Università degli Studi della Basilicata
Dipartimento di Ingegneria e Fisica dell'Ambiente

Dottorato di Ricerca in Ingegneria Industriale e dell' Innovazione,
XXII Ciclo
Settore Scientifico Disciplinare: Automatica ING-INF/04

Ph.D. Thesis

A Null-Space-based Behavioral Approach to Multi-Robot Patrolling

Alessandro Marino

Advisor

Prof. Ing. Fabrizio Caccavale

Co-Advisor

Prof. Ing. Gianluca Antonelli

Acknowledgments

At the end of these three years, I would like to thank some people that have been close to me during the course of this thesis.

First, I feel very privileged to have been part for a while of the *Prisma Group*. In particular, I would like to thank Prof. Fabrizio Caccavale, my mentor, for the occasion he offered me and for his support in this years. A special thanks also to Prof. Gianluca Antonelli, like my mentor, he always supported my research with his helpful suggestions and he has been an unlimited source of ideas.

Very helpful have been the support of Prof. Lynne E. Parker, during my visit at the Distributed Intelligence Laboratory of University of Tennesee. She allowed me to study in a international context and heavily contributed to my research during our weekly meeting. During this period, I met a lot of persons, but, in particular, I would like to thank Yuanyan, Yifan and Rasko for the nice time spent together.

Thanks to Francesco Pierri for having been a constant presence *on my right side*, to Paolo Renna, the *secretary* and real chief of the Mechanical Engineering Department, he always helped me about the tedious bureaucracy, and Rocco Padalino for the timely afternoon coffee breaks.

Last but not the last, thanks to my family for having tolerated all my stress periods, giving me the forces to go on.

Abstract

This thesis, that gathers the work carried out by the author in the last three years of research, deals with the development of a patrolling algorithm formulated in the framework of Null-Space-based Behavioral (NSB) control. In particular, it concerns with the coordination control of robotic systems composed by multiple autonomous vehicles aimed at surveilling a given area. It is easy to imagine the advantages that, in performing such a mission, multi-robot systems present respect to single autonomous robots, e.g., improving the mission efficiency in terms of time, quality and robustness. The main problem is how to define the motion command for each vehicle in the team to achieve the mission in a coordinated and safe way, even in the presence of a large number of teammates as well *friend* vehicles and/or *intruders*. In order to achieve such a mission, a centralized formulation is first provided in the following; to overcome the problems related to such a solution, a general decentralized architecture in the NSB framework has been provided, i.e., each robot is able to decide the next action to perform only based on local sensors information. The main idea is to define a set of *elementary behaviors*. These behaviors are composed in the NSB framework in more meaningful *actions*. The main advantage, with respect to other command fusion approaches, is that the output of the produced actions is predictable. After having defined the set of actions, a supervisor needs to be designed to select the action to perform. Two supervisors have been designed: a Finite State and a Fuzzy Logic supervisors. The approach has been developed in a proper mathematical framework and has been tested in simulation and experimentally in different environments at the Distributed Intelligence Laboratory of University of Tennessee, under the supervision of Prof. Lynne E. Parker. The thesis is organized as follow:

- Chapter 1 presents an introduction to Multi-Robot Systems (MRSs). Thus, starting by the definition of an MRS, and after some considerations concerning the meaning of cooperation among multiple vehicles, this chapter tries to present a taxonomy of these systems in order to give an idea of the main aspects concerning MRSs. In the same time, the state of the art of the research in this field is given by describing the main applications of MRSs and the principal control structures.

- Chapter 2 describes in a detailed way the main behavioral approaches to the control of robotic systems: namely the layered control system, as a significant case of competitive approach, and the motor schema control, as a significant case of cooperative approach. Then, the Null-space-Based approach will be introduced in its theoretical and mathematical characteristics, providing some examples for task formulation and combination, remarking the differences from layered control system and the motor schema control.

- In Chapter 3 the patrolling mission is described. Details about the application of the NSB to this kind of mission for multi-robot systems will be presented. In particular, after the definition of the elementary behaviors combined into actions, a finite state and a logic fuzzy supervisors will be described in detail. Finally, the proposed approach will be tested in simulation as well experimentally.

Contents

List of Figures

Chapter 1
Multiple-Robot-Systems

This chapter introduces the basic concepts behind the idea of using teams of autonomous robots to accomplish a task. Moreover, a taxonomy and the state of the art in this research field are presented. Finally, the main topics and contributions of this thesis will be briefly presented.

1.1 Main Aspects of Multiple-Robot-Systems

Starting in early 1980's, the attention of researchers was attracted by the idea of creating groups of mobile robots able to collaborate, in order to accomplish one or more predefined tasks. The basic principle behind this new approach to the robot coordination was directly inspired by the observation of natural systems. In nature, in fact, it is possible to see a group of animals that working together to achieve a common purpose; typical examples can be found in the sea, on the ground and in the air, and more evolved animals can collaborate to perform more complex social behaviors (see Figure 1.1). Ants and termites can work together to build enormous nests or to transport a prey much heavier than the individuals; birds can fly in compact formations during migrations, in order to save energy; fishes organize themselves in schools with thousand individuals to protect from enemies. Predators find it easier to chase down a fish swimming all alone than trying to cut out a single fish from a huge group. Other animals, like lions, are able to organize group of them to perform more complex tasks like hunting; in fact, naturalists observed that in a group of lions some of them have to scare the prey, while some others have to intercept and kill it. Alternatively, they can organize themselves to attack and kill a pray which is too big for a single predator. An example is reported in Figure 1.1, where four lions attaching a gnu can be seen. As the animal behavior has evolved in collaborative direction, in order

Figure 1.1: Example of social animals. Top left: ants collaborate to form a living bridge. Top right: flock of seagulls. Bottom left: batterly fishes schooling. Bottom right: four lions collaborate during hunting.

to increase the survival probability of the species, researchers start thinking that maybe a multi-robot mobile system, where all the agents collaborate, can increase the chances to accomplish a predefined task.
Generally speaking, the term Multi-Robot System (MRS) includes different typologies of robotic systems, e.g., multiple industrial manipulators, mobile robots with manipulators on board, or team of autonomous vehicles; in this thesis, the term will be used referring to a team of cooperating mobile robots (grounded, aerial, underwater or marine surface vessels).

From an engineering point of view, an MRS can improve the effectiveness of a robotic system from the viewpoint of either the performance in accomplishing certain tasks, or the robustness and reliability of the system.
In [41] it is emphasized that several require multi robots to be accomplished. This situation can arise not only when the robots can accomplish different functions, but also when they have the same capabilities. Moreover, even when a single robot can achieve the given task, the possibility of deploying a team of robots can improve the performance of the overall system. With regards to the robustness, in [82] it is emphasized that an MRS can be designed and implemented in such a way to guarantee two important features: adaptivity and fault tolerance.
Adaptivity refers to the ability of an MRS to modify its own behavior over time, depending on changes in the surrounding environment, in the system mission, composition or capabilities, so that the performance of the entire system can either be improved or, at least, not degraded. Fault tolerance is the ability of an MRS to deal with individual robot failures or communication failures, between two or more robotic agents, that may occur at any time during the mission. Robustness is, therefore, the ability of the MRS to be both adaptive and fault tolerant. All these aspects will be deeply discussed in Sections 1.2, 1.3.

The discipline concerning MRS is still relatively young and a common framework is difficult to identify. Interesting contributions towards a classification of the work on MRSs are the surveys by Cao et al. [24] and by Dudek et al. [41]. In [24] several axes for characterizing an MRS are discused, while in [41] a classification of MRSs, more focussed on the communication and computational aspects, is presented.

1.2 A Multi-Robot-System Taxonomy

Based on [50], a taxonomy of Multi-Robot-Systems will be presented in this section. Starting from [24], in [50] six classification dimensions are individuated: *Cooperation, Knowledge, Coordination, Organization, Communication and System Composition*. A schema of this taxonomy is given in Figure 1.2 In addition to these levels, other two more general

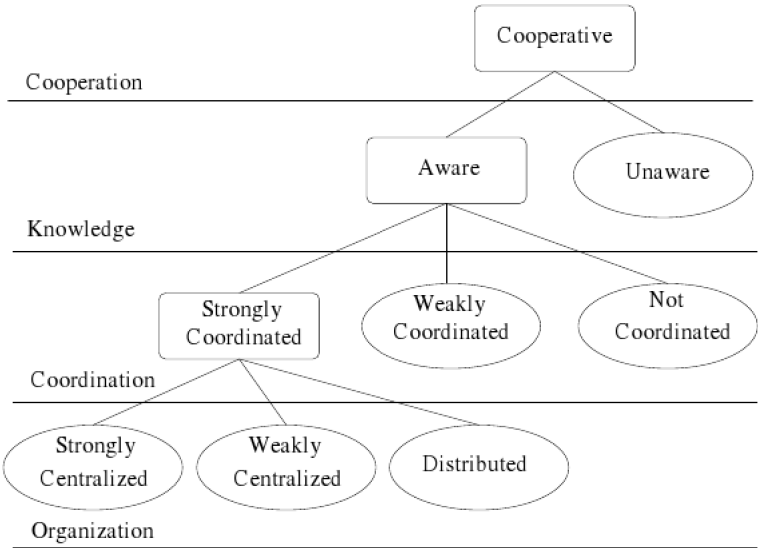

Figure 1.2: A taxonomy of Multi-Robot-Systems.

levels have been indivuated in [50] and described in the following, i.e., the *Communication* and the *Composition* levels, describing, respctively, the way vehicles exchange information, and the type of vehicles forming the team.

Cooperation Level

The first level we will consider is the configuration level. The meaning of cooperation among robots has been widely discussed in the scientific community and different definitions have been proposed. The most widely accepted definitions of cooperation are:

- a joint collaborative behavior that is directed toward some goal; in which there is a common interest or reward [25];
- a form of interaction, usually based on communication [73];
- a join effort for doing something that creates a progressive result, such as increasing performance or saving time [90];

- a situation in which several robots operate together to perform some global task that either cannot be achieved by a single robot, or whose execution can be improved by using more than one robot, thus obtaining higher performances (see [78]).

The first definition leads to the study of task decomposition, task allocation, and other distributed artificial intelligence issues; the second underlines the requirements of communication or other common resources; the third is related to the performance measurements of cooperation; finally, the last definition is based on the motivation behind the use of collaborative systems. Moreover, these definitions point out different aspects of cooperation: the task, the mechanism of cooperation, and the system performance. The task can be considered as the aim of the multi-robot system, thus, it changes depending on the different applications and the typologies of MRS. The task of the system is usually decomposed in elementary sub-tasks (task de- composition) easier to program and control. These sub-tasks can be distributed among multiple resources (task allocation), while the overall behavior of the system depends on how these sub-tasks are recombined to obtain the global behavior of the system. The mechanism of cooperation represents the logic that originates the cooperation and it may depend on the control architectures and strategies, on aspects of the tasks specification or on the dynamics of the interaction among the behaviors. Thus, the MRS has to exhibit a collective behavior or a set of actions that accomplishes the same behavior that was required for the single (more complex) robot. As will be described in the following, to exhibit this cooperative intelligent behavior, the members of the MRS have to communicate directly through an explicit communication channel or indirectly through oindividual robot sensing.

