able to perform the mission in very critical conditions and under strong requirements (see Section 3.1). In detail, a suitable set of elementary behaviors is defined, on the basis of typical patrolling mission requirements. Then, these behaviors are composed, via NSB, into more meaningful and complex actions, which represent given sub-goals. Once the set of actions is defined, the Supervisor selects the action to be performed by each robot in order to achieve mission goals.

The proposed control architecture can be adopted by multi-robot systems to perform different classes of missions, by properly formulating the set of behaviors and actions. For the sake of clarity, in this paper the designed procedure has been presented by direct application to the patrolling mission that is a challenging scenario that helps to understand the main features of the approach.

The paper has been organized as follows. In Section 2, the contributions of the paper are strengthened and summarized, while in Section 3,

an overview of the patrolling mission is carried out and the main assumptions and requirements are determined. A description of the designed architecture is given in Section 4. A short review of the NSB approach together with the definition of the elementary behaviors and actions is given in Section 5. The supervisor in charge of selecting the proper action to execute is described in Section 6. Simulations and experimental results are presented in Sections 8 and 9. Finally, conclusions are drawn in Section 10.

## 2 Main Contributions and Novelty

The contribution of the paper aims to be twofold. First, a framework has been designed that allows handling complex missions by means of a three-layered architecture. This objective is achieved thanks to the use of the NSB in the middle layer, that allows building a set of well-structured behaviors and actions, whose output is fully predictable. Moreover, the NSB framework allows the designer to focus on the design of the top level rather than on the low level effects. Second, the feasibility of the approach has been demonstrated by showing a practical example of how the proposed architecture can be used to accomplish the patrolling task. In addition, experiments have been run on a real setup composed by three fully autonomous vehicles.

## 3 The Patrolling Mission: Main Requirements and Assumptions

As stated above, the mission to accomplish is the surveillance of a given area, i.e., a border or any military and civil facility. An early example of a robotic surveillance system is the Mobile Detection Assessment and Response System-Exterior (MDARS-E) [19] whose goal is to provide multiple mobile platforms that perform random patrols within assigned areas of warehouses. Sandia National Laboratories has developed the Surveillance And Reconnaissance Ground Equipment (SARGE) robotic vehicle for the US Department of Defense [35]. SARGE is a teleoperated robot for battlefield surveillance applications without computing power, aimed at supporting autonomous navigation or vision processing.

Recently, commercial border patrol applications have been proposed as well, such as Guardium [21] and the unmanned autonomous speed boat Protector manufactured by Ltd [36].

Perimeter surveillance algorithms form the basis for effective execution of monitoring tasks in a number of applications fields, e.g., monitoring of oil spills [15], contaminant clouds, algae bloom [8], forest fires [12] and border security [23].

In Machado et al. [26], an analysis of the main patrolling task issues and some multi-agent-based solutions are presented. Several features (agent type, agent communication, coordination scheme, agent perception and decision-making) are evaluated by using different criteria.

In Kingston et al. [25], a decentralized solution to the multi-robot perimeter surveillance is presented, where changes of both the shape of the perimeter and the number of robots are taken into account; moreover, critical coordination information is exchanged to optimize some performance index. In Agmon et al. [1], the authors analyze non-deterministic paths for a group of homogeneous mobile robots patrolling a frontier, under the assumption of a hostile agent trying to enter the area, where the latter has full knowledge of the algorithm.

In this paper, a linear border to be patrolled was considered. The border may be any imaginary (like in the case of aerial vehicles as in [25]) or real line surrounding strategical facilities or even countries' borders. In a real mission scenario, the first task the vehicles have to be able to perform is to get to the border itself from, for example, a base station. After the border has been reached, the vehicles have to move along the border while keeping staying on the border. In addition, static or dynamic obstacles need to be considered. Obstacles can be represented by other teammates or natural obstacles close to the border. Finally, the case of a friend vehicles that try to enter the border can be of interest. A friend vehicle

is any agent that is allowed entering the area to be patrolled and that has not to be stopped by patrolling vehicles.

In the following, the main requirements and assumptions related to the border patrolling mission are listed.

### 3.1 Requirements

A patrolling mission requires a high degree of autonomy and robustness. Several aspects and constraints might affect the mission achievements, e.g., occurrence of robot faults, large number of patrolling robots, presence of friends or intruders interfering with patrolling robots, limited communication range and computational capabilities. For these reasons, the following requirements need to be met.

*Requirement 1: Decentralization* The control architecture needs to be fully decentralized, i.e., there is not a central unit in charge of computing the patrolling robots' motion commands, by gathering information from them.

*Requirement 2: Communication* Explicit communication between patrolling robots is forbidden. This requirement is devoted to supply more robustness to the control architecture as well as to enhance security, since, in certain applications, encryption of communications would be needed, at the expense of an increased software complexity. Moreover, in some scenarios (e.g., underwater environments and/or wide areas to be patrolled) reliable communications could be not available. Moreover, even in the case that explicit communications can be assumed, the gathered data may require significant time to transmit (e.g., complete video footage). Of course, this requirement is not independent from the previous one, since, without any form of communication, a centralized architecture cannot be implemented.

*Requirement 3: Collision avoidance* It is obvious to require that each patrolling robot must avoid collisions with other teammates, friends or intruders, even in the case of a large number of robots. In addition, in the case of a large number of robots, interference phenomena must be avoided, i.e., the adoption of a large number of patrolling robots must not lead to mission failure or other unexpected and undesired events.

*Requirement 4: Robustness* Patrolling missions are usually performed in challenging environments. Hence, it is mandatory to deal with robot faults, by providing a suitable degree of tolerance to faults of the single agents in the patrolling team. More generally, sudden changes in the number of patrolling robots (due, e.g., to an increase of the available robots or failures of single robots) must be taken into account.

*Requirement 5: Computational burden* Usually, in multi-robot systems, relatively simple robots are employed to achieve a given mission. Thus, the computational burden of the control algorithms versus the available computational power of the single robotic units must be taken into account.

### 3.2 Assumptions

The following assumptions are adopted to develop the proposed solution to the multi-robot border patrolling problem, according to the above defined requirements.

*Assumption 1: Border geometry* The general perimeter surveillance problem is reduced to the linear surveillance problem, by assuming that the perimeter to be monitored is homeomorphic to a line, and thus it can be represented as a single path between two points. In addition, closed perimeters are allowed.

*Assumption 2: Visibility range* It is assumed that each robot has a limited visibility range, where it recognizes the presence of other agents and detects the border. For the sake of simplicity, the visibility area is modeled as a circle around the robot. This is not an unrealistic assumption, since it well approximates the case of pan-tilt-zoom camera sensors, laser range finders or omni-directional cameras.

*Assumption 3: Localization capabilities* It is assumed that each robot is able to localize itself in the environment and knows a local (i.e.,

limited to its visibility range) geometric description of the border (or is able to locally estimate the geometry of the border). Localization and, in general, Simultaneous Localization and Mapping (SLAM) [38] are well-established problems in the literature and will be not addressed in this paper. Indeed, each robot needs to estimate only its position with respect to the border and other agents that are in its visibility range. How this goal is achieved highly depends on the available sensors and the practical goals to be faced. A notable example is given in Bruemmer et al. [10], where a swarm is able to locate and surround a water spill using social potential fields implemented via IR, chirps and light sensing; moreover, in Clarka and Fierro [15], cameras are used to locate the perimeter of the area to patrol. Finally, it assumed that each robot is able to follow the border [40].

*Assumption 4: Safety area* Each robot must be characterized by its own safety area, in which other agents are not allowed to enter. The safety area is a circle around the robot (smaller than the visibility area). Such an assumption is necessary to meet the collision avoidance requirement, particularly in the case of teams composed by a large number of robots, where conflicts and interferences may arise.

*Assumptions 5: Awareness* Each robot does not know the exact number of patrolling robots. However, it is aware of the existence of other agents (other patrolling robots, friends or intruders). As described in the following, the awareness assumption will be implicitly taken into account by properly defining the elementary behaviors.

## 4 Control Architecture

### 4.1 Paradigm

By taking into account the requirements defined in Section 3.1, an approach belonging to the swarm robotics class has been adopted. In swarm robotics (see Dorigo and Sahin [16] for a survey and Dudek et al. [17] for a taxonomy), the achievement of the mission is the result of the cooperation of independent agents forming the swarm. The swarm intelligence aims at developing robust task solving by minimizing the complexity of the individual units; its central idea is to distribute the control over a group of numerous minimalist robots rather than gathering and redistributing information with the help of a central unit. There are two main advantages of this approach: first, scalability from a few to thousands of units, second, increased system robustness, not only through unit redundancy but also through the unit simple design. In this approach, the overall behavior is generally emergent, i.e., each unit does not have a global cognition of the assigned mission and reacts to external stimuli via simple behaviors, where, in general, the stimuli-behaviors couples are organized in such a way that the cooperation implicitly arises from the interaction among robots and with the environment.

Moreover, this approach is clearly modular, since new emergent behaviors at the swarm level can be obtained and, in general, more complex missions could be accomplished by properly defining new behaviors at the level of the single unit. There are, of course, some drawbacks connected to this approach. Namely, formal methods are still not available that can generate (starting from a suitable description of the overall mission) a set of stimuli-behaviors couples, whose interaction leads to task accomplishment. This is true especially in the presence of dynamically changing environments.

### 4.2 Outline of the Control Architecture

Since the approach is completely decentralized and disallows any form of communication, the control architecture described in the following must refer to the single robot in the patrolling team.