Knowledge Level

Among the cooperative systems, a first important characterization can be based on how much knowledge each robot has about the presence of other robots in its own team. With regards to this aspect, awareness is defined as the property of a robot to have knowledge of the existence of the other members of the MRS [110]. On the contrary, unaware robot performs its task as if it is the only robot in the system. Cooperation among unaware robotic agents is the weakest form of cooperation. For example, in a box-pushing task many robots can contribute to the achievement of the common goal, while behaving as single entities, i.e. without taking into account the presence and the actions of the other robots, as in [60]. Looking at Figure 1.2, among aware systems, a third level

in the taxonomy can be introduced: the Coordination Level, that is concerned with the mechanisms used for robot cooperation.

Coordination Level

The coordination mechanism can be defined as the way each robot takes into account the actions of the other members of the team to achieve the common global goals.
In *strongly* coordinated systems (as in [72] and [54]), each robot exerts its influence on the behavior of other robots, based on predefined or learned rules concerning the way the robots have to interact.
On the contrary, *weakly* coordinated systems do not rely on a predefined coordination protocol. Finally, not coordinated systems do not rely on any form of coordination. Clearly, all these approaches have their advantages and disadvantages. A not coordinated MRS, or a weakly coordinated system implies a less complex design of the overall system and better robustness with respect to faults; but, as a consequence, performance can be reduced because the robots execute contrasting tasks or interferences arise; in a few cases it is not possible to achieve the goals at all. A coordinated MRS, on the contrary, can avoid, or at least reduce, these problems in the face of a more complex design is needed. In sum, the more the environment is dynamic and the goal is complex, the more a strongly coordinated MRS is effective in achieving its goal. For these reasons, hybrid approaches try to be a tradeoff between the coordinated and not coordinated architectures. In particular, their are based on the idea that one or more high level supervisors allocate tasks and resources, and single low level robots exploit local information to accomplish a predefined task. Examples of hybrid architectures are presented in [109] and [51].

Organization Level

In the case of strongly coordinated systems, the organization, defined as the level of centralization of the cooperation mechanism play a central role. In [50], centralization is defined as the organization of a system having an agent that is in charge of deciding the actions of the other robots; i.e., the leader is involved in the decisional process for the whole team, while the other members act according to the decisions of the leader. Of course, a centralized system is likely to have a hierarchical structure, in which the robots operating under the supervision of a leader, can be leaders of sub-teams forming the MRS. In strongly centralized systems, the leader remains the same during the mission. But, in certain cases, it is possible that the role of leader is dynamically assigned during the mis-

sion deployment, depending on environment changes or failures of the current leader. This is the case in weakly centralized systems.

As opposed to centralization, in distributed systems, robotic agents are completely autonomous in the decisional process with respect to others, i.e, a leader does not exist.

Weak and strong centralization allow assign in a simple way the tasks among the robots, since, at each time instant, only one of them is in charge of this assignment.

Centralized MRSs have the disadvantage that they strongly rely on communication. Thus, a communication failure results in a failure of the entire system. Moreover, a strongly centralized system can fail in accomplishing its task when the leader fails; a weakly centralized system tries to recover from a leader failure by selecting a new leader, as in [78]. An advantage of the strongly centralized MRSs is that a customized robotic agent can be the leader, e.g., by conferring to it the necessary computing and decisional capabilities.

On the contrary, a distributed MRS, as in [81], is characterized by a greater robustness to the above mentioned problems, since each team member decides autonomously; however, a greater complexity is required to achieve coordination between them.

In Figure 1.3 (left), an example of centralized architecture is shown, while on the right a 18 Swarm-Bots team organize themselves in order to drag a body in an hypothetic disaster scenario [15]. In practice, many systems are not strictly centralized/decentralized. In

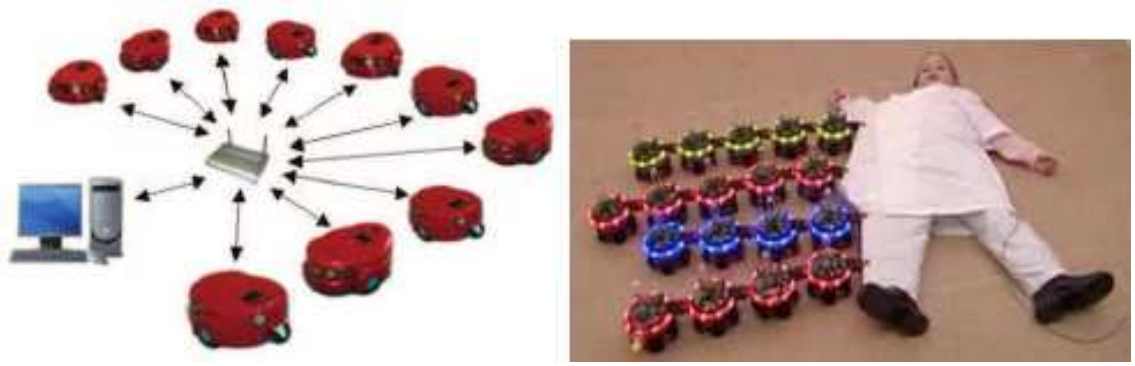

Figure 1.3: Examples of centralized and decentralized group of robots. Left: centralized architecture of the Decabot project. Right: a task performed by a team with decentralized architecture.

fact, many largely decentralized architectures use leader agents [107]; in particular, different hybrid centralized/decentralized architectures were demonstrated to take advantages of both the typologies [17], [43], [38], [104].

Communication Level

Strongly coordinated systems need to communicate in order to execute the coordination protocol, while weakly coordinated and not coordinated systems can be developed with or without communication, while unaware systems do not use communication at all. Two types of communications can be distinguished. In the *direct* communication case, dedicated on-board hardware is used to explicitly exchange information with other robots. On the other side, in *indirect* communication, information exchange is achieved by stigmergy [60], i.e. by sensed environment modifications. Obviously, direct communication is the easier and most powerful way to exchange information between the members of an MRS, but because of hardware failures or noise, it is also a critical point. Stigmergic communication is, of course, a more robust information exchange mechanism, but it implies a more complex MRS design, since each member of the team needs to interpret the surrounding environment changes and less information can be exchanged between team members.

Composition Level

The last level refers to the team composition. In particular, teams can be heterogeneous or homogeneous. In heterogeneous teams, robots have differences either in the hardware devices or in the software control procedures. In homogeneous teams, robots are exactly the same both in the hardware and in the control software. Multi-mobile robot systems based on this approach usually present different kind of vehicles, with different abilities. Examples are presented in [105], [101]. In [105], the authors demonstrate that Unmanned Aerial Vehicles (UAVs) and ground vehicles can collaborate to perform reconnaissance and surveillance tasks (Figure 1.4); similar results can be found in [29], [26], [67]. In [101], authors show the possibility of using three different robots (a mobile manipulator, a crane and a mobile robot equipped with a camera) to perform an assembly task, as the three robots have to collaborate to connect a beam at a given location (Figure 1.4). Other important references in this sense are [30] and [82]. In particular in the last reference, an architecture, called ALLIANCE, has been presented. The ALLIANCE architecture has been developed in order to study cooperation in an hetewith the goal of achieving a fault tolerant, reliable and adaptive mechanism for cooperative robot control. The homogeneity or the heterogeneity of the members can influence the way in wich robustness is achieved. In an homogeneous system, every agent can execute the same actions as the other team members with the same results, so that a failure of a member can be easily compensated

Figure 1.4: Example of heterogenous systems. Left: three robots cooperating for surveillance. Right: three robots cooperating in an assembly task.

by another robot in the system; hence, fault tolerance is guaranteed. On the contrary, adaptivity cannot be ensured, since there is no differentiation among the robots. Therefore, homogeneous systems are typically used in the so called swarm-type approach [15], where a large number of autonomous robots are used to form a distributed system. This method allows the system to achieve the desired goals when the repetitive execution of a set of actions is needed and time is not a critical resource.

On the other hand, heterogeneous robots allow the MRS to adapt more easily to the different situations which could emerge in a dynamic environment, since they offer a better chance to deal with new and unpredicted needs [82].

Obviously, working with heterogeneous systems implies a higher effort in realizing the software needed to control the overall system.

1.3 Main applications, test-beds and current research in Multi-Robot Systems

In this section, an overview of the main application of MRSs will be given. These applications are, of course, also test-beds to compare and test the effectiveness of more general architecture and strategy. Finally, a brief description of the main current research branch in the field of MRS, as individuated in [80] is reported.

The following applications have been individuated.

Foraging and Coverage

The foraging task requires the components of the MRS to pick up objects in the environment; this is of course representative of tasks like toxic waste cleaning, mine cleaning,

service robots [91], [52]. A major issue in this test-bed is to avoid interferences among the robots during the task execution. The coverage task is very similar to foraging, since it requires the robots to explore all the points of the free space in the environment. The main issue for coverage is, therefore, to find effective techniques for cooperatively scanning all the environment. Applications for coverage are: demining, snow removal, lawn mowing and car body painting.

Multi-Target Observation

In the multi-target observation task, a team of robots detects and tracks a set of moving objects. The robots have to maximize the time during which each of the moving target are observed by at least one of the robotic agents forming the team. The multi-target observation (also known as CMOMMT: Cooperative Multi-Robot Observation of Multiple Moving Targets) has been firstly introduced in [79]. Multi-Robot Observation has many connections with security, surveillance and recognition problems [79], where targets moving around in a bounded area must be observed.

Object Transportation

The task of box pushing requires the robotic agents to cooperatively push boxes in order to reach a desired configuration. The box pushing task has analogies with problems like stockage, truck loading and unloading, car moving and parking. While in the box pushing task the objects are assumed to be on a plane, more generally, object transportation focuses on lifting and carrying objects [102], thus increasing the complexity of the task. Typical applications are transportation or assembly of heavy and/or large objects in industrial environments, such as terrestrial buildings, or planetary habitats [102]. In most applications, object transportation is frequently used as test-bed for issues like motion coordination and formation control.

Exploration and Flocking

Exploration and flocking are used to indicate different tasks that differ in the way they are realized, but have the common feature to require MRS members to coordinate their motion in order to improve performance.

Tasks like flocking, formation maintenance or cooperative map building can be considered in the same class. In the exploration task, the robots need to distribute in the environment in order to collect as much information as possible about the surrounding area. In the

flocking task the goal for the robotic agents is to move together, such as in a flock. The formation task consists in the robots moving in the environment forming particular shapes. Cooperation among the robotic agents is also used to localize each other, and to fuse information acquired from the environment. Map building of unknown environments is a common issue related to exploration, and in particular, a very interesting topic is the cooperative simultaneous localization and mapping, in which the robots need to localize while moving and building the map of the environment [44]. The problem of exploration and flocking is related to several applications, such as motion coordination in industrial application and exploration of dangerous environments. Another example of exploration task is given by the RoboCup Robot-Rescue league [56], that is a setting for experimenting MRSs involved in searching victims in an unknown environment, representing a disaster scenario.

Soccer

Soccer has been considered in the last years as an interesting test-bed for research in multi-agent and multi-robot cooperation (see Figure 1.5). The uncertain dynamics and hostile environment in which the robots operate makes coordination of the multi-robot system a challenging problem.

In particular, in the Middle-Size league and the Four Legged league, all the robot

Figure 1.5: An example of robots playing soccer.

sensors must be on board; therefore robots are more autonomous and have to deal with high uncertainty in reconstructing global information about the environment. On the other hand, in the Small-Size league, the robotic agents can take advantage of a top view of the environment provided by a camera on the top of the field; therefore, the coordination

approaches in this league are mostly centralized. The use of coordination in the soccer domains has demonstrated significant improvements in the performance of the teams.