As it can be seen in Fig. 1, the control architecture is composed by three layers. Starting from the top layer, they are the *Supervisor Level*, the *Action Level* and the *Robot Level*. The first two levels are abstract levels. In particular, the Action Level, described in detail in Section 5, defines the set of actions the robot can undertake. Clearly, the set of actions depends on the mission to be achieved, involving different skills. As already stated, this means that the architecture does

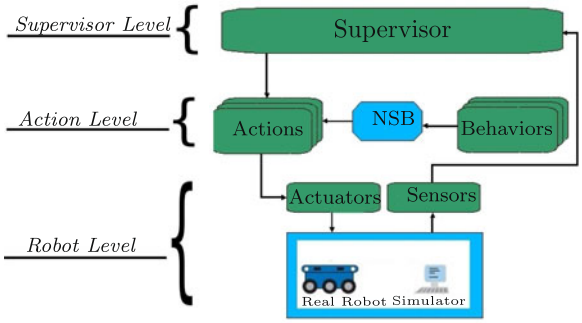


**Fig. 1** Sketch of the control architecture

not depend on the particular mission, but can be applied to different missions by properly defining the action set. Once the actions have been defined, each robot chooses the proper action to perform, according to its internal state and environmental information. To this aim, a Supervisor needs to be designed. Finally, the Robot Level comprises the computing hardware/software and the mechanical features of the single robot; the structure of this layer depends on the adopted robotic platform and is primarily concerned with the available sensors and actuators.

A description of the platform used to test the proposed approach is given in Section 9. In the following, the architecture will be detailed with reference to the patrolling mission

## 5 Behaviors and Actions

In this section, the set of behaviors required by the patrolling mission is defined. Then, these behaviors are combined into more complex and meaningful actions by means of the Null-Space-based-Behavioral (NSB) approach.

### 5.1 Review of the NSB Approach

A complex mission for a robotic system can be encoded via different basic tasks (behaviors) to be properly combined to achieve mission goals. As already stated, the NSB approach can be defined as a competitive-cooperative approach, trying to overcome the disadvantages of arbitration and command fusion approaches; it is based on the task-priority control approach for robotic manipulators [32]; it has been experimentally compared to the layered-control system and motor-schema control in the case of a wheeled mobile robot [5], and successfully applied to control of a platoon of ground vehicles [6].

Let us consider the task function $\boldsymbol{\sigma} \in \Re^m$ which is the mathematical representation of a generic task to be achieved by a robotic system. It is assumed that

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{p}) \in \Re^m, \tag{1}$$

where $\boldsymbol{p} \in \Re^n$ is a vector describing the system configuration (e.g., in case of a team of robots, $\boldsymbol{p}$ is the vector containing the positions of all the vehicles) and $m$ is the dimension of the task space. Assuming that $\boldsymbol{\sigma}$ is a differentiable function, it is useful to consider the relationship between the first derivatives of $\boldsymbol{\sigma}$ and $\boldsymbol{p}$

$$\dot{\boldsymbol{\sigma}} = \frac{\partial \boldsymbol{\sigma}(\boldsymbol{p})}{\partial \boldsymbol{p}} \dot{\boldsymbol{p}} = \boldsymbol{J}(\boldsymbol{p})\boldsymbol{v}, \tag{2}$$

where $\boldsymbol{J} \in \Re^{m \times n}$ is the task Jacobian matrix. Equations 1 and 2 represent the so-called direct task model of the system, i.e., the equations needed to compute the value of the task function, $\boldsymbol{\sigma}$, given the system configuration, $\boldsymbol{p}$. The inverse task model problem is defined as the determination of the desired system's configuration, $\boldsymbol{p}_d$, corresponding to a desired task function, $\boldsymbol{\sigma}_d$. It can be easily recognized that the inverse task model problem is of the utmost importance, since, by means of the desired task function, the desired high level aggregate behavior of the system is specified, while the corresponding desired configuration represents the set of commands to be delivered to the actuators. A possible solution to the inverse task model problem can be sought at the differential level, by inverting the (locally linear) mapping (Eq. 2); this solution has been widely studied in robotics (see, e.g., [37] for a tutorial). A typical requirement is to pursue minimum-norm velocity in a closed loop version, leading to

$$\boldsymbol{v}_d = \boldsymbol{J}^{\dagger}\left(\dot{\boldsymbol{\sigma}}_d + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}\right), \tag{3}$$

where $\boldsymbol{J}^{\dagger} = \boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{J}^T\right)^{-1}$ is the pseudo-inverse of the Jacobian matrix, $\boldsymbol{\Lambda}$ is a suitable positive-

definite matrix gain and $\widetilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_d - \boldsymbol{\sigma}$ is the task error.

It can be easily shown that Eq. 3 leads to the following asymptotically stable error system

$$\dot{\widetilde{\boldsymbol{\sigma}}} + \boldsymbol{\Lambda}\widetilde{\boldsymbol{\sigma}} = \mathbf{0}. \tag{4}$$

When dealing with complex missions, several tasks need to be handled. If $N$ tasks (encoded by functions $\boldsymbol{\sigma}_{d,1}, \ldots, \boldsymbol{\sigma}_{d,N}$) are considered, the NSB solution to the task combination can be formulated in a recursive way, by computing the series of velocity vectors as follows

$$\begin{cases} \boldsymbol{v}_{d,i} = \boldsymbol{J}_i^{\dagger}\left(\dot{\boldsymbol{\sigma}}_{d,i} + \boldsymbol{\Lambda}\widetilde{\boldsymbol{\sigma}}_i\right) \\ \boldsymbol{v}^{(i)} = \boldsymbol{v}_{d,i} + \boldsymbol{N}_{s,i}\boldsymbol{v}^{(i+1)}, \end{cases} \quad i = 1, 2, \cdots N, \tag{5}$$

where $\boldsymbol{J}_i$ is the $i$-th task Jacobian, $\boldsymbol{v}^{(N+1)} = \mathbf{0}$, $\boldsymbol{v}_d = \boldsymbol{v}^{(1)}$ and the matrix

$$\boldsymbol{N}_{s,i} = \boldsymbol{I} - \boldsymbol{J}_{s,i}^{\dagger}\boldsymbol{J}_{s,i}^{T}$$

is a projector onto the null-space of $\boldsymbol{J}_{s,i}$ (i.e., its columns form a basis of the null space of $\boldsymbol{J}_{s,i}$). The matrix $\boldsymbol{J}_{s,i}$ is obtained by stacking the matrices $\boldsymbol{J}_k$ with $k = 1, 2, \cdots, i$. In the case of two tasks $\boldsymbol{\sigma}_1 \in \Re^{m_1}$ and $\boldsymbol{\sigma}_2 \in \Re^{m_2}$, as will be the case in this paper, Eq. 5 becomes

$$\begin{cases} \boldsymbol{v}_{d,1} = \boldsymbol{J}_1^{\dagger}\left(\dot{\boldsymbol{\sigma}}_{d,1} + \boldsymbol{\Lambda}\widetilde{\boldsymbol{\sigma}}_1\right) \\ \boldsymbol{v}_{d,2} = \boldsymbol{J}_2^{\dagger}\left(\dot{\boldsymbol{\sigma}}_{d,2} + \boldsymbol{\Lambda}\widetilde{\boldsymbol{\sigma}}_2\right) \\ \boldsymbol{v}_d = \boldsymbol{v}_{d,1} + \boldsymbol{N}_1\boldsymbol{v}_{d,2}. \end{cases} \tag{6}$$

In sum, the commanded velocities corresponding to a lower priority task are projected onto the null-space of the immediately higher priority task; then, eventually conflicting velocity components are cut off before being added to the higher priority task velocity components. In this way, lower priority behaviors are executed only in their components not affecting higher priority behaviors. Convergence to zero of task errors can be guaranteed for properly defined tasks [3]. On the other hand, a differentiable analytic expression of the defined behaviors is required, so as to compute the required Jacobian matrices.

### 5.2 Elementary Behaviors and Actions

In the case of the border patrolling of a linear border, a set of elementary behaviors is defined:

- `Reach Frontier`
- `Patrol Frontier Clockwise`
- `Patrol Frontier Counter-Clockwise`
- `Teammate Avoidance`
- `Friend Avoidance`

whose semantics and analytical expressions are given in Appendix A.1.

As motivated above, it is appropriate to compose the elementary behaviors into more complex behaviors; the latter are sometimes defined as *behavior sets* in the literature. Similar to the concept of behavior set, here a higher abstraction layer is introduced: the *action*. An action is given by the proper composition, achieved via NSB, of several elementary behaviors and represents a macroscopic attitude of the robotic system. Only one single action can be active at once.

For the specific case of the border patrolling task, the following set of actions is obtained by combining the elementary behaviors defined above:

- `Action Reach Frontier`
- `Action Keep Going`
- `Action Patrol Clockwise`
- `Action Patrol Counter-Clockwise`
- `Action Teammate Avoidance`
- `Action Friend Avoidance`

According to the definitions in Appendix A.2, each action is given by elementary behaviors arranged in priority; e.g., the `Reach Frontier` action properly combines the elementary behaviors `Reach Frontier` and `Teammate Avoidance`, depending on the sensed presence of other patrolling robots in the visibility range and the distance from the border. It is worth noticing that such actions require that each robot is able to recognize other agents and their nature (friends or teammates).

### 5.3 Behaviors Compatibility

In the NSB framework it is possible to define conditions for behavior compatibility [3], allowing

the robot to properly handle conflicts among different behaviors (algorithmic singularities). Issues related to singularities are deeply discussed in Chiaverini [14], where a singularity-robust task-priority strategy, in the case of two tasks, is derived. According to this technique, a secondary task is fulfilled if it does not conflict with a higher level task, while it is released in the case it is conflicting. The more general case of compatibility of three or more tasks handled in the NSB framework has been discussed in Antonelli [3], by resorting to a Lyapunov-based analysis. A straightforward application of the results of the above mentioned papers guarantees proper definition of all the *actions* used in this paper; in particular, collisions between robots and between robots and obstacles are avoided. Compatibility of the behaviors combined in actions is discussed in the Appendix.

## 6 The Supervisor

The set of actions defines the robot's skills needed to react to situations encountered in the environment. Hence, each robot has to be equipped with a Supervisor that is in charge of selecting the action to be executed, according to the robot's internal state and/or external stimuli. According to the assumptions and requirements defined in Section 3, the Supervisor of each robot decides the next action to be performed, based only on locally available information. Although several paradigms might be used, two appealing tools for the design of the Supervisor are represented by Finite State Automata and Fuzzy Logic Engines. The Finite State Automata Supervisor [27] will be described in the following, while details on the Fuzzy Logic Supervisor can be found in Marino et al. [29].

A comprehensive description of hierarchical state machines and of their properties is carried out in Alur and Yannakakis [2]. Finite state machine action selection mechanisms assume that [11]:

- there are only a limited number of salient situations the agent can find itself in,
- these situations are mutually exclusive,
- actions to be performed can be easily mapped to situations.

To build a Finite State Automata, two sets must be determined:

- the sets of the states the agent can be in,
- the sets of causes forcing the agent to change its state.

The structure of the supervisor that selects the proper action is shown in Fig. 2.

As it can be seen, the supervisory layer is arranged in a hierarchical way, by defining three levels. At the top level two scenarios have been identified. The first one corresponds to the situation where a friend agent is in the safety area

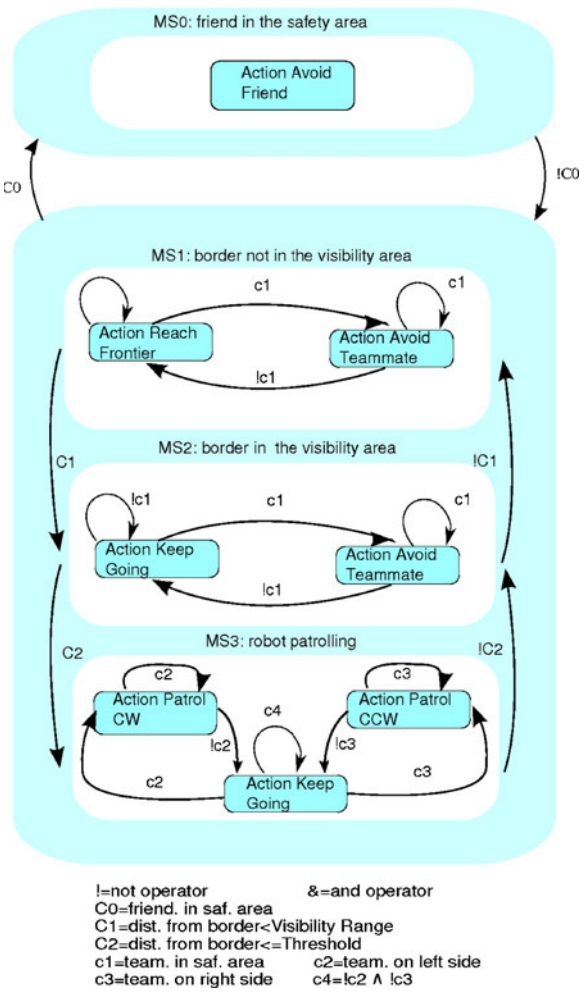


**Fig. 2** Sketch of the supervisor

of the patrolling robot. In the second scenario no friend agents are in the visibility area of patrolling agents. The second level defines four macro-states: `MS0`, for the first scenario, `MS1`, `MS2` and `MS3` for the second scenario. At each time instant the robot can be in one of the macro-states. In macro-state `MS0`, `Action Avoid Friend` is active. On the other hand, if no friend agents are present (second scenario), one of the states `MS1`, `MS2` or `MS3` is active. In the macro-state `MS1`, the robot tries to reach the border (and, at the same time, avoid the teammates). In the macro-state `MS2` the robot behaves as if it is patrolling the border (while avoiding teammates); however, in this state, the robot still cannot be considered as performing the patrolling mission. In the macro-state `MS3`, the robot performs the patrolling mission and does not allow robots approaching the border to influence its motion.

The reason for the distinction between macro-states `MS2` and `MS3` is that, in the case of a large number of patrolling robots, interference between robots can be effectively counteracted. In fact, when a robot is in the macro-state `MS3`, it cannot be influenced by other teammates that are not in the same macro-state; the avoidance of other teammates that are in the macro-state `MS3` is achieved by activating `Action Patrol Clockwise` and/or `Action Patrol Counter-Clockwise`.

In detail, the supervisor acts as described in the following.

*Scenario 1* One or more friend agents are in the safety area of the robot

- *Macro-State MS0*:
  - `Action Avoid Friend` is active

*Scenario 2* No friend agents are in the safety area of the robot

- *Macro-State MS1*: The distance between the robot and the border is larger than the Visibility Range
  - **if** no teammate is in the safety area **then** `Action Reach Frontier` is active,
  - **if** one or more teammates are in the safety area **then** `Action Avoid Teammate` is active.
- *Macro-State MS2*: The distance between the robot and the border is smaller than the Visibility Range and larger than a given threshold, $\zeta_{FS}$
  - **if** no teammate is in the safety area **then** `Action Keep Going` is active,
  - **if** one or more teammates are in the safety area **then** `Action Avoid Teammate` is active.
- *Macro-State MS3*: The distance between the robot and the border is smaller than a given threshold, $\zeta_{FS}$
  - **if** no teammate is in the visibility range **then** `Action Keep Going` is active,
  - **if** there is a teammate on the left **then** `Action Patrol Clockwise` is active,
  - **if** there is a teammate on the right **then** `Action Patrol Counter-Clockwise` is active.

Finally, it is worth noticing that, when the system has enough degrees of mobility, a more advanced Supervisor could be designed to further combine, via NSB, the actions' outputs in order to achieve more sub-goals at once. For instance, when a Fuzzy Logic Supervisor is used, several actions can fire at once with different degrees of activation. These can be used as priority of the fired actions in the NSB approach.

## 7 Analysis

The FSA has been designed according to guidelines described in Murphy [31, pp. 163–209]. The main properties that have been considered and checked are the *consistency* and *completeness* [33]. Namely, a FSA is said to be *consistent* if only a single transition rule can be enabled at the same time for all the states. By considering the structure of the supervisor, it can be verified that this property holds.

**Proposition 7.1** *The FSA shown in Fig. 2 is consistent.*

*Proof* Since the supervisor is arranged in a hierarchal way with three levels (Fig. 2), with scenarios at the top layer, macro tasks in the middle layer and actions at the lowest layer, it is required to prove that in each time instant:

1. only one scenario can be active,
2. for each scenario, only one macro task can be active,
3. for each macro-task, only one action can be active.

1) *Consistency of the Scenario Layer*
With regards to the scenarios, the following transition has been defined:

- `C0`: there is a friend in the safety area of the considered robot;

this condition, essentially, identifies two situations depending on whether a friend vehicle is *close* or not to the given patrolling vehicle. Since the two conditions are mutually exclusive, it is obvious that only one scenario can be active at a time.

2) *Consistency of the Macro-Task Layer*
With regards to the macro-tasks, there is only a macro-task in the first scenario (`MS0`), thus, no conflict occurs in this scenario. With regards to the second scenario, the following transitions for the macro-tasks (`MS1, MS2, MS3`) have been defined:

- `C1`: the distance from the border is smaller than the visibility range,
- `C2`: the distance from the border is smaller than the threshold $\zeta_{FS}$ defined in Section 6.

From each of the considered macro-tasks, only one output transition is defined, thus, conflicts cannot occur.

3) *Consistency of the Actions Layer*
In the case of the actions in each macro-state, the following situations have to be considered:

- the macro-state `MS0` is active. In this case, only the action `Action Avoid Friend` can be active;
- the macro-state `MS1` is active. Two actions might be active (`Action Reach Frontier` and `Action Avoid Teammate`). Only one transition rule, namely `c1`, is present; this means that conflicts do not arise;
- the macro-state `MS2` is active. Two actions might be active (`Action Keep Going` and `Action Avoid Teammate`). Also in this case, only one transition, namely `c1`, is present; thus, any conflict is avoided;
- the macro-state `MS3` is active. Three actions might be active (`Action Keep Going`, `Action Patrol Clockwise` and `Action Patrol Counter-Clockwise`). Three transition rules are present, namely `c2, c3, c4`. Because `c2` and `c3` cannot both be true at same time, the following combinations are admitted:

| c2 | c3 | c4=!c2 ∧ !c3 |
|---|---|---|
| 1 | 0 | 0 |
| 0 | 1 | 0 |
| 0 | 0 | 1 |

this implies that only one transition rule is active at a time. In addition, each action is characterized by only one output transition. Thus the consistency is proved. □

Moreover, a FSA is said to be *complete* [33] if it operates correctly for all possible input/state sequences.

**Proposition 7.2** *The FSA shown in Fig. 2, combined with the behavior-action level and the properties of the NSB approach, is complete.*

*Proof* Completeness can be inferred from the main NSB properties. In the environmental model that has been considered, in each time instant the *inputs* a vehicle takes into account are the distance from the border, the distance from the friend vehicles and the distance from teammates or other obstacles. Based on the macro state `MSi (i=0,1,2,3)`, any sequence of the considered inputs is admissible for the designed FSA.