1.3.1 Current research in Multi-Robot Systems

Finally, after having given taxonomy and having described the main current test-beds for MRSs, it is worth briefly describing the current state of the art in autonomous multiple mobile robotic systems [80]. They can be summarized as follows.

Architectures, task planning and control

This research area addresses the issues of action selection, the communication structure, heterogeneity versus homogeneity in robots, achieving coherence in local actions, the resolution of conflicts, and other related issues. Each architecture that has been developed for multi-robot teams tends to focus on providing a specific type of capability to the distributed robot team.
As highlighted in [80], relatively little work has been aimed at unifying these architectures, in order to be applicable to a much wider variety of domains than is currently possible with existing architectures.

Communication

A lot of work is being done by researchers to study the effect of communication on the performance of multi-robot teams in a variety of tasks. Clearly, in many cases, communication of even a small amount of information can lead to great benefit [14] [64]. Other work is aimed at achieving fault-tolerance in multirobot communication, such as maintaining distributed communication networks and ensuring reliability.

Localization, mapping and exploration

A lot of work on localization, mapping and exploration has been done for single autonomous robots, but only in the last years they have been applied to multi-robot teams. In addition, very often, the research has considered existing algorithms, developed for single-robot mapping, localization, or exploration, and has extended them to multiple robots, as opposed to developing new distributed algorithms. One exception is some of the work in multi-robot localization, which explicitly takes advantage of multiple robots to improve positioning accuracy [77].

As for single-robot approaches, research for multi-robot localization approaches use familiar categories, based on the use of landmarks, scan-matching, graphs, and using either range sensors (such as sonar or laser) or vision sensors. While the single-robot version of this problem is fairly well understood, much work remains to be done for the multi-robot case. For example, one question is about the effectiveness of multi-robot teams over single-robot approaches, and in which cases the addition of robots brings diminishing returns. This issue has begun to be studied, but much remains to be determined for the variety of approaches available for localization, mapping and exploration.

Reconfigurable robotics

Relatively recent works are aimed at creating physical robot systems that are able to reconfigure (e.g., to connect and reconnect in various ways to generate a desired shape), depending on the required functions (see Figure 1.6). These systems have the theoretical capability of showing great robustness, versatility and, even, self-repair capabilities. Most of the work in this area involves identical modules with interconnection mechanisms that allow either manual or automatic reconfiguration [115], [28]. Research in this area is still very new, and the practical applications of these systems are yet to be demonstrated. Clearly, this is a rich area for continuing advances in multi-robot systems.

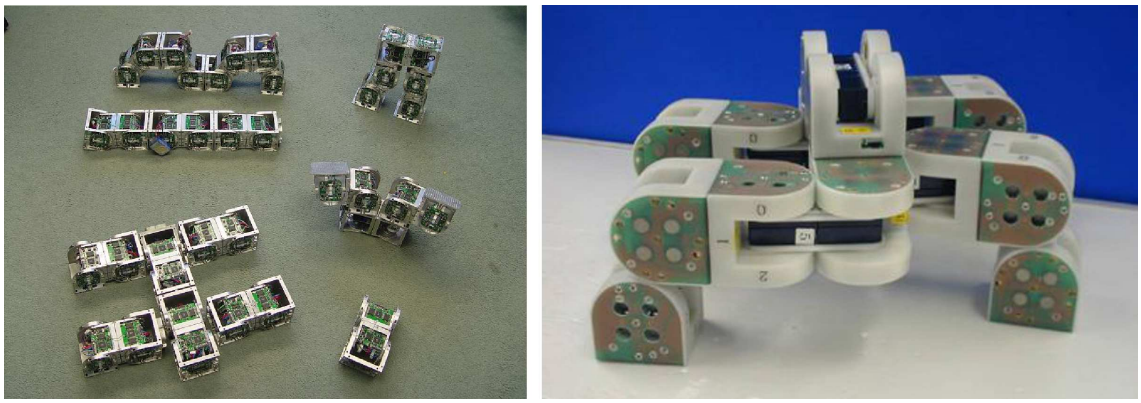

Figure 1.6: An example of reconfigurable robots composed in different shapes.

Learning

Building a robot that learns to perform a task is one of the major challenges in artificial intelligence. Self-improving robots would relieve humans from much of the effort of

programming and would allow operation in highly dynamic or partially known environments. The increasing interest in robot learning has partially been caused by advances in the areas of reinforcement learning, behavior-based architectures and genetic algorithms, Because of their complexity, the types of application that are typically used as test-bed for this area include predator/prey [18], box-pushing [66], foraging, multi-robot soccer, cooperative target and observation [79].

Chapter 2

The Null-Space-Based Behavioral Control

In this chapter, a generic technique for the control of industrial manipulators, single autonomous vehicles as well as for multi-robot systems, namely the Null-Space-Based Behavioral Control (NSB), is presented. After a general overview of the more common behavioral approaches existing in literature, the NSB approach will be addressed. More in detail, the NSB is a behavior based technique, inherited from a class of kinematic inverse algorithms for serial manipulators and properly extended to autonomous vehicles. This technique tries to combine the cooperative nature of the motor schema control and the predictability of the layered control system, and at the same time, overcome their disadvantages. A general and rigorous mathematical description of the NSB will be carried out in the following; moreover, a geometrical and intuitive explanation will be given. Finally, it will be shown how the NSB approach can be applied to the case of the multi-robot systems, by formulating two tasks in the NSB framework and discussing them in detail.

2.1 Review of the main behavioral approaches

Before introducing the main aspects of behavioral robotics and of the NSB approaches, it is useful to briefly illustrate what are the most difficult aspects we need to face when dealing with robot systems.

Autonomous robotic systems present several difficult challenges. An autonomous robotic system must be extremely self- reliant to operate in complex, partially known and challenging environments using its limited physical and computational resources. In spite of this difficulties, the control system must ensure in real time that the robot will achieve

its tasks. Moreover, the control is required to be reactively fast but also thorough, while preserving some properties like stability and robustness. More in detail, control and computational architectures should be designed to meet at least the following requirements:

- Reactivity to the environment. The robot should be reactive to sudden changes in the environment and capable to take into account external events within time bounds compatible with the correct, efficient and safe execution of its tasks.
- Intelligent behavior. In real environments, different compromises need to be taken into account in order to exhibit intelligent behavior. For example,consider the simple case of a robot trying ro reach a goal. An obstacle along the path may require the robot to drive away form the obstacle, moreover, the escape strategy must always take into account the main objective (i.e. to reach the goal). External stimuli must be always guided by the objectives of its main tasks.
- Resolving multiple goals. In the case of mobile robots, situations requiring conflicting concurrent actions are inevitable. The control system should provide means to fulfill those multiple goals.
- Robustness. The robot must handle noisy inputs, unexpected events and malfunctions.
- Reliability. Robots should operate without failures or performance degradation over a certain period.

Then one of the challenge when dealing with robotic systems consists in performing a given mission in highly unstructured and complex environments. This complexity prevents, on one side, from the possibility to elaborate once for all an off-line and static plan that allows to accomplish the mission; on the other hand, imposes additional sub-missions as obstacle-avoidance or other additional constraints that can be in conflict with the main mission. As stated in [22], both in the case of single and multi-robot systems, the best approach seems to be the decomposition of the problem in several sub-problems, eventually solvable in parallel.
Then, the problem becomes how to compose the commands required by each task in one single motion command to the robot. The subproblems are commonly termed *behaviors*, *functional modules*, *motor schemes*, or *tasks*; these terms will be used as synonyms; the

term behaviors will be preferred in the following, and any method aimed at properly composing the elementary behaviors will be named as *behavioral approach*. According to [22], based on selective sensory information, each behavior produces immediate reactions to control the robot with respect to a particular objective, such as obstacle avoidance or wall following. Behaviors with different objectives may produce conflicting actions that are seemingly irreconcilable. Thus a major issue in the design of behavior-based control systems is the formulation of an effective mechanism for coordination of the behaviors' activities [86]. This problem is known as the action selection problem (also refereed to as the behavior coordination problem). Numerous action selection mechanisms have been proposed over the last decade; a classification of the main approaches, whose a scheme is reported in Figure 2.1 can be found in [95] .

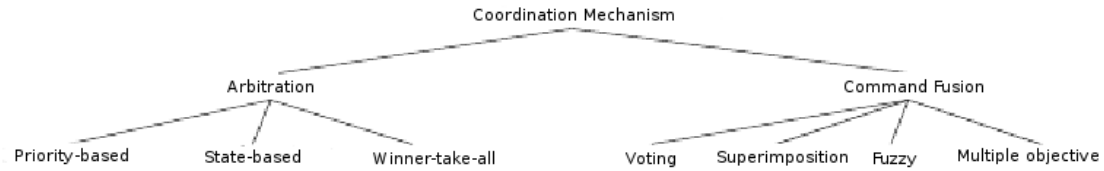

Figure 2.1: Classes of action selection mechanism.

In particular, two main categories have been identified. The first one groups all the *arbitration* mechanisms, i.e., competitives approaches aimed at selecting one single behaviors among a set. The second category is the *command fusion* mechanisms, whose aim is to let the different behaviors *cooperate*, in order to accomplish the overall mission by eventually exploiting the system *redundancy*. It is worth reminding, that all the above concepts are suitable both for single-robot and multi-robot systems.

2.1.1 Arbitration Mechanisms

Arbitration mechanisms select one behavior at once and give it ultimate control of the robot until the next selection cycle. Arbitration mechanisms for action selection can be divided into: priority based, state- based and winner-take-all. These are different techniques for selecting an appropriate action at any moment and resolving conflicting situations.

Priority-based approach

In priority based mechanisms an action is selected by a central module based on *a priori* assigned priorities. The Subsumption Architecture (SA) designed by Rodney Brooks [22]

represents a way to incrementally build the control system of a robotic system. In Figures 2.2 and 2.3, the graphical representation of the single behavior and the general scheme of this architecture are depicted. SA consists of a series of behaviors each performing some functionalities (e.g. obstacle avoidance, wander around, explore), and action selection consists of higher-level behaviors subsuming the output of lower-level.

Figure 2.2: Single task representation.

It is clearly a *competitive* approach, where each behavior is related to a layer that is an asynchronous module working independently from the others, and elaborating an output that is a motion command for the robot. Each behavior is able to process data from the levels below and it is allowed to inject data into the their interface to inhibit or suppress their output.

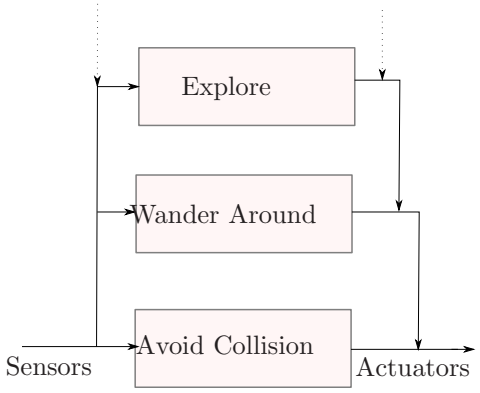

Figure 2.3: The Subsumption Architecture.

In this way, the higher priority behavior is always achieved, while lower priority one is achieved only when all upper behaviors in the hierarchy have zero output commands. Clearly, this mechanism allows to easily add a new behavior simply assigning its priority level.