In fact, any sequence of the following situations is admissible:

- a friend is in the visibility area. In this case, the macro state MS0 is active and then the Action Avoid Friend. As stated in Section A.2.5, this action guarantees that no collision between friends occurs;
- no friend and no obstacle is in the visibility area. In this case, either the vehicle is in the macro states MS1 or MS2. Thanks to the Action Reach Frontier (see Section A.2.1) it is ensured that the vehicle gets to the border and starts the patrolling mission;
- an obstacle is in the visibility area. In this case, the robot is either in the macro state MS1 or MS2. The Action Avoid Teammate in Section A.2.4 guarantees that no collision occurs, since the collision avoidance has the highest priority task in the NSB sense. Thus completeness is proven. □

In sum, in the case of one robot, the FSA ensures the robot reaches the border and starts the patrolling mission (in macro-state MS3). In fact, whatever is the starting macro-state, since Reach Frontier is always the primary behavior (in the absence of obstacles or other teammates), the robot is forced to reach the border (i.e., the FSA is in the macro-state MS3). In the case of multiple robots (but in the absence of friends), the transition towards MS3 occurs if there is enough space along the border to let the robot stay on it. The behavior of robots left out of the border (whose supervisor is in macro-states MS1 or MS2), thanks to the distinction between macro-state MS2 and MS3, do not affect the patrolling robots. In the presence of friend vehicles and independently from the other conditions, the macro-state MS0 becomes active. In this case, the Reach Frontier Behavior is not the primary behavior; hence, the robot reaches the border only after the friend vehicles have crossed it (the behaviors composing Action Avoid Friend are not compatible [3]).

### 7.1 Advantage of the use of NSB

The considered task priority algorithm guarantees that the higher priority tasks are not affected by the lower ones. For the generic task, in fact, it has been demonstrated that its error converges to zero within the null-space of the higher ones and it is not affected by the lower ones [5]. The actions used in this paper are properly defined as discussed in the Appendix; in addition, tuning of the feedback gains is extremely simplified with respect to cooperative behavioral approaches. In fact, the NSB decouples the tasks spaces, meaning that the parameter tuning can be done for each task independently from the remaining ones and independently from its priority. This property is a significant advantage with respect to any cooperative behavior approach. In fact, the coupling among the tasks/behavior is done by selecting the priority, thus at a higher level, and not by selecting the gains.

## 8 Simulations

Several simulations, performed in the presence of both closed and open borders with different sizes and shapes, have been carried out by using both Matlab [30] and Player/Stage [39] environments. A video of the simulation can be downloaded at the footnote link.[1]

### 8.1 Simulations in the Presence of a Large Number of Robots

In order to test the approach in the presence of a large number of robots in the team, a free environment with a closed border has been considered in a numerical simulation. The team is composed by 60 robots, with visibility and safety areas equal to 20 m. The matrix gains in Eqs. 10–12 of Appendix A.1 have been chosen

[1] A video of the simulation is available at: http://webuser.unicas.it/lai/robotica/video/SimPatrolling.avi.

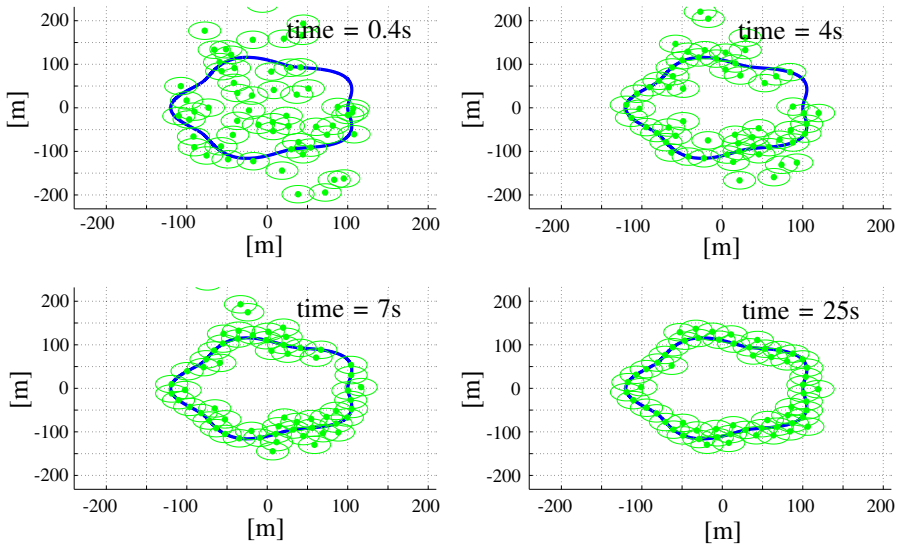


**Fig. 3** Sample frames at different time instants of robots performing a patrolling mission without friend agents

as $\lambda_{rf} = 5\boldsymbol{I}$, $\lambda_{cw} = 8\boldsymbol{I}$, $\lambda_{ccw} = 8\boldsymbol{I}$, $\lambda_{ta} = 10\boldsymbol{I}$ and $\lambda_{fa} = 10\boldsymbol{I}$. The gains have been selected by setting the desired velocities to be assumed when approaching and patrolling the border, as well as when escaping from teammate and friend agents. Friend vehicles are not present in this simulation.

Figure 3 shows the robot positions at four different time instants. Robots are represented by green points surrounded by their visibility ranges; the continuous blue line represents the closed border to be patrolled. As it can be seen, the task is executed in four different phases (each corresponding to a frame in Fig. 3):

- in the first phase, all robots are trying to reach the border, while avoiding teammates;
- in the second phase, some robots reach the border and start patrolling the border;
- in the third phase, most of the robots are patrolling the border,
- in the fourth phase, the robots reach the maximum density along the border, and the patrolling robots are not affected by other agents trying to reach the border.

It is useful to remark that a swarm approach should also prevent robot collisions in critical conditions like this one. To this aim, in Fig. 4 (top), the time history of the minimum value of the distances among all possible robot pairs is depicted, i.e.:

$$d_{r,\min}(t) = \min_{\forall i \neq j\ i,\, j \in \mathcal{N}_r} \parallel p_{i,r}(t) - p_{j,r}(t) \parallel, \tag{7}$$

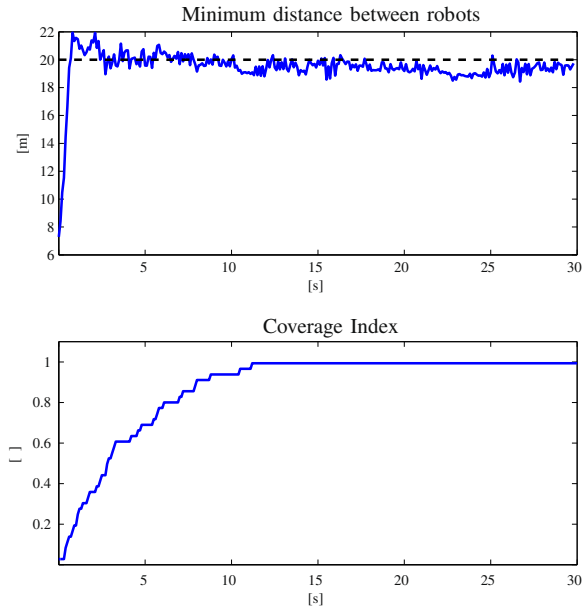


**Fig. 4** *Top*: time history of minimum of distances between all possible robot pairs. The *dashed line* represents the radius of the safety area. *Bottom*: time history of the coverage index over time

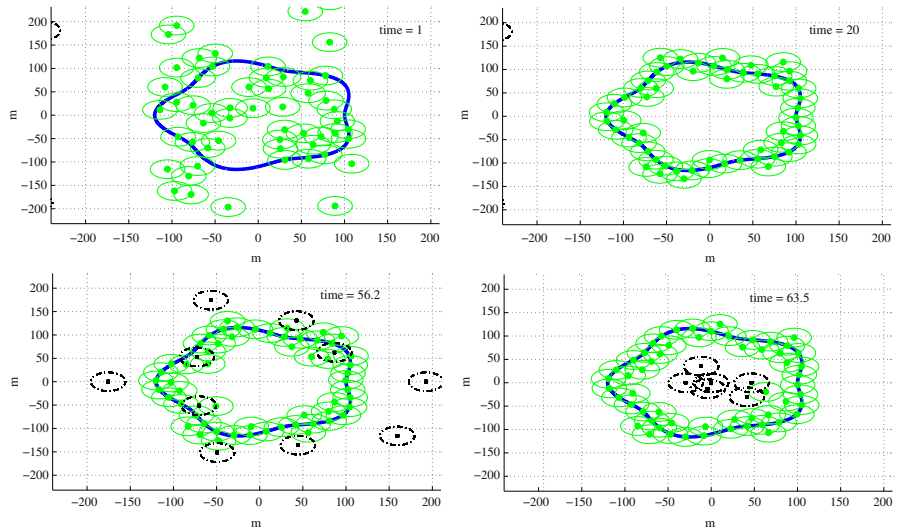


**Fig. 5** Sample frames at different time instants of robots performing a patrolling mission with friend agents. *Dashed vehicles* are the friend agents

where $p_{i,r}(t)$ is the position of the $i$-th robot at instant $t$, and $\mathcal{N}_r$ is the set of patrolling robots.

As it can be seen in Fig. 4 (top), even in the case of a large number of robots, collisions are avoided. Moreover, the structure of the supervisor avoids interference between robots. In fact, the fourth frame in Fig. 3 shows that robots still approaching the border do not affect the motion of robots already performing the patrolling mission (i.e., robots in the macro-state `MS3`). In Fig. 4 (bottom), the time history of a coverage index of the border (defined as the ratio between the portion of border that is in the visibility area of the patrolling robots and the overall length of the border) is shown. As it can be seen, it monotonically approaches the maximum value 1.