However, it can be shown that a combinatorial explosion in the number of possible behaviors occurs when decomposing a mission according to the SA [112]. Another important limitation of SA is the lack of explicit goals coding and its goal-handling capabilities. The overall behavior of a SA based robot is an emergent property of the interaction of its behaviors and depends on the static and dynamic properties of the environment. In certain

cases it would be convenient to have more direct control of the robot; however, typically there is no way to guarantee such a property in the SA case.

State-based arbitration approach

According to *state-based arbitration* [58], a set of behaviors that is adequately competent to handle the situation corresponding to the given state is selected.
Using this formalism, systems are modeled in terms of Finite State Automata (FSA), where states correspond to execution of actions/behaviors and events, corresponding to observations and actions, cause transitions between the states.
An example of such a system is depicted in Figure 2.4, where the mission consists in *finding* and *traversing* a door while *avoiding collisions*. In [58] the agent and its interaction with the environment are modeled by a FSA and is denoted the plant. The design objective is to specify an additional FSA, called the supervisor, that interacts with the plant and modifies the its behavior, in order to assure the desired behavior. The supervisor achieves its objective by enabling and disabling the controllable events to modify the state transition function of the plant, so that only desired subsets of event sequences are generated by the plant.

For complex systems and tasks with an high number of behaviors and events, this formalism can be very tedious and complex. For this reason, in [59] linguistics tools that automatically can transfer textual descriptions of plans to models and FSAs implementing the desired behavior have been developed.

Winner-take-all approach

The last arbitration mechanism we will consider is the *winner-take-all* approach, whose best expression is the work of Maes [65]. In this approach, based on a society-of-mind type of idea [75], the system consists of a set of behaviors or competence modules which are connected in a network. In the network, each behavior is represented by a tuple $(c_i,\ a_i,\ d_i,\ \alpha_i)$ describing: 1) The preconditions c_i wich have to be fulfilled for the activation of a competence module. 2) The effects after successful execution in form of two lists a_i (the add-list) and d_i (teh delete list). More in detail, a precondition p belongs to a_i (d_i) if it becomes true (false) after the execution of the module i. 3) The activation level, α_i, which is a measure of applicability of the behavior. When the activation level of an executable behavior exceeds a specified threshold, it is selected to perform its action.

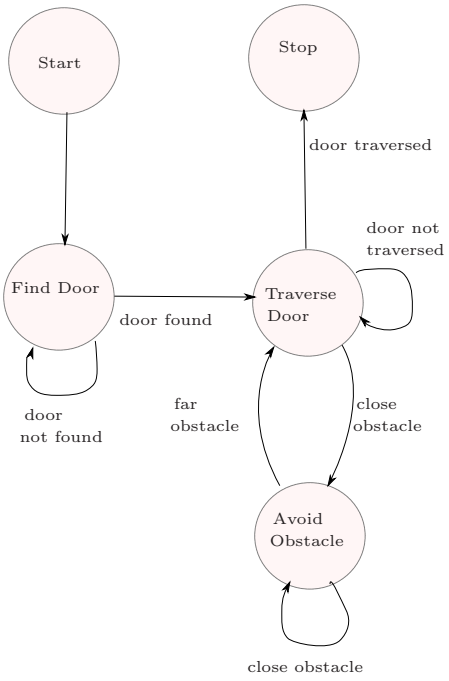

Figure 2.4: A Finite State Automata example. The mission consists in *finding* and *traversing* a door while *avoiding collisions*.

The activation level of the behaviors is influenced by an external injection or removal of activation energy, due to some external stimuli, as well as by internal exchange of activation energy between the modules within the network. External injection of energy is influenced, for example, by:

- current state environment: it is seen as a precondition for the behavior execution;
- current goal: a behavior receives as more energy as it is able to achieve the goal;
- protected goals: the energy is decreased if the behavior can undo a goal that has been already achieved.

Internal sources of energy can be:

- activation of successors: an executable behavior spreads activation to behaviors with preconditions that will become true after its activation;
- activation of predecessors: a behavior that is not executable spreads activation to behaviors that may make it executable, i.e., behaviors which will make some preconditions true when executed;

- inhibition of conflictors: every behavior decreases the activation level of behaviors, called conflictors, that can make its preconditions false.

The activation energy exchange takes place continuously and at each cycle an executable behavior with the highest activation level is selected. Unlike traditional planners, this mechanism does not have an explicit representation of plans, and it does not implement a specific search algorithm to construct plans. The system *emergently* chooses and performs the next step of the sequence, which is influenced not only by the system goals but also the state of the environment at each step. Due to its distributed nature, the system is fault-tolerant, since the failure of a single module/behavior does not necessarily lead to the failure of the entire system. However, this approach does not allow to analytically proof tasks fulfilment; moreover, several parameters and thresholds are involved in this approach, thus making difficult their tuning and the application in complex systems.

2.1.2 Command Fusion Mechanisms

While arbitration mechanism belong to the class of competitive approaches, command fusion mechanisms try to establish a sort of *cooperation* between behaviors, combining recommendations from multiple behaviors to form a control action that represents their consensus.
Different command fusion approaches differ in the way they achieve the consensus between behaviors and, according to Figure 2.1, they have been classified in Voting, Superimposition, Fuzzy and Multiple Objective approaches.

Voting approach

The fist approach belonging to this class we will consider is the *voting* mechanism, whose best example is the Distributed Architecture for Mobile Navigation (DAMN) described in [92]. DAMN consists of a set of behaviors voting for or against the set of actions constituting the possible set of actions of the agent. Behavior votes represent the behavior's preference over the set of possible actions. Then, an external arbiter selects the *best* action, which is chosen as the action with the maximum weighted sum of the received votes, where each behavior is assigned a weight by the mode manager. These weights reflect the relative importance or priority of the behavior in a given context. The DAMN architecture is represented in Figure 2.5. DAMN has been successfully applied to indoor as well as outdoor real-world applications, such as the outdoor CMU Navlab vehicles, and indoor

real and simulated robots at the Hughes Research Labs. In [92] the DAMN architecture is used to control heading and speed of a mobile robot, while in [93] is used to choose the field of view for a motorized camera head. Further, [113] investigates and compares a number of Action Selection Mechanisms (ASMs), including DAMN and Maes' Activation Networks, in a simulated environment. The experimental results showed that DAMN is superior to the other ASMs it is compared to.

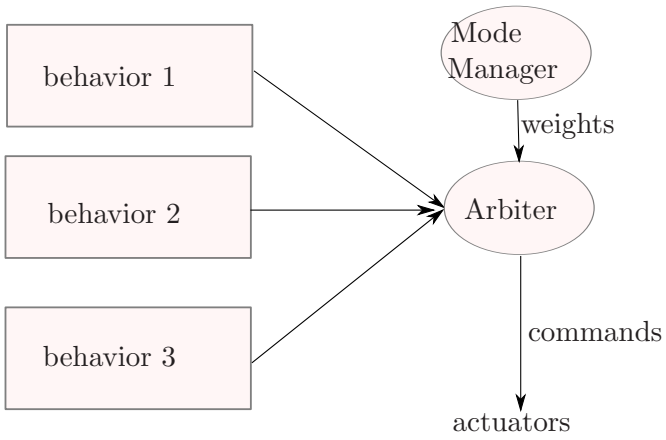

Figure 2.5: The DAMN architecture scheme. A mode manager is in charge of generating a set of weights depending on the current state. According to the generated weights and behaviors' votes an arbiter selects the best action to activate.

Fuzzy behavior-based control

In fuzzy behavior-based control, each behavior is synthesized by a rule controlled by an inference engine to produce a multivalued output. Behaviors that compete for the control of the robot must be coordinated to resolve potential conflicts. Then, fuzzy behavior coordination is performed by combining the fuzzy outputs of the behaviors using an appropriate operator, such as the fuzzy set union (e.g., the max operator). Then defuzzification (e.g., center of gravity, COG) is used to select a final crisp action ultimately used for control [95]. This scheme should lead to the selection of an action that represents the consensus among the behaviors and thus comprises the action that best satisfies the decision objectives that they encode.

Figure 2.6 shows a simple example of fuzzy decision mechanism; nevertheless the latter can be integrated in very complex layered architecture, as, e.g., in [96].

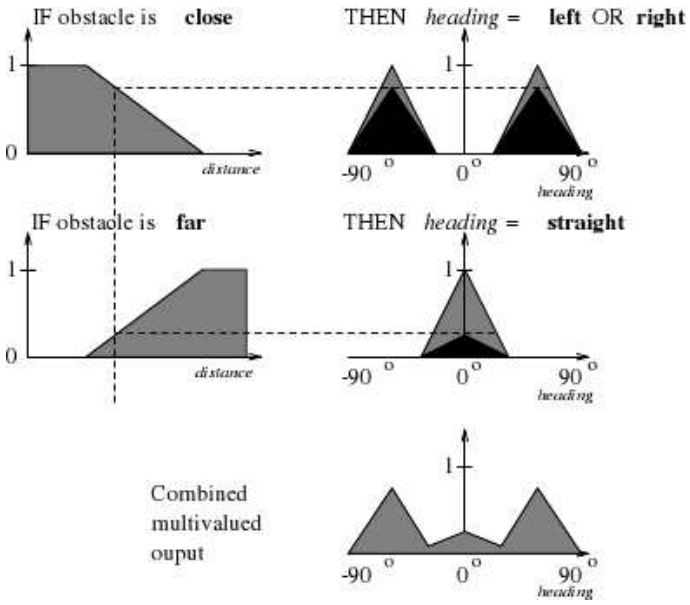

Figure 2.6: A fuzzy behavior-based example.

Multiple objective behavior approach

The philosophy behind this approach is that in real-world robotics it is not feasible to aim for optimal control even, if an optimal strategy exists, because unstructured environments are unpredictable and robotic sensors and actuators are imperfect. It is, thus, more appropriate to search for control strategies that in practice are efficient and sufficiently satisfying, i.e., *good enough* rather than optimal; then, the control strategies should explicitly define a notion of optimality that is both suitable and practically feasible. In multiple objective behavior coordination, a behavior i is formalized as an objective function that defines a mapping from an action space, $\boldsymbol{X}$, to the interval $[0; 1]$: $o_i : \boldsymbol{X} \rightarrow [0;\ 1]$. The action space, $\boldsymbol{X} = [\boldsymbol{x}_1; \boldsymbol{x}_2; \ldots \boldsymbol{x}_n]$, is the set of all possible actions. Once the link between behaviors and objective functions is established, we can formulate the coordination of multiple behaviors as a multiple objective decision problem, that optimizes a number of objective functions simultaneously, where the objective functions correspond to behavior multivalued outputs. From the set of alternative actions $\boldsymbol{X}$, an action $\overline{\boldsymbol{x}}$ should be chosen as

$$\overline{\boldsymbol{x}} = \max_{\boldsymbol{x} \in \boldsymbol{X}} [o_1(\boldsymbol{x}), o_2(\boldsymbol{x}), \ldots o_n(\boldsymbol{x})], \tag{2.1}$$

that corresponds to the action that best satisfies the corresponding behaviors. The *optimal* solution is based on the concept of Pareto-Optimal Solution, and we remand to [31],[87]

for the mathematical detail of solution to problem (2.1).