## 8.2 Simulations in the Presence of Friend Vehicles

Simulations showing the approach performance in the presence of 10 friend agents (trying to enter the border) are carried out. The team is composed by 60 robots with visibility and safety areas equal to 20 m. The gains are the same as in the previous simulations. Figure 5 shows the robot positions at different time instants. As in the previous case, the different phases (each corresponding to a frame in the figure) are the following:

- in the first phase, robots are randomly distributed and try to reach the border, while avoiding other teammates;

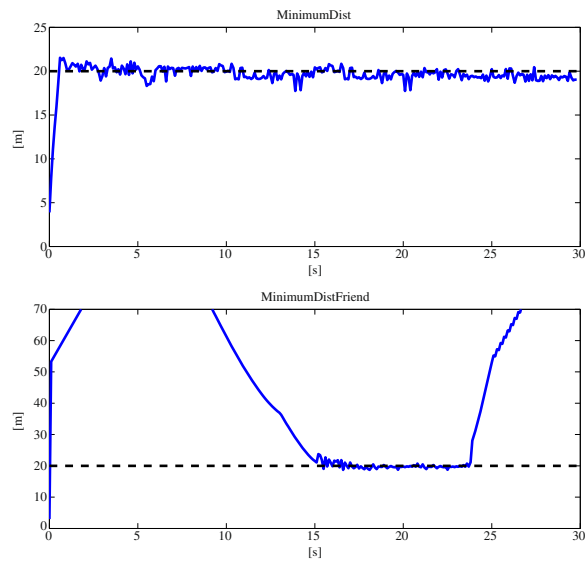


**Fig. 6** *Top*: time history of minimum of distances between all possible robot pairs. The *dashed line* represents the radius of the safety area. *Bottom*: time history of the minimum of distances between all robot-friend pairs

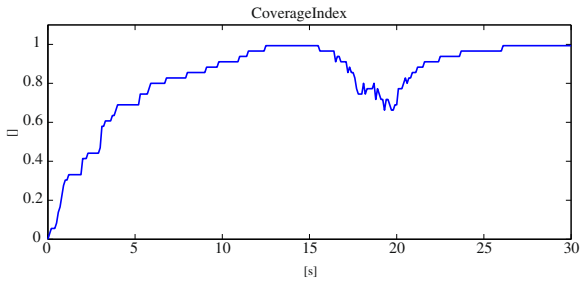


**Fig. 7** Time history of the coverage index over time in the presence of several friend vehicles

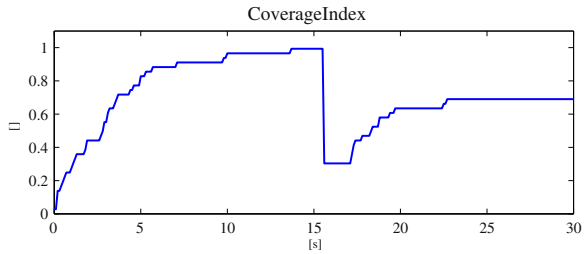


**Fig. 9** Time history of the coverage index over time in the case of several faults

- in the second phase, some robots have reached the border and start to perform the patrolling mission, activating `Action Keep Going` and `Action Patrol Clockwise` (or `Patrol Counterclockwise`);
- in the third phase, friend agents, represented by cross markers surrounded by their safety areas (dash-dot circles), start to approach the border and cross it, without any collision with patrolling robots;
- in the fourth phase, friends gain the center of the bordered zone; in this situation, patrolling robots are no longer affected by friends' motion.

In Fig. 6 (top), the time history of the minimum value of the distances among all the possible robot pairs, computed according to Eq. 7, is depicted.

In Fig. 6 (bottom), the minimum distance over time of all the possible robot-friend pairs is shown, i.e.:

$$d_{f,\min}(t) = \min_{\forall i \in \mathcal{N}_r,\, j \in \mathcal{N}_f} \| \, p_{i,r}(t) - p_{j,f}(t) \, \|, \qquad (8)$$

where $p_{j,f}(t)$ is the position of $j$-th friend agent at the time instant $t$ and $\mathcal{N}_f$ is the set of friend agents. Friend agents cross the border between 15 s and 24 s; even during border crossing the safety distance (dashed line) is not violated. Thus, the proposed architecture is capable of avoiding collisions, and the mission is safely accomplished.

In Fig. 7, the time history of the coverage index is shown. Again, since interference situations do not occur, the coverage index monotonically approaches 1; the index decreases only when friends cross the border.

### 8.3 Simulations in the Presence of Friend Vehicles and Multiple Sudden Faults

In this simulation case study, the initial conditions and the parameter values are the same as in the previous one. In addition, robot failures occur. Namely, 50 % of the initial patrolling robots suddenly fail. As in the previous simulations, at the beginning, robots are randomly distributed and reach a final configuration corresponding to the fourth frames of Figs. 3 and 5. Starting from this configuration, Fig. 8 (left) shows the configuration of robots immediately after the occurrence of the faults. The second frame shows that the

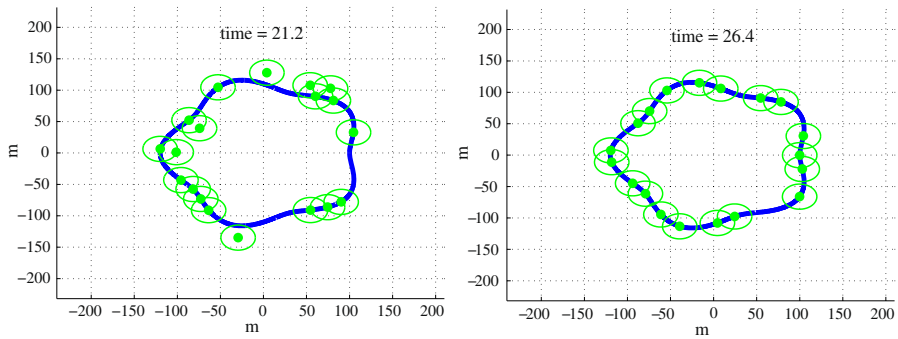


**Fig. 8** In the *left frame*, several robots fail. In the *right frame*, the remaining robots automatically redistribute around the border

**Fig. 10** The experimental setup at the Distributed Intelligence Laboratory of University of Tennessee. *Left*: a Pioneer-3DX mobile robot. *Right*: the team used in the experiments

surviving robots redistribute along the border after the failures, thus keeping the mission objective. In Fig. 9, the time history of the coverage index is shown. Initially, the index almost reaches the value 1; at 15.5 s, when 50 % of the robots suddenly fail, the remaining robots redistribute to maximize the coverage index.

### 8.4 Discussion

The numerical simulations have been designed in order to stress the algorithm in severe tests. Dozens of different case studies with different simulation parameters such as, e.g., the number of robots, the visibility range, the robot speed, etc., have been run. The results show that, even in the presence of a large number of robots, collisions and interference between robots are avoided. The results confirm that the approach is capable of achieving nearly optimal values of the adopted performance indexes. The presence of a generic number of friend robots in a cloud configuration is also handled in a proper way, while ensuring satisfactory performance. Finally, simultaneous faults of several robots affect significantly the performance only during a transient phase; the expected values for the given number of working robots are then promptly reached.

## 9 Experiments

The proposed control architecture has been implemented on an experimental setup composed of three Pioneer-3DX mobile robots (Fig. 10), available at the Distributed Intelligence Laboratory of University of Tennessee. Videos about the experiments can be downloaded at the footnote links.[2]

The Pioneer-3DX robot has a 0.44 m × 0.38 m × 0.22 m aluminum body with 0.165 m diameter. This differential drive platform is non-holonomic and can rotate in place by moving both wheels, or it can swing around a stationary wheel on a circle of 0.32 m radius. A rear castor balances the robot. In addition to motor encoders, the robot base includes eight ultrasonic transducers (range-finding sonar) arranged to provide a 180-degree forward coverage at a sampling rate of 25 Hz. The computational board includes a 32-bit RISC-based controller running Linux. Additional hardware includes ethernet communication, a Wi-Fi radio turret for remote control, a Pan-Tilt-Zoom color camera and a laser rangefinder. Since the orientation $\theta$ of the robot is not of interest and our aim is not to solve the control problem for non-holonomic systems, the following direct task model has been considered

$$\dot{\boldsymbol{p}}_b = \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -b\sin(\theta) \\ \sin(\theta) & +b\cos(\theta) \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{9}$$

where $\boldsymbol{p}_b$ is a point whose distance from the wheels' axis is equal to $b$. Taking into account the velocities generated by the control algorithm, the dimensions and the maximum linear and angular velocities of the Pioneer-3DX robots, $b$ has

[2]The videos of the experiments on an open and closed border are available at: http://webuser.unicas.it/lai/robotica/video/ExpOpenBorder.avi and http://webuser.unicas.it/lai/robotica/video/ExpClosedBorder.avi.

been chosen equal to 0.1 m. This value prevents saturation of the actuators and ensures a control point located inside the robot chassis.

The Player/Stage environment [39] has been used to implement the proposed architecture. Player/Stage is based on a client-server paradigm; it provides a set of tools to simulate the system (Stage), as well as the software primitives for communicating with the robotics hardware (Player). In Fig. 11 (left) a portion of the patrolled area is shown, together with its representation in Player/Stage (right); in addition, an open (top) and closed (bottom) borders have been considered. The blue line represents the border, the red, green and cyan polygons represent the robots, together with their visibility ranges (assumed to be equal to the corresponding safety areas). The open border (top) is 10 m long; the closed border (bottom), composed by segments joined by arcs, has an overall length of 51 m. The results in the following refer to the closed border; nevertheless, similar results have been obtained for the open border. The robots know the exact description of the border and they approach it at a speed of 0.3 m/s ($\lambda_{rf} = 0.3$), patrol at a speed of 0.35 m/s ($\lambda_{cw} = \lambda_{ccw} = 0.35$) and escape other teammates at a speed of 0.3 m/s ($\lambda_{ta} = 0.3$). Localization in the environment is achieved via a pre-built map and a localization driver based on an adaptive particle filter [20], available within the Player software. The visibility range and safety area are equal to 2.5 m, the threshold value $\zeta_{FS}$ for the transition between macro-states `MS2` and `MS3` is 0.6 m. Moreover, if a robot is in the patrolling state, every 30 s it can decide to invert its motion direction or to keep going in the same one, according to a random variable.

Figure 12 (top) shows the time history of the distance from the border for the three robots. In normal operating conditions (i.e., in the absence of faults), the mutual distances are within 0.1 m, this value is acceptable for the experimental conditions and assumed requirements. Moreover, peaks are reached during rotation movements due to sensor noise and, above all, to the neglected robot dynamics in the control law.

When a fault occurs to robot number 2, after 28 min from the mission start, the robot is manually driven far from the border and is reactivated at minute 41. Two faults occur to robot number 3, after 15 and 33 min from the mission start. As can be noticed in Fig. 12, thanks to the decentralized

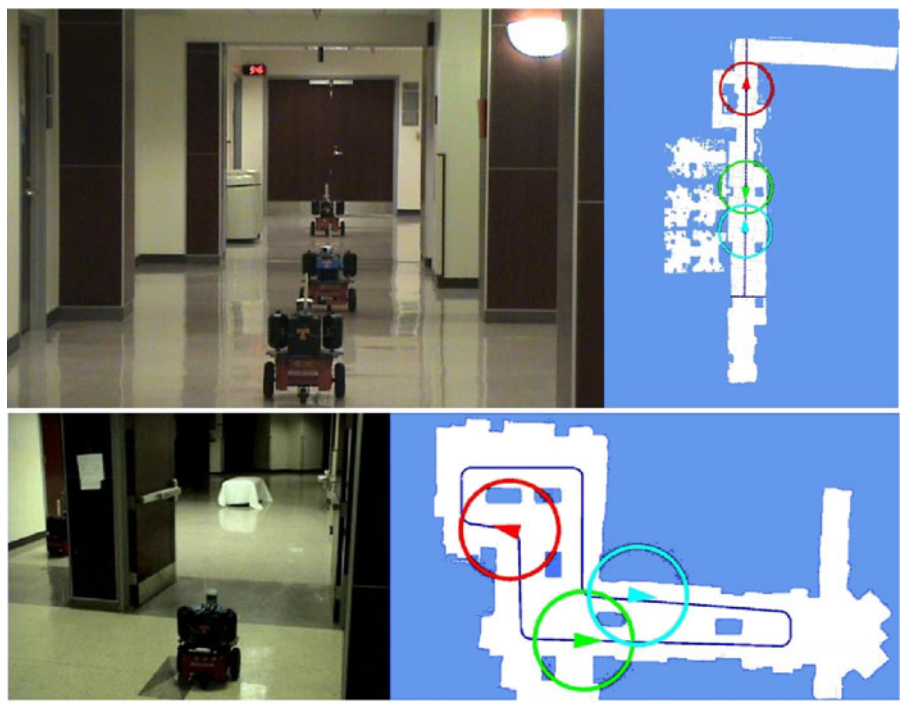

**Fig. 11** *Left*: a portion of the environment. *Right*: environment representation in Player/Stage, the path and the three patrolling robots

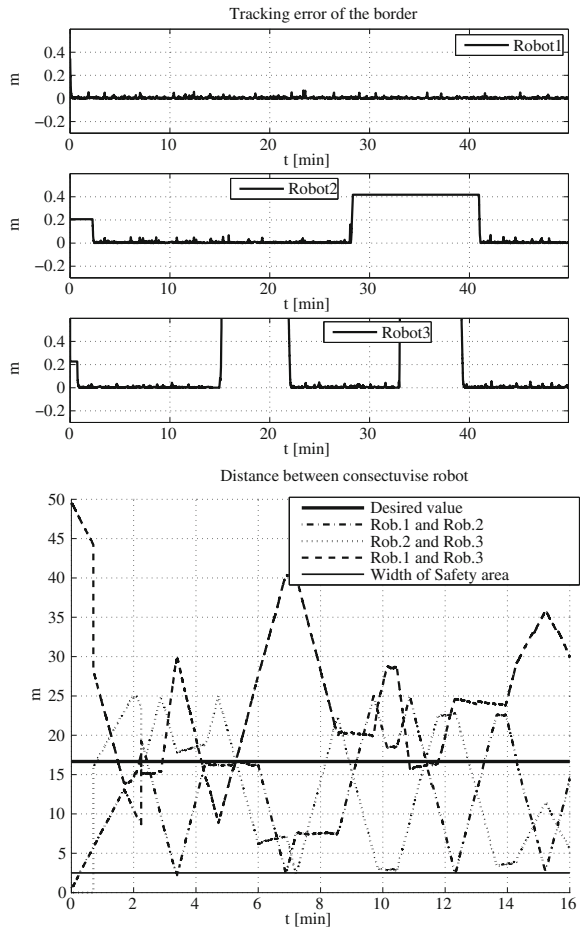


**Fig. 12** *Top*: distance from the border. *Bottom*: distance between two consecutive robots

structure of the control law, faults do not affect the overall behavior of the swarm. For the sake of clarity, it is worth showing the time history of the distance between two consecutive robots (Fig. 12, (bottom)).

### 9.1 Discussion

In addition to intensive tests based on numerical simulation, the approach has been also validated by performing experiments with 3 robots. The experiments, even if involving a limited number of robots, have been conducted without particular shortcuts (e.g., an ad-hoc localization method). The faults have been emulated by turning on/off the robots during the patrolling of the remaining ones. The experiments' duration has been limited only by battery life. The achieved results clearly show that the proposed approach is also practically viable.

According to Chevaleyre [13], at each time instant, the distance between two consecutive robots should be one third of the overall border length; moreover, all robots should move in the same direction. On the contrary, according to Agmon et al. [1], robots should move synchronously but in a nondeterministic way, so as to maximize the probability of intercepting an intruder. Both cases require a centralized supervisor or information exchanges between robots, so it is straightforward to imagine that the proposed approach can be characterized by better performance only by increasing the number of vehicles or by removing some constraints.

## 10 Conclusion

A higher layer above the traditional Null-Space-Behavioral (NSB) approach has been designed. At the basis of the developed architecture, there is the concept of action, obtained by combining elementary behaviors in the NSB framework. Once the set of actions has been defined, a Supervisor can be designed that chooses the action to be executed. The developed architecture has been applied to the challenging border patrolling mission. To this aim, a set of constraints and requirements have been formulated and the mission achieved thanks to the designed architecture. The algorithm is fully scalable; the computational load, in fact, is simply independent of the number of robots as no communication occurs between robots and the control strategy of each robot does not depend on the number of robots. The robustness, in a wide sense, has been designed by avoiding the need for a central computational unit or dead-lock communication among the robots. This intrinsic robustness of the approach has been confirmed by simulation and experimental results. Future work will be devoted to the application of the developed architecture to different missions, e.g., coverage and flocking, by properly redefining the behavior and action sets. Moreover, it would be of utmost importance to test the approach by adopting different classes of vehicles (e.g., unmanned

underwater vehicles and/or unmanned aerial vehicles) and in the case of heterogeneous teams of robots.

Comparison of control strategies for unstructured robotics is an open issue in the robotics community. The European network EURON has a specific committee [18] and the American NIST (National Institute of Standards and technologies) has a specific annual conference on it (PERMIS, Performance Metrics for Intelligent Systems Workshop: Permis [34]). Hence, future work might be devoted at investigating comparison metrics in this compelling case study.

## Appendix

### A.1 Elementary Behaviors

In the following the elementary behaviors for the patrolling mission are defined and described in the framework of the NSB. Definition of behaviors depends, of course, on the mission to be executed. In this paper, therefore, they will be determined by identifying the elementary components required by patrolling missions.

#### *A.1.1 Reach Frontier*

This behavior pushes the robot towards the border to be patrolled (Fig. 13).

Then, given the robot position $\boldsymbol{p}_r \in \Re^2$ and the border $B$, $\boldsymbol{p}_B \in \Re^2$ is the closest point to $\boldsymbol{p}_r$ belonging to $B$. The behavior is then encoded by the function $\sigma_{rf}$

$$\begin{cases} \sigma_{rf} = \|\boldsymbol{p}_r - \boldsymbol{p}_B\|, \ \sigma_{rf,d} = 0, \\ \boldsymbol{J}_{rf} = \boldsymbol{r}_{rf}^T, \ \boldsymbol{J}_{rf}^\dagger = \boldsymbol{r}_{rf}, \ \boldsymbol{N}_{rf} = \boldsymbol{I}_2 - \boldsymbol{r}_{rf}\boldsymbol{r}_{rf}^{\mathrm{T}}, \\ \boldsymbol{v}_{rf} = -\lambda_{rf}\boldsymbol{r}_{rf}\sigma_{rf}, \end{cases} \tag{10}$$

where $\sigma_{rf,d}$ is the desired value of the task function, $\boldsymbol{r}_{rf} = (\boldsymbol{p}_r - \boldsymbol{p}_B)/\|\boldsymbol{p}_r - \boldsymbol{p}_B\|$, $\boldsymbol{J}_{rf}$ is the task Jacobian, $\boldsymbol{I}_2$ is the $(2 \times 2)$ identity matrix, $\boldsymbol{N}_{rf}$ is the null-space projection matrix, $\lambda_{rf}$ is a positive scalar gain and $\boldsymbol{v}_{rf}$ is the commanded velocity.

It is worth noticing that the computation of the point on the border closest to the robot, $\boldsymbol{p}_B$, is needed. Hence, a discretization of the border and/or a proper analytical approximation of its shape [1] are required.

Of course, boundary detection and boundary tracking capabilities are required to accomplish this behavior. However, such problems are behind the scope of this paper and were widely discussed in other works (e.g., [22, 24, 40]).