Superposition-based command fusion approach

The last command fusion approach we will consider is the superposition based command fusion approach. A first example of this mechanism can be found in [53]. In this work, an approach to motion planning is proposed, where the robot, represented as a point in configuration space, moves, for example, under the influence of an artificial potential field produced by an attractive force at the goal configuration and repulsive forces at the obstacles. Action selection in this terminology corresponds, at each configuration, in moving in the direction indicated by the negated gradient of the total potential.
Potential field approaches have been successfully used for planning collision-free path for robotic arms [53] as well as for mobile robot platforms [36]. However, the usual formulations of potential field methods do not avoid the occurrences of minima other than the goal. In [36] the application of harmonic functions to potential field path-planning is described. One of the properties of this approach is that it guarantees the exclusion of local minima within the solution region.
The best and perhaps the most famous expression of such an approach is the work described in [10]. Ronald Arkin in [10] proposes an approach based on the concepts of *motor schema* and potential fields. The concept of schema originates in psychology and neurology, and it is defined as "*the basic unit of motor behavior from which complex actions can be constructed. It consists of both the knowledge of how to act as well as the computational process by which it is enacted*". There are two types of schemas: motor schemas, which are concerned with control of actuators, and perceptual schemas, which are concerned with sensing of environment features. For example, given an obstacle position, an obstacle avoidance motor schema can control the robot away from the obstacle, while, in order to detect the obstacles, an obstacle detection perceptual schema can be instantiated to keep track of obstacles. The output of a motor schema is always a velocity command; then, since more than one motor schema can be instantiated at each time instant to handle the different goals, the different outputs can be added to generate an overall combined motor action $\boldsymbol{v}_d$. Since motor schemas can have different importance, each velocity command is multiplied by a gain whose value is usually determined empirically for a given mission and environment

$$\boldsymbol{v}_d = \sum_i \alpha_i \boldsymbol{v}_i,$$

where $\boldsymbol{v}_i$ is the output velocity command generated by the i-th motor schema and α_i is the corresponding gain. A representation of such an architecture is depicted in Figure 2.7. Clearly, in general, no task is completely achieved but, on the basis of gain vectors, a compromise solution is found.

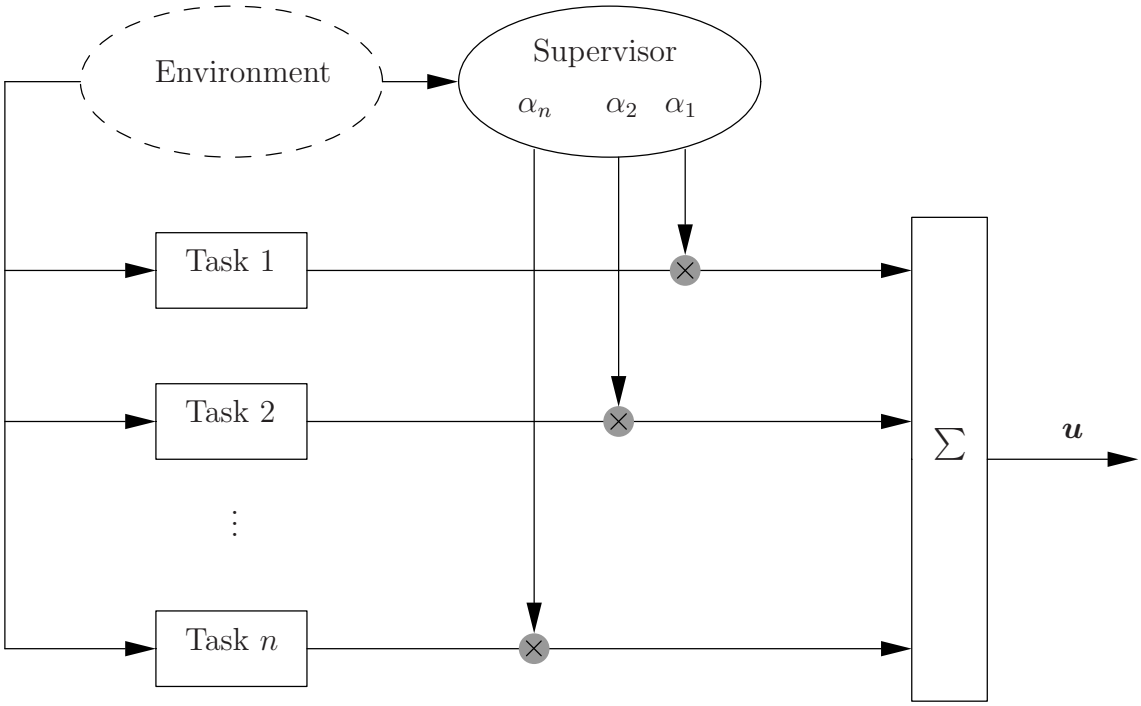

Figure 2.7: The Motor-Schema architecture.

2.2 The Null-Spaced-based Behavioral approach

As discussed above, command fusion (cooperative) approaches allows to *combine* the output of several tasks, trying to achieve different goals but leading to troubles in case of conflicting tasks, while arbitration (competitive) approaches allow to perform one task at a time, leading to an underused system but to a predictable output. The Null-Space-based-Behavioral (NSB) approach can be defined as a competitive-cooperative approach, trying to overcome the disadvantages of the above approaches. A good starting point is the work done by B. Bishop in [21], where the concept of redundancy for standard robot manipulators is extended to the control of platoons of autonomous vehicles.

A redundant manipulator is a system having a number of joints degrees of freedom larger than task variables. The control of such systems has been widely studied in the past [100],

extra-degrees of freedom can be accommodated to satisfy secondary objectives arranging the tasks in priorities [76],[63]. The same concept of redundancy can be applied to the problem of multi-robot systems, where the redundancy can be used to accomplish secondary tasks. In [21] four different tasks have been considered with two different priority levels, and the overall robots' velocities is the sum of the output velocities of each behavior. Then, the proposed solution does not take into account the problem of algorithmic singularities [35], roughly defined as a condition where two or more behaviors are in conflict; in addition the maximum number of achievable tasks is not taken into account, thus making the solution very similar to the approach proposed in [10]. In [103] the controller pursues a primary task that combines multiple objectives (according to the augmented task strategy [13]) and uses, in proximity of obstacles, a null-space projection to locally achieve their avoidance. Then, also in this case, the combination of different tasks at the same level of priority requires structurally compatible tasks, making this solution not robust toward algorithmic singularities [35].

The NSB approach is based on the task-priority kinematic control approach for ground fixed manipulators [76]. In particular, the task-priority method allows to properly handle the coflicts among different tasks. The problems related to singularities are deeply discussed in [35], 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 coflict with an higher level task, while it is released in case of coflict. The more general case of compatibility of three or more tasks handled in the NSB framework has been discussed by Antonelli in [7], resorting to a Lypunov-based analysis. Finally, the NSB approach has been experimentally compared to the layered-control system and motor-schema control in the case of a single wheeled vehicle [8] and succesfully applied to control of a platoon of grounded vehicles [9],[11].

2.2.1 The NSB mathematics

In this section, the mathematical formulation of the NSB approach. Let us consider a platoon composed by n vehicles, the vector expressing the position of the i-*th* vehicle in the inertial reference frame is

$$\boldsymbol{p}_i = \begin{bmatrix} x_i & y_i & z_i \end{bmatrix}^T \in \mathcal{R}^{3\times 1}, \tag{2.2}$$

while the corresponding velocity is

$$\dot{\boldsymbol{p}}_i = \boldsymbol{v}_i = \begin{bmatrix} \dot{x}_i & \dot{y}_i & \dot{z}_i \end{bmatrix}^T \in \mathcal{R}^{3\times 1}. \tag{2.3}$$

Correspondingly, the configuration of the overall platoon in compact form is

$$\boldsymbol{p} = \begin{bmatrix} \boldsymbol{p}_1^T & \boldsymbol{p}_2^T & \dots & \boldsymbol{p}_n^T \end{bmatrix}^T \in \mathcal{R}^{3n\times 1}, \tag{2.4}$$

and

$$\boldsymbol{v} = \begin{bmatrix} \boldsymbol{v}_1^T & \boldsymbol{v}_2^T \dots \boldsymbol{v}_n^T \end{bmatrix}^T \in \mathcal{R}^{3n\times 1}. \tag{2.5}$$

The task assigned to the platoon can be defined by the function

$$\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{p}) \in \mathcal{R}^{m\times 1}, \tag{2.6}$$

depending on the platoon state $\boldsymbol{p}$. Assuming $\boldsymbol{\sigma}$ differentiable, it is useful to consider the relationship between the first derivatives of $\boldsymbol{\sigma}$ and $\boldsymbol{p}$:

$$\dot{\boldsymbol{\sigma}} = \frac{\partial \boldsymbol{\sigma}(\boldsymbol{p})}{\partial \boldsymbol{p}} \boldsymbol{v} = \boldsymbol{J}(\boldsymbol{p})\boldsymbol{v}, \tag{2.7}$$

where the matrix $\boldsymbol{J} \in \mathcal{R}^{m\times 3n}$ is the configuration dependent Jacobian matrix of the transformation $\boldsymbol{\sigma} = \boldsymbol{\sigma}(\boldsymbol{p})$. The relationship (2.7) can be adopted for any type of robotic system, no matter the number and type of robots; the system can be, for example, composed by one or more industrial manipulators or wheeled robots; the only difference will involve the state dimension and, consequently, the Jacobian dimension. Relationships (2.6),(2.7) are the so-called direct kinematic equations of the system that allows to calculate the value of the task function given the system configuration (in this case the robots' positions $\boldsymbol{p}$). The direct kinematic problem is generally easy to solve, while the inverse kinematic problem defined as the calculation of the motion references $\boldsymbol{p}_d$ corresponding to a desired time history $\boldsymbol{\sigma}_d$ of the task function, is usually more complex. It is a very important problem since, generally, by means of the task function $\boldsymbol{\sigma}_d$, the desired high level aggregate behavior of the system is specified, while the corresponding desired positions $\boldsymbol{p}_d$ are the ultimate commands sent to the robots. The inverse kinematics problem has been widely studied in the past, mainly with reference to serial link manipulators [100], where the main problem is the calculation of the joint trajectories, given the desired time history of the end-effector pose (position/orientation).

The inverse kinematics problem in the case of a $(n \times n)$ square Jacobian matrix can

be apparently easily resolved by assigning $\boldsymbol{\sigma}_d$ and computing the corresponding vehicles' velocities

$$\boldsymbol{v}_d = \boldsymbol{J}(\boldsymbol{p}_d)^{-1}\dot{\boldsymbol{\sigma}}_d. \tag{2.8}$$

However, more interesting is the case of a low rectangular matrix $\boldsymbol{J}$ (i.e. with more columns than rows), a common case for multi-robot-systems. In this case, infinite solutions to the inverse kinematic problem exist; one possible choice is

$$\boldsymbol{v}_d = \boldsymbol{J}^{\#}\dot{\boldsymbol{\sigma}}_d, \tag{2.9}$$

where the dependency of the Jacobian on the platoon state has been omitted and $\boldsymbol{J}^{\#} \in \mathbb{R}^{3n\times m}$ is the Moore-Penrose inverse of matrix $\boldsymbol{J}$, having the following properties:

- $\boldsymbol{J}\boldsymbol{J}^{\#}\boldsymbol{J} = \boldsymbol{J}$, i.e., $\boldsymbol{J}^{\#}$ is a generalized inverse of $\boldsymbol{J}$.
- $\boldsymbol{J}^{\#}\boldsymbol{J}\boldsymbol{J}^{\#} = \boldsymbol{J}^{\#}$, i.e., $\boldsymbol{J}^{\#}$ is an outer pseudoinverse of $\boldsymbol{J}$.
- $\boldsymbol{J}^{\#}\boldsymbol{J}$ is symmetric.
- $\boldsymbol{J}\boldsymbol{J}^{\#}$ is symmetric.