#### *A.1.2 Patrol Frontier Clockwise (Counter-Clockwise)*

Once the robot is close to the border, it should move according to the border shape. To this aim, given the border $B$ and a point $\boldsymbol{p}_B$ belonging to $B$, $\boldsymbol{r}_{cw}$ is the unit vector tangent to the border in $\boldsymbol{p}_B$ and oriented in the clockwise direction of the border. The behavior is, then, directly defined as:

$$\begin{cases} \boldsymbol{v}_{cw} = \lambda_{cw}\boldsymbol{r}_{cw}, \\ \boldsymbol{J}_{cw} = \boldsymbol{r}_{cw}^{\mathrm{T}} \\ \boldsymbol{N}_{cw} = \boldsymbol{I}_2 - \boldsymbol{r}_{cw}\boldsymbol{r}_{cw}^{\mathrm{T}}, \end{cases} \tag{11}$$

where $\boldsymbol{v}_{cw}$ is the velocity vector encoding the behavior, $\boldsymbol{r}_{cw}^T$ plays the role of the task Jacobian, $\boldsymbol{N}_{cw}$ is the null-space projection matrix and $\lambda_{cw}$ is a positive scalar gain. It is worth noticing that the behavior simply commands the robot to move along the direction determined by the border tangent (Fig. 14), no matter what the robot position is (along the border or out of the border). If the vector tangent to the border is oriented in the counter–clockwise direction we can define `Patrol Frontier Counter-Clockwise` behavior (by replacing $\lambda_{cw}$ and $\boldsymbol{r}_{cw}$ with $\lambda_{ccw}$ and $\boldsymbol{r}_{ccw}$ in Eq. 11, $\boldsymbol{v}_{ccw}$ and $\boldsymbol{N}_{ccw}$ are easily obtained).

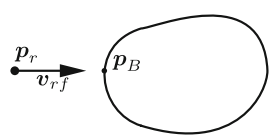


**Fig. 13** Graphical representation of the `Reach Frontier Behavior`

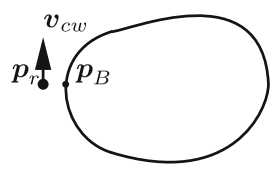


**Fig. 14** Graphical representation of the `Patrol Clockwise Behavior`

*A.1.3 Teammate Avoidance*

In order to avoid collisions between robots, a suitable behavior has to be defined. Let $\boldsymbol{p}_r$, be the robot position, $\boldsymbol{p}_t$ the position of the closest teammate to the robot and $d_t$ the safety distance, i.e., the radius of the circular safety area. The behavior is, then, defined as follows:

$$\begin{cases} \sigma_{ta} = \|\boldsymbol{p}_r - \boldsymbol{p}_t\|, \ \sigma_{ta,d} = d_t, \\ \boldsymbol{J}_{ta} = \boldsymbol{r}_{ta}^{\mathrm{T}}, \ \boldsymbol{J}_{ta}^{\dagger} = \boldsymbol{r}_{ta}, \ \boldsymbol{N}_{ta} = \boldsymbol{I}_2 - \boldsymbol{r}_{ta}\boldsymbol{r}_{ta}^{\mathrm{T}}, \\ \boldsymbol{v}_{ta} = \lambda_{ta}\boldsymbol{r}_{ta}\left(d_t - \sigma_{ta}\right), \end{cases} \tag{12}$$

where $\sigma_{ta,d}$ denotes the desired value of the behavior function, $\boldsymbol{r}_{ta} = \left(\boldsymbol{p}_r - \boldsymbol{p}_t\right)/\|\boldsymbol{p}_r - \boldsymbol{p}_t\|$, $\boldsymbol{J}_{ta}$ is the task Jacobian, $\boldsymbol{N}_{ta}$ is the null-space projection matrix, $\lambda_{ta}$ is a positive scalar gain and $\boldsymbol{v}_{ta}$ is the commanded robot velocity.

Hence, this behavior allows the robot to keep the other teammates out of the border of its own safety area (Fig. 15).

*A.1.4 Friend Avoidance*

A *friend* is an agent that moves independently from patrolling robots and that is allowed to cross the border. Therefore, when a friend tries to cross the border, patrolling robots should keep a desired distance from it, without affecting its motion. This behavior is similar to the `Teammate Avoidance` behavior; then, given the robot position, $\boldsymbol{p}_r$, the friend position, $\boldsymbol{p}_f$, and a safety distance $d_f$, the behavior is defined as follows

$$\begin{cases} \sigma_{fa} = \|\boldsymbol{p}_r - \boldsymbol{p}_f\|, \ \sigma_{fa,d} = d_f, \\ \boldsymbol{J}_{fa} = \boldsymbol{r}_{fa}^{\mathrm{T}}, \ \boldsymbol{J}_{fa}^{\dagger} = \boldsymbol{r}_{fa}, \ \boldsymbol{N}_{fa} = \boldsymbol{I}_2 - \boldsymbol{r}_{fa}\boldsymbol{r}_{fa}^{\mathrm{T}}, \\ \boldsymbol{v}_{fa} = \lambda_{fa}\boldsymbol{r}_{fa}\left(d_f - \sigma_{fa}\right), \end{cases} \tag{13}$$

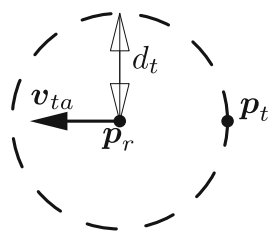


**Fig. 15** Graphical representation of the `Teammate Avoidance Behavior`. The *dashed line* represents the border of the safety area

where $\sigma_{fa,d}$ denotes the desired value of the behavior function, $\lambda_{fa}$ is a positive scalar gain, $\boldsymbol{N}_{fa}$ is the null-space projector matrix and $\boldsymbol{v}_{fa}$ is the commanded robot velocity. Similarly to the previous case, this behavior, keeping the robot far from the friend agent, allows it to move without being affected by patrolling agents.

A.2 Actions

The above defined behaviors are not yet suitable to accomplish the patrolling mission, since they only encode simple atomic tasks. For example, the `Reach Frontier` behavior allows the robot to reach the border, while the two `Patrol Frontier` behaviors only allow the robot to move in a direction tangent to the border, but not necessarily on the border. Then, once the set of actions is defined, it is necessary to combine them to produce a set of more complex and meaningful actions. This combination is obtained via NSB, in order to obtain a predictable output; thus, priorities among the elementary behaviors are to be assigned.

*A.2.1 Action Reach Frontier*

This action allows the robot to reach the border, e.g., when it is far from it. In this case, the definition of the action simply coincides with the elementary behavior `Reach Frontier`:

$$\boldsymbol{v}_{Arf} = \boldsymbol{v}_{rf}. \tag{14}$$

This action is useful when the robot is particularly far from the border. In such a situation, the only objective of the robot is to reach the frontier before starting other behaviors; hence, no secondary tasks are present.

*A.2.2 Actions Patrol Clockwise (Counter-Clockwise)*

This action allows the robot to stay on the border, while covering it in the clockwise direction. To this aim, two behaviors are needed. The primary behavior is `Reach Frontier`, while the secondary

one is `Patrol Frontier Clockwise`, leading to

$$\boldsymbol{v}_{Apcw} = \boldsymbol{v}_{rf} + \boldsymbol{N}_{rf}\boldsymbol{v}_{cw}. \tag{15}$$

By activating this action, the robot is able to reach or keep on the border while patrolling it. As will be shown in Section 5.3, since the two elementary behaviors composing the action are compatible in the NSB sense, the priority of behaviors can be exchanged. In Fig. 16, a typical robot motion under the effect of `Action Patrol Clockwise` is shown.

While, in the case of `Action Patrol Counter-Clockwise` the action is obtained considering the `Patrol Frontier Counter-Clockwise` behavior as secondary behavior. With regards the compatibility in the NSB sense of the defined behaviors, it easy to show that the compatibility condition [3] $\boldsymbol{J}_{rf}\boldsymbol{J}_{cw}^{\dagger} = \boldsymbol{0}$ ($\boldsymbol{J}_{rf}J_{ccw}^{\dagger} = \boldsymbol{0}$) globally holds.

### *A.2.3 Action Keep Going*

This action allows the robot to stay on the border, while covering it in the clockwise or counter–clockwise direction. This action is obtained by combining the `Reach Frontier` and the `Patrol Frontier Clockwise` (or `Patrol Frontier Counter-Clockwise`) behaviors in the NSB sense, i.e.,

$$\boldsymbol{v}_{Akg} = \boldsymbol{v}_{rf} + \boldsymbol{N}_{rf}\boldsymbol{v}_{p}, \tag{16}$$

where $\boldsymbol{v}_p$ is the vector tangent to the border at the closest point belonging to the border. The value of the velocity vector $\boldsymbol{v}_p$ is decided according to some criteria. For example, it can be varied, depending on the value of a random variable, every $T$ seconds and set equal to $\boldsymbol{v}_{cw}$ or $\boldsymbol{v}_{ccw}$. This can be useful for conferring some unpredictability to the mission. The compatibility of the two behaviors can be demonstrated via the same arguments used for the previous action.

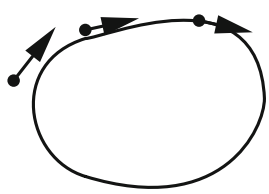

**Fig. 16** Typical robot motion under the effect of `Action Patrol Clockwise`

### *A.2.4 Action Teammate Avoidance*

A teammate entering the safety area of a robot must be avoided, while the robot tries to stay on the border or to reach it; in this way, it can restart the patrol mission once the teammate vehicle is far enough. This action can be obtained by combining the behaviors `Teammate Avoidance` and `Reach Frontier` in the NSB sense, i.e.,

$$\boldsymbol{v}_{Ata} = \boldsymbol{v}_{ta} + \boldsymbol{N}_{ta}\boldsymbol{v}_{rf}. \tag{17}$$

In Fig. 17, a typical path generated by this action is shown.