One possible expression of $\boldsymbol{J}^{\#}$ is

$$\boldsymbol{J}^{\#} = \boldsymbol{J}^{\dagger} = \boldsymbol{J}^{T}\left(\boldsymbol{J}\boldsymbol{J}^{T}\right)^{-1}. \tag{2.10}$$

It can be shown that the choice (2.10) allows to *locally* minimize the norm of vehicles' velocities [99]. The vector $\boldsymbol{v}_d$ needs to be continuous-time integrated to compute the corresponding desired robot positions $\boldsymbol{p}_d$, used to update the Jacobian $\boldsymbol{J}$.
The following algorithm is usually adopted known as CLIK (Closed Loop Inverse Kinematics) algorithm:

$$\begin{cases} \boldsymbol{v}_d(t) &= \boldsymbol{J}(\boldsymbol{p}(t))^{\dagger}\left[\boldsymbol{\sigma}_d + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}\right] \\ \boldsymbol{p}(t) &= \int_{\mathbf{t_0}}^{\mathbf{t}} \boldsymbol{v_d}(\mathbf{t}), \end{cases} \tag{2.11}$$

where $\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_d - \boldsymbol{\sigma}(p(t))$ and $\boldsymbol{\Lambda} \in \mathcal{R}^{m\times m}$ is a positive definite matrix. It can be easily shown that equation (2.11) leads to the following asymptotically stable system

$$\dot{\tilde{\boldsymbol{\sigma}}} + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}} = \mathbf{0}. \tag{2.12}$$

However, since in practical applications a discrete-time implementation is needed, the discrete-time integration must be adopted, leading to

$$\begin{cases} \boldsymbol{v}_d(t_k) &= \boldsymbol{J}^{\dagger}(p_d(t_{k-1}))\left[\dot{\boldsymbol{\sigma}}_d(t_k) + \boldsymbol{\Lambda}\tilde{\boldsymbol{\sigma}}\right] \\ \boldsymbol{p}_d(t_k) &= \boldsymbol{p}_d(t_{k-1}) + \boldsymbol{v}_d(t_k)\Delta t, \end{cases} \tag{2.13}$$

where $\tilde{\boldsymbol{\sigma}} = \boldsymbol{\sigma}_d(t_{k-1}) - \boldsymbol{\sigma}(p_d(t_{k-1}))$ and Δt is the sampling time.
As stated above, when dealing with Multi-Robot-Systems several tasks need to be handled in the same time. Without loss of generality, let us consider two tasks $\boldsymbol{\sigma}_{d,1} \in \mathcal{R}^{m_1 \times 1}$, $\boldsymbol{\sigma}_{d,2} \in \mathcal{R}^{m_2 \times 1}$ and the corresponding Jacobians $\boldsymbol{J}_1 \in \mathcal{R}^{m_1 \times 3n}$, $\boldsymbol{J}_2 \in \mathcal{R}^{m_2 \times 3n}$ with $m_1 + m_2 \leq 3n$, we can consider the following continuos time algorithm

$$\begin{cases} \boldsymbol{v}_d &= \boldsymbol{J}^{\dagger} \left[\dot{\boldsymbol{\sigma}}_d + \boldsymbol{\Lambda} \tilde{\boldsymbol{\sigma}} \right] \\ \boldsymbol{J} &= \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \end{bmatrix}, \boldsymbol{\sigma}_d = \begin{bmatrix} \boldsymbol{\sigma}_{d,1} \\ \boldsymbol{\sigma}_{d,2} \end{bmatrix}, \tilde{\boldsymbol{\sigma}} = \begin{bmatrix} \tilde{\boldsymbol{\sigma}}_1 \\ \tilde{\boldsymbol{\sigma}}_2 \end{bmatrix}, \end{cases} \tag{2.14}$$

that consists in stacking all the tasks in an overall Jacobian, $\boldsymbol{J}$, and applying the solution (2.10). It can be shown that solution (2.14) is not robust, since conflicting tasks result in a singular Jacobian $\boldsymbol{J}$. A more robust solution has been presented in [35] where a priority mechanism is used. Let us consider the task velocity

$$\boldsymbol{v}_{d,i} = \boldsymbol{J}_i^{\dagger} \left[\dot{\boldsymbol{\sigma}}_{d,i} + \boldsymbol{\Lambda}_i \tilde{\boldsymbol{\sigma}} \right], \tag{2.15}$$

where the subscript i denotes the $i-th$ task and task 1 is the highest priority task.

Considering N tasks, the NSB solution to the task combination can be formulated in an iterative way, defining the velocity vector as follows

$$\boldsymbol{v}^{(i)} = \boldsymbol{v}_{d,i} + \boldsymbol{N}_i \boldsymbol{v}^{(i+1)}, \quad i = 1, 2, \cdots N, \tag{2.16}$$

with $\boldsymbol{v}^{(n+1)} = \boldsymbol{0}$, $\boldsymbol{v}^{(1)} = \boldsymbol{v}_d$, and the matrix $\boldsymbol{N}_i = \mathcal{N}(\boldsymbol{J}_i) = \left(\boldsymbol{I} - \boldsymbol{J}_i^{\dagger} \boldsymbol{J}_i^T \right)$ is the null-space projector matrix of Jacobian $\boldsymbol{J}_i$ (i.e. its columns form a basis of the null space of matrix $\boldsymbol{J}_i$). For example, in the case of only three tasks (2.16) becomes

$$\boldsymbol{v}_d = \boldsymbol{v}_{d,1} + \boldsymbol{N}_1 \left(\boldsymbol{v}_{d,2} + \boldsymbol{N}_2 \boldsymbol{v}_{d,3} \right). \tag{2.17}$$

In sum, velocities computed by (2.13), 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 higher priority task velocity components. Figure 2.8 shows a representation of the NSB approach in the case of three tasks, whose priorities are assigned by a supervisor, where finally the corresponding task velocities are composed in the NSB sense 2.16. A simple example of such an architecture is given in [34], where a robot manipulator moves in an environment with obstacles performing one or more tasks handled in the NSB framework. In particular, when the robot is close

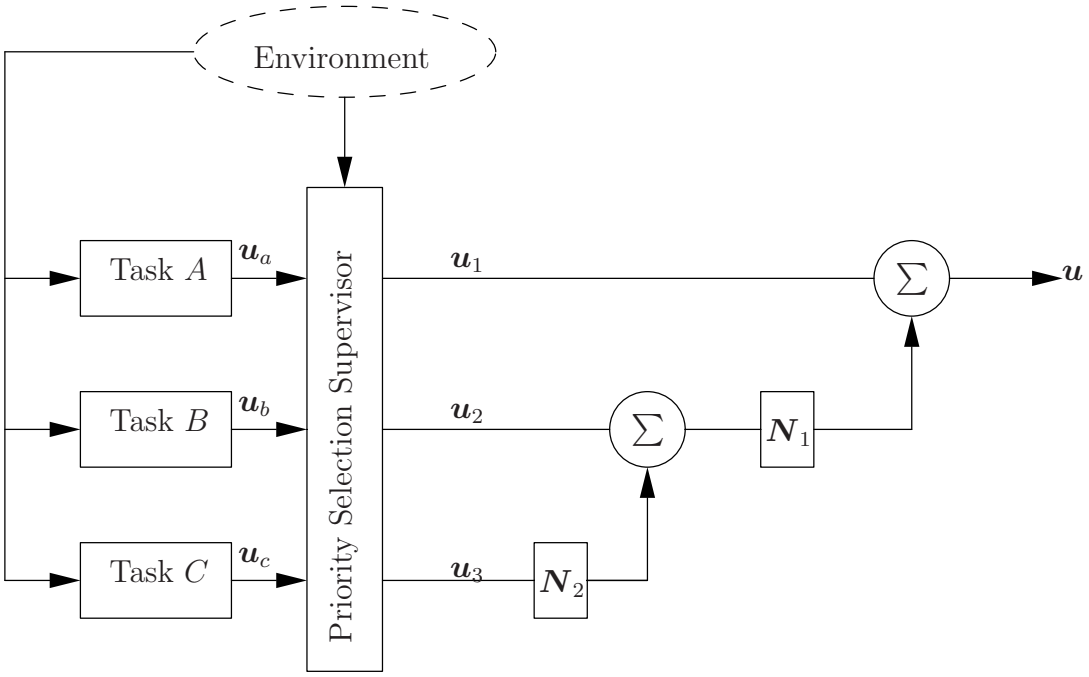

Figure 2.8: The Null-Space-Based schema in the case of three tasks.

to an obstacle, to preserve the manipulator integrity the supervisor assigns the maximum priority to the obstacle avoidance task and lower priorities to the other tasks.

The NSB approach can be considered a competitive-cooperative approach; in fact, it allows to combine several tasks at the same time, while avoiding conflicts between tasks, as lower priority tasks are fulfilled only in their components not affecting higher priority tasks. Moreover, it is worth noting that the motor-schema control and the layered control systems can be cast as particular cases of the NSB approach. By considering N tasks stacked in the vector $\boldsymbol{\sigma}$

$$\boldsymbol{\sigma} = \begin{bmatrix} \boldsymbol{\sigma}_1^T & \boldsymbol{\sigma}_2^T & \cdots & \boldsymbol{\sigma}_N^T \end{bmatrix}^T,$$

and the corresponding Jacobian

$$\boldsymbol{J} = \begin{bmatrix} \boldsymbol{J}_1 & \boldsymbol{J}_2 & \cdots & \boldsymbol{J}_n \end{bmatrix},$$

by using a weighted pseudoinverse $\boldsymbol{J}_{\boldsymbol{W}}^{\dagger} = \boldsymbol{W}^{-1}\boldsymbol{J}^T\left(\boldsymbol{J}\boldsymbol{W}^{-1}\boldsymbol{J}^T\right)^{-1}$ and

$$\boldsymbol{W} = diag\left\{ w_1\boldsymbol{I}, \quad w_2\boldsymbol{I}, \quad \cdots, \quad w_n\boldsymbol{I} \right\},$$

with $\boldsymbol{I}$ proper dimension identity matrices, it is possible to obtain the motor-schema control by choosing $w_i \in [0, \infty)$, and the layered-schema system by choosing $w_i = 1$ and

$w_j = 0, \forall i \neq j$ to selectively activate the task i.

2.2.2 NSB geometric interpretation

The NSB approach to task combination can be better illustrated in a geometrical way in the simple case of a 2-DOF mobile robot moving in a plane. Let's consider two tasks, with task 1 and being $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$ the corresponding velocities; such a situation is depicted in Figure 2.9.

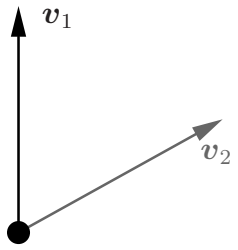

Figure 2.9: Velocity vectors in the case of two behaviors: $\boldsymbol{v}_1$ is the velocity vector corresponding to the highest priority task, $\boldsymbol{v}_2$ is the velocity vector corresponding to the lowest priority task.