Differently from the previous action, it is not possible to ensure task compatibility in all situations. In fact, it is not possible to ensure that the compatibility condition $\boldsymbol{J}_{rf}\boldsymbol{J}_{ta}^{\dagger} = \boldsymbol{0}$ does not hold in general. Only the velocity components of the secondary behavior that do not conflict with the primary behavior will be executed, leading to the robot moving on the border of the teammate safety area (Fig. 17).

### *A.2.5 Action Friend Avoidance*

This action is similar to `Action Teammate Avoidance`. In the case a friend vehicle enters the safety area of a robot, the friend must be avoided, while trying to stay on the border, so that it can restart the patrol mission once the friend is far enough. This action can be obtained by

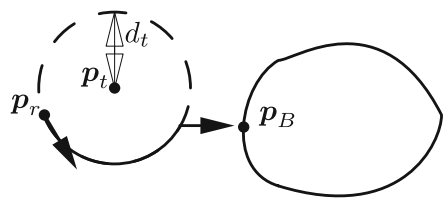


**Fig. 17** Typical robot motion under the effect of `Action Teammate Avoidance`; $\boldsymbol{p}_r$ is the position of the robot, $\boldsymbol{p}_t$ is the position of the teammate closest to the robot, and $\boldsymbol{p}_B$ is the point on the border closest to the robot

combining the `Friend Avoidance` and `Reach Frontier` behaviors

$$\boldsymbol{v}_{Afa} = \boldsymbol{v}_{fa} + \boldsymbol{N}_{fa}\boldsymbol{v}_{rf}. \tag{18}$$

Since `Reach Frontier` is the secondary behavior, only its velocity components that do not conflict with the primary task will be executed. Also in this case, as in the previous one, it is not possible to ensure that $\boldsymbol{J}_{rf}\boldsymbol{J}_{fa}^{\dagger} = \boldsymbol{0}$ holds in any condition; nevertheless, it is guaranteed that no collision happens being the `Friend Avoidance` the highest priority behavior.

## References


1. Agmon, N., Kraus, S., Kaminka, G.A.: Multi-robot perimeter patrol in adversarial settings. In: Proceedings 2008 IEEE International Conference on Robotics and Automation. Pasadena, CA, pp. 2339–2345 (2008)
2. Alur, R., Yannakakis, M.: Model checking of hierarchical state machines. ACM Trans. Program. Lang. Syst. **23**(3), 273–303 (2001)
3. Antonelli, G.: Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. IEEE Trans. Robot. **25**(5), 985–994 (2009)
4. Antonelli, G., Chiaverini, S.: Kinematic control of platoons of autonomous vehicles. IEEE Trans. Robot. **22**(6), 1285–1292 (2006)
5. Antonelli, G., Arrichiello, F., Chiaverini, S.: The null-space-based behavioral control for autonomous robotic systems. J. Intell. Serv. Robot. **1**(1), 27–39 (2008)
6. Antonelli, G., Arrichiello, F., Chiaverini, S.: Experiments of formation control with multirobot systems using the null-space-based behavioral control. IEEE Trans. Control Syst. Technol. **17**(5), 1173–1182 (2009)
7. Arkin, R.: Motor schema based mobile robot navigation. Int. J. Rob. Res. **8**(4), 92–112 (1989)
8. Bertozzi, A.L., Kemp, M., Marthaler, D.: Determining environmental boundaries: asynchronous communication and physical scales. In: Proceedings of the Block Island Workshop on Cooperative Control, vol. 309, pp. 25–42 (2004)
9. Brooks, R.: A robust layered control system for a mobile robot. IEEE J. Robot. Autom. **2**(1), 14–23 (1986)
10. Bruemmer, D.J., Dudenhoeffer, D., Anderson, M.O., McKay, M.D.: A robotic swarm for spill finding and perimeter formation. In: Spectrum 2002: International Conference on Nuclear and Hazardous Waste Management, Reno, Nevada, USA (2002)
11. Bryson, J.J.: Action selection and individuation in agent based modelling. In: Proceedings of Agent 2003: Challenges of Social Simulation, pp. 317–330 (2003)
12. Casbeer, D.W., Kingston, D.B., Beard, A.W., Mclain, T.W., Li, S., Mehra, R.: Cooperative forest fire surveillance using a team of small unmanned air vehicles. Int. J. Syst. Sci. **37**, 360 (2006)
13. Chevaleyre, Y.: Theoretical analysis of the multi-agent patrolling problem. In: Procedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology, Beijing, China, pp. 302–308, 20–24 Sept 2004
14. Chiaverini, S.: Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. IEEE Trans. Robot. Autom. **13**(3), 398–410 (1997)
15. Clarka, J., Fierro, R.: Mobile robotic sensors for perimeter detection and tracking. ISA Trans. **28**, 3–13 (2007)
16. Dorigo, M., Sahin, E.: Swarm robotics—special issue editorial. Auton. Robots **17**, 115–147 (2004)
17. Dudek, G., Jenkin, E., Wilkes, D.: A taxonomy for swarm robots. In: In Proceedings of 1993 IEEE International Conference on Intelligent Robots and Systems, pp. 441–447 (2003)
18. Euron: Special interest group in good experimental methodology and benchmarking (2008). http://www.euron.org/activities/benchmarks/index. Accessed 6 May 2009
19. Everett, H.R., Laird, R.T., Gilbreath, G., Heath-Pastore, T.A., Inderieden, R.S., Grant, K., Jaffee, D.M.: Multiple resource host architecture for the mobile detection assessment and response system. In: Technical Document 3026, Space and Naval Warfare Systems (1998)
20. Fox, D., Burgard, W., Dellaert, F., Thrun, S.: Monte carlo localization: efficient position estimation for mobile robots. In: Proceedings of the National Conference on Artificial Intelligence, pp. 343–349 (1999)
21. GNIUS: Autonomous unmanned ground vehicles (2005). http://www.defense-update.com/products/g/guardium. Accessed 6 Dec 2009
22. Hsieh, M.A., Loizou, S.G., Kumar, V.: Stabilization of multiple robots on stable orbits via local sensing. In: ICRA, Rome, Italy, pp. 2312–2317 (2007)
23. Inderieden, R., Everett, H., Heath-Pastore, T., Smurlo, R.: Overview of the mobile detection assessment and response system. In: DND/CSA Robotics and KBS Workshop, St. Hubert, Quebec (1995)
24. Kalantar, S., Zimmer, U.R.: Distributed shape control of homogeneous swarms of autonomous underwater vehicles. Auton. Robots **22**, 37–53 (2007)
25. Kingston, D., Beard, R., Holt, R.S.: Decentralized perimeter surveillance using a team of UAVs. IEEE Trans. Robot. **24**(6), 1394–1404 (2008)
26. Machado, A., Ramalho, G., Zucker, J., Drogoul, A.: Multi-agent patrolling: an empirical analysis of alternative architectures. In: Multi-Agent Based Simulation, pp. 155–170 (2002)
27. Marino, A., Parker, L., Antonelli, G., Caccavale, F.: Behavioral control for multi-robot perimeter patrol: a finite state automata approach. In: Proceedings 2009 IEEE International Conference on Robotics and Automation. Kobe, J. (2009)

28. Marino, A., Parker, L., Antonelli, G., Caccavale, F.: A fault-tolerant modular control approach to multi-robot perimeter patrol. In: IEEE International Conference on Robotics and Biomimetics (ROBIO 2009), Guilin, China (2009)
29. Marino, A., Parker, L., Antonelli, G., Caccavale, F.: A fuzzy-based multiple robots border patrol. In: Proceedings 2009 17th Mediterranean Conference on Control and Automation, Thessaloniki, Greece (2009)
30. Matlab: http://www.mathworks.com (2010). Accessed 11 Aug 2010
31. Murphy, R.R.: Introduction to AI Robotics, 1st edn. MIT Press, Cambridge (2000)
32. Nakamura, Y., Hanafusa, H., Yoshikawa, T.: Task-priority based redundancy control of robot manipulators. Int. J. Rob. Res. **6**(2), 3–15 (1987)
33. Ouimet, M., Lundqvist, K.: Automated verification of completeness and consistency of abstract state machine specifications using a SAT solver. In: Electronic Notes in Theoretical Computer Science. Proceedings of the Third Workshop on Model Based Testing (MBT 2007), vol. 190(2), pp. 85–97 (2007)
34. Permis: Performance metrics for intelligent systems workshop (2004) www.isd.mel.nist.gov/research_areas/research_engineering/Performance_Metrics/past_wkshp.html. Accessed 8 July 2009
35. Pletta, J.B., Sackos, J.: An advanced unmanned vehicle for remote applications. Sandia National Laboratory Report (1998)
36. Rafael Armament Development Authority Ltd: http://www.rafael.co.il (2006). Accessed 3 Feb 2009
37. Siciliano, B.: Kinematic control of redundant robot manipulators: a tutorial. J. Intell. Rob. Syst. **3**(3), 201–212 (1990)
38. Thrun, S., Leonard, J.J.: Springer handbook of robotics. In: Siciliano, B., Khatib, O. (eds.), Simultaneous Localization and Mapping, Ch. 37, pp. 871–889 Springer, Heidelberg, Germany (2008)
39. Vaughan, R.T., Gerkey, B., Howard, A.: The player/stage project: tools for multi-robot and distributed sensor systems. In: Proceedings of the 11th International Conference on Advanced Robotics, (ICAR), Coimbra, Portugal, pp. 317–323 (2003)
40. Zhang, F., Haq, S.: Boundary following by robot formations without gps. In: Proceedings of 2008 International Conf. on Robotics and Automation, Pasedena, CA, pp. 152–157 (2008)