In Figure 2.10 is depicted the overall output vector as generated by the motor-schema approach. The overall output velocity is a weighted combination of the two tasks velocities and, in general, it does not ensure the fulfilment neither of the first or the second task.

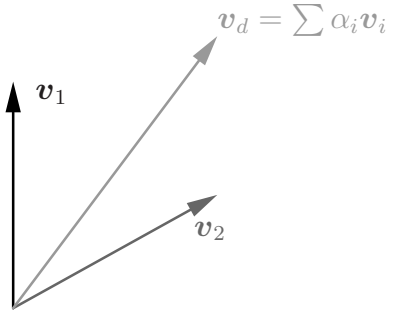

Figure 2.10: Task velocity composition in the motor-schema control case. The overall output velocity $\boldsymbol{v}_d$ is a weighted sum of the task velocities $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$.

In Figure 2.11 is depicted the overall output vector as generated by the layered-control approach. The overall output velocity is equal to the velocity generated by the highest priority task.

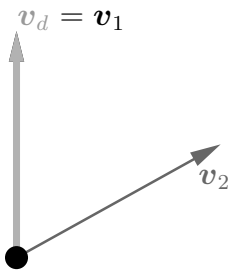

Figure 2.11: Task velocity composition in the layered-control case. The overall output velocity $\boldsymbol{v}_d$ is equal to the velocity generated by the highest priority task $\boldsymbol{v}_1$.

Finally, in Figure 2.12 the Null-Space-Based-behavioral control case is depicted. The velocity corresponding to task 2 is projected into the null space of the task 1 Jacobian ($\mathcal{N}_1$); then, the velocity components affecting the primary task are cut off and the remaining component is added to the velocity $\boldsymbol{v}_1$ corresponding to task 1. Clearly, in this simple case the compatibility condition among these two tasks is equivalent to the orthogonality between the vectors $\boldsymbol{v}_1$ and $\boldsymbol{v}_2$.

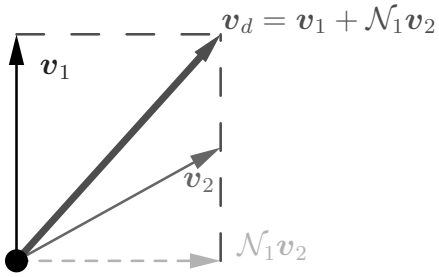

Figure 2.12: Task velocity composition in the NSB case. The velocity $\boldsymbol{v}_2$ is projected into the null space of the task 1 and added to $\boldsymbol{v}_1$.

As further example of these concepts, let consider the task of keeping n robots on a circumference with radius r and center $\boldsymbol{c}$; clearly the robots can move on the circumference without affecting the given task. All this motions are in the null space of this task; the main idea is to use them to satisfy lower priority tasks, as can be, for example, the escorting task [11].

2.2.3 Task Compatibility analisys

After having introducted the NSB approach, it is worth investigating the mathematical conditions for tasks'compatibility and fulfilment. first, We will consider case of two tasks,

then, the case of three tasks one and, finally, the general case of N tasks.
It is straightforward to show that the first task is always fulfilled, since from equations (2.16)

$$\boldsymbol{\sigma}_1 = \boldsymbol{J}_1\boldsymbol{v}_d = \boldsymbol{J}_1\boldsymbol{v}_{d,1} + \boldsymbol{J}_1\mathcal{N}_1\boldsymbol{v}^{(2)} = \boldsymbol{J}_1\boldsymbol{v}_{d,1} = \boldsymbol{\sigma}_{d,1}, \tag{2.18}$$

as $\boldsymbol{J}_1\mathcal{N}_1$ is identically null. With reference to the case of two tasks, in [35] a detailed analysis of algorithm singularities has been carried out. In particular, the tasks' compatibility can be expressed by means of the annihilating condition [7]

$$\boldsymbol{J}_2\boldsymbol{J}_1^{\dagger} = \boldsymbol{O}, \tag{2.19}$$

or, equivalently,

$$\mathcal{R}(\boldsymbol{J}_2^T) \subseteq \mathcal{N}(\boldsymbol{J}_1), \tag{2.20}$$

where $\mathcal{R}(\cdot)$ denotes the range of a matrix. In this case, in fact,

$$\begin{aligned} \boldsymbol{\sigma}_2 &= \boldsymbol{J}_2\boldsymbol{v}_d = \boldsymbol{J}_2\boldsymbol{v}_{d,1} + \boldsymbol{J}_2\mathcal{N}_1\boldsymbol{v}_{d,2} = \boldsymbol{J}_2\boldsymbol{J}_1^{\dagger}\boldsymbol{\sigma}_{d,1} + \boldsymbol{J}_2(\boldsymbol{I} - \boldsymbol{J}_1^{\dagger}\boldsymbol{J}_1^T)\boldsymbol{v}_{d,2} \\ &= \boldsymbol{J}_2\boldsymbol{v}_{d,2} - \boldsymbol{J}_2\boldsymbol{J}_1^{\dagger}\boldsymbol{J}_1^T\boldsymbol{v}_{d,2} = \boldsymbol{J}_2\boldsymbol{v}_{d,2} = \boldsymbol{\sigma}_{d,2}. \end{aligned} \tag{2.21}$$

In [7] the tasks combination strategy illustrated above is denoted as *successive inverse-based* projection strategy.
With reference to thecase of three tasks, let us define two tasks as *independent* if

$$\rho(\boldsymbol{J}_i^{\dagger}) + \rho(\boldsymbol{J}_j^{\dagger}) = \rho([\boldsymbol{J}_i^{\dagger}\ \boldsymbol{J}_j^{\dagger}]),$$

where $\rho(\cdot)$ denotes the rank of a matrix; then three tasks are compatible if two successive tasks are annihilating, if all couple of tasks are independent and by means of a proper choice of matrix $\boldsymbol{\Lambda}_i$ in equation (2.13). Finally, in the general case of N tasks, simple conditions for compatibility do not exist, and a very restricting sufficient condition is to require the annihilation among all the tasks. The last important question that needs to be addressed is the number of tasks that can be handled simultaneously. Let us suppose that the systems has n-DOF and that the rank of the Jacobian of the primary task (supposed non-singular) is r_1, then the available DOFs for the secondary task, supposing to be not conflicting with the primary one, is $n - r_1$. Iterating such a reasoning, and supposing all the tasks not conflicting, it is possible add tasks as far as $\sum r_i \leq n$; if the equality relationship is satisfied, no further tasks can be added as they will projected in an empty space.

2.2.4 Examples of application of NSB

In this Section two simple tasks formulated and handled in the NSB sense are presented. For a detailed list and dissertation about tasks handled in the NSB framework see [9],[11]. Let us consider a platoon of n grounded robots moving in a plane; the position of each robot is

$$\boldsymbol{p}_i = \begin{bmatrix} x_i & y_i \end{bmatrix}^T \in \mathcal{R}^{2\times 1}, \tag{2.22}$$

being the component z_i identically null. In this situation, the number of overall available DOFs is $2n$, being 2 the DOFs of each robot.

Barycenter

The barycenter of the platoon is defined as the mean value of robot positions:

$$\boldsymbol{\sigma}_b = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{p}_i, \tag{2.23}$$

the corresponding Jacobian matrix is

$$\boldsymbol{J}_b = \frac{1}{n}\begin{bmatrix} \cdots & 1 & 0 & \cdots \\ \cdots & 0 & 1 & \cdots \end{bmatrix} \in \mathbb{R}^{2\times 2n}, \tag{2.24}$$

that is a rank-2 matrix and independent of the platoon configuration. The pseudo-inverse of matrix $\boldsymbol{J}_b$ is

$$\boldsymbol{J}_b^{\dagger} = \begin{bmatrix} \vdots & \vdots \\ 1 & 0 \\ 0 & 1 \\ \vdots & \vdots \end{bmatrix} \in \mathbb{R}^{2n\times 2}. \tag{2.25}$$

It can be shown that the null projector matrix $\mathcal{N}_b = \left(\boldsymbol{I} - \boldsymbol{J}_b^{\dagger}\boldsymbol{J}_b\right)$

$$\mathcal{N}(\boldsymbol{J}_b) = \begin{bmatrix} 1-\frac{1}{n} & 0 & -\frac{1}{n} & 0 & \cdots & -\frac{1}{n} & 0 \\ 0 & 1-\frac{1}{n} & 0 & -\frac{1}{n} & \cdots & 0 & -\frac{1}{n} \\ -\frac{1}{n} & 0 & 1-\frac{1}{n} & 0 & \cdots & -\frac{1}{n} & 0 \\ 0 & -\frac{1}{n} & 0 & 1-\frac{1}{n} & \cdots & 0 & -\frac{1}{n} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -\frac{1}{n} & 0 & \frac{1}{n} & 0 & \cdots & 1-\frac{1}{n} & 0 \\ 0 & -\frac{1}{n} & 0 & \frac{1}{n} & \cdots & 0 & 1-\frac{1}{n} \end{bmatrix} \in \mathbb{R}^{2n\times 2n}. \tag{2.26}$$

has rank $2(n-1)$. Then, if $n = 1$ this task saturates all the available DOFs.

Variance

Another useful task when dealing with platoons of robots, is the task variance that allows to control the spreading around the barycenter. One possibile formulation of this task is the following:

$$\boldsymbol{\sigma}_v = \frac{1}{n}\sum_{i=1}^{n}\begin{bmatrix} x_i - \boldsymbol{\sigma}_{b,x} \\ y_i - \boldsymbol{\sigma}_{b,y} \end{bmatrix} \in \mathbb{R}^{2\times 1}. \tag{2.27}$$

The corresponding Jacobian matrix has the following expression:

$$\boldsymbol{J}_v = \frac{2(n-1)}{n^2}\begin{bmatrix} \cdots & x_i - \boldsymbol{\sigma}_{b,x} & 0 & \cdots \\ \cdots & 0 & y_i - \boldsymbol{\sigma}_{b,y} & \cdots \end{bmatrix} \in \mathbb{R}^{2\times 2n}, \tag{2.28}$$

whose rank is 2 except for the unlikely case of all vehicles concentrated in the same point; this task has no sense when only one vehicle is considered. The pseudoinverse $\boldsymbol{J}_v^{\dagger}$ has the following expression

$$\boldsymbol{J}_b^{\dagger} = \frac{n^2}{2(n-1)}\begin{bmatrix} \vdots & \vdots \\ \frac{x_i - \boldsymbol{\sigma}_{b,x}}{\sum_{k=1}^{n}(x_k - \boldsymbol{\sigma}_{b,x})} & 0 \\ 0 & \frac{y_i - \boldsymbol{\sigma}_{b,y}}{\sum_{k=1}^{n}(y_k - \boldsymbol{\sigma}_{b,y})} \\ \vdots & \vdots \end{bmatrix} \in \mathbb{R}^{2n\times 2}. \tag{2.29}$$

As in the previous case, in the hypotheses of a full rank matrix $\boldsymbol{J}_v$, the null projector matrix $\mathcal{N}_v$ has rank $2(n-1)$.

Example of compatibility analysis of tasks barycenter and variance

Let us consider again the case of an n robot platoon. A very common problem when dealing with such systems could be the control of the barycenter of the platoon and, at the same time, the spreading of the platoon around the barycenter [11]. This can be easily achieved by combining, in the NSB framework, the task barycenter and the task variance, considering the first one as the highest priority task and the second one as the lowest priority task. The overall commanded velocity is given by

$$\boldsymbol{v}_d = \boldsymbol{J}_b^{\dagger}\left[\dot{\boldsymbol{\sigma}}_{d,b} - \boldsymbol{\Lambda}(\boldsymbol{\sigma}_{d,b} - \boldsymbol{\sigma}_b)\right] + (\boldsymbol{I} - \boldsymbol{J}_b^{\dagger}\boldsymbol{J}_b)\boldsymbol{J}_v^{\dagger}\left[\dot{\boldsymbol{\sigma}}_{d,v} - \boldsymbol{\Lambda}(\boldsymbol{\sigma}_{d,v} - \boldsymbol{\sigma}_v)\right]. \tag{2.30}$$

As discussed in Section 2.2.3, in order to get the convergence of $\boldsymbol{\sigma}_b$ and $\boldsymbol{\sigma}_v$, respectively, to $\boldsymbol{\sigma}_{d,b}$ and $\boldsymbol{\sigma}_{d,v}$, it is necessary to investigate the condition

$$\mathcal{R}(\boldsymbol{J}_v^T) \subseteq \mathcal{N}(\boldsymbol{J}_b).$$

This condition holds when the columns of $\boldsymbol{J}_v^T$ can be expressed as linear combination of columns $\mathcal{N}_b = \mathcal{N}(\boldsymbol{J}_b)$. To this aim, let us consider, for example, the first column of $\boldsymbol{J}_v^T$

$$\boldsymbol{c}_1 = \begin{bmatrix} x_1 - \boldsymbol{\sigma}_{b,x} & 0 & x_2 - \boldsymbol{\sigma}_{b,x} & \cdots & x_n - \boldsymbol{\sigma}_{b,x} \end{bmatrix}^T \in \mathbb{R}^{2n \times 1}.$$

We have to show that

$$\exists \boldsymbol{\alpha}_x \colon \boldsymbol{c}_1 = \mathcal{N}(\boldsymbol{J}_b)\boldsymbol{\alpha}_x. \tag{2.31}$$

for some value of the vector of coefficients $\boldsymbol{\alpha}_x \in \mathbb{R}^{2n \times 1}$. It is straightforard to see that, taking into account that $\boldsymbol{\sigma}_{b,x} = \frac{1}{n}\sum_{i=1}^{n} x_i$,

$$\boldsymbol{\alpha}_x = \begin{bmatrix} x_1 & 0 & x_2 & \cdots & x_n & 0 \end{bmatrix}^T.$$

The same reasoning holds for the second column of $\boldsymbol{J}_v^T$, where, in this case

$$\boldsymbol{\gamma}_y = \begin{bmatrix} 0 & y_1 & 0 & y_2 & \cdots & y_n \end{bmatrix}^T.$$

In sum, $\mathcal{R}(\boldsymbol{J}_v^\dagger) \subseteq \mathcal{N}(\boldsymbol{J}_b)$, i.e. the two tasks barycenter+variance are fully compatible.

Chapter 3

The Patrolling Mission

In this chapter a brief overview of the main multi-robot security applications and of some experimental setup will be done. In particular, we will concentrate on the multi-robot patrolling mission, defining its aims, applications and the state of art in this field. Then, a general decentralized architecture in the framework of the Null-Space-behavioral approach will be described in detail and applied to this mission.

3.1 Collaborative Multi-Robot Systems for Security Tasks

As stated in Chapter 1, MRSs present several advantages in performing tasks with respect to a single robot system. In particular, collaborative multi-robot teams have great potential to add capabilities and minimize risk within 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 from they own, they are not very sophisticated in terms of artificial intelligence (AI); hence, most of them are remote-controlled by human operators. This is due also to the risks connected to robots' failures in such a scenarios. For these reasons, the terms *unmanned ground vehicles* (UGVs) or *unmanned aerial vehicles* (UAVs) are , often, used in lieu of the term *robot*.

There has been a rapidly increasing military research and development effort into different application scenarios, from tiny spy robots, that can be hurled into enemy buildings for surveillance purposes, to full-size support vehicles that can drive themselves and supply troops in the field. The U.S. Military has invested and is investing heavily in research and development towards testing and deploying increasingly automated systems in the military field. The Defense Advanced Research Projects Agency (DARPA) has organized

competitions in 2004 and 2005 for a prize of 2 million dollars to develop unmanned ground vehicles capable of navigating through rough terrain, like deserts.
The motivations behind these huge efforts are in the importance of the applications in wich these robots can be involved.

One of the relevant tasks is de-mining [111]. It consists in clearing roads and fields from mines before, during or after conflicts. Different types of de-mining that are currently distinct disciplines need to be combined. Especially tactical and post-conflict de-mining have quite different requirements in de-mining and accuracy. Robots should be capable of detecting all types of mines and optionally mark them. Wherever possible, they should either remove the mine or disarm it in place and, when a mine is found, warn the operator providing the mine's location and type, including the estimated time and success rate to clear the mine. Optionally, the UGV should inform the operator when the mine is marked or disarmed. It should be possible to operate the vehicles both in a teleoperated and in an autonomous mode when clearing the assigned area or route. In particular, the UGV should be usable on all kinds of roads and fields as well as in urban terrain, operate under all climatic and weather conditions and be able to receive and use airborne information on possible mine locations.

In the convoying or transport of goods task [85], UGVs are used to transport necessary goods both on roads and in heavy terrain. This type of UGVs should be able to transport goods in an adequate time, meaning in a time comparable to that of a human driver.
The UGV should either be capable of loading and unloading itself or be capable to be handled by one or more specialized UGVs in the convoy. The location of the convoy should be known at all times. Even if the operator have to be able to take over control of these vehicles at any time, they have also to be able to move autonomously in certain conditions. Finally, very important is the capability of this type of UGV to mix in normal traffic, including all the involved legal issues.

Another important tasks is to check vehicles and people for explosives and weapons at checkpoints. In this task, robot systems need to approach a suspected vehicle or person and search for weapons and explosives. Whenever such devices have been found, robots, in particular UGVs, should alert the human operator and forbid the vehicle and person to

move away. Moreover, the type of explosive needs to be identified alerting all the people in the vicinity. For optimal usability, the UGV should memorize cars and persons spotted at the checkpoint for later analysis.

Finally, the last important task have been individuated is the reconnaissance and surveillance task. In this task, UGVs are used for zone, area, route and point reconnaissance. UGVs, but also UAVs, have to give information about location and movement of persons and vehicles. The robots should also provide information on contaminated locations, map information on proper maps and warn the operator when a threat for an area is detected.
These vehicles should work under all weather conditions, in all climates and in all sorts of terrain, and communicate with other manned or unmanned vehicles to point targets and to identify persons and vehicles with high reliability.

The use of autonomous robots in this field is promising because of the advantages they present. Some of them are:

- the lack of training required for robotic pilots;
- autonomous vehicles can perform maneuvers which could not, otherwise, be done with human pilots;
- vehicle designs do not require a life support system;
- autonomous robots can actively collect and maintain timely and accurate situational awareness;
- more robotic units can be controlled by a single operator.

However, the largest drawback of robotic systems is their partial inability to accommodate for non-standard conditions. Advances in artificial intelligence in the future might help to rectify this.
The most prominent system currently in use is the UAV (as IAI Pioneer and RQ-1 Predator in Figure 3.1) which can be remotely operated from a command center in reconnaissance tasks. In particular, RQ-1 Predator is advised above all for its long term autonomous flight capabilities. This vehicle is also equipped with high resolution visual and infrared sensors, integrated GPS (Global Positioning System), and can eventually communicate

with humans. Sandia National Laboratories developed the Surveillance And Reconnais-

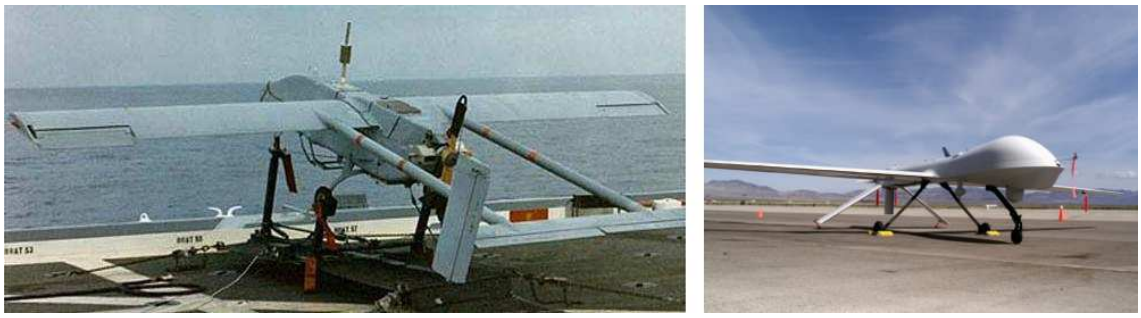

Figure 3.1: Example of UAV vehicles. On the left: IAI Pioneer. On the right: RQ-Predator.

sance Ground Equipment (SARGE) robotic vehicle for the US Department of Defense [88]. SARGE, in Figure 3.2 (top left), is a teleoperated robot for battlefield surveillance applications without computing power, aimed at supporting autonomous navigation or vision processing. The second version, SARGE II, employs differential GPS (DGPS) for autonomous navigation and neural network processing of Sandia's Laser Radar (LADAR) range images for local obstacle detection and avoidance.

Robotic System Inc. developed the Mobile Detection Assessment and Response System-Exterior (MDARS-E) [42], in Figure 3.2 (top right). The MDARS-E platform is built around a diesel-powered, hydrostatically driven vehicle. Its navigation employs a DGPS receiver integrated with inertial and landmark referencing sensors. MDARS-E uses low-bandwidth radio-frequency communications to transmit its real-time position for display on a site map and innovative video compression, image processing, radar and spread spectrum techniques. The MDARS goal is to provide multiple mobile platforms that perform random patrols within assigned areas of warehouses.

In [98], an Autonomous Robot for Surveillance Key Applications (ARSKA) is presented. This UGV, in Figure 3.2 (bottom left), is built around an all-terrain vehicle and is equipped with ultrasonic sensors for contour-following and obstacle avoidance. Two positioning systems have been used: an external optical measurement device, and DGPS. ARSKA can be controlled using a ground station, whose basic functions are task preparation, monitoring and on-line telecontrol of the vehicle. ARSKA cab be used for survellance tasks.

Figure 3.2: Example of military vehicles. On the top left: SARGE. On the top right: MDARS. On bottom left: ARSKA. On bottom right: Guardium

3.2 The Patrolling Task

The patrolling task is defined as the act of walking or traveling around an area, at regular intervals, in order to protect or supervise it [33].

The mission of patrolling a given region or border is of critical interest in modern societies. Border patrol is probably entering a new era since several countries are replacing humans performing such tasks with robots with increasing autonomy. As reported in Section 3.1, examples, where advanced testing facilities have been installed, include United States, Israel and South Korea. Similar projects, with guardian capabilities of civilian and military spaces, run in Japan, Singapore and Europe.

Perimeter surveillance algorithms form the basis for effective monitoring in a number of applications including monitoring oil spills [37], contaminant clouds, algae bloom [19], forest fires [27], and border security [49]. In [94], algorithms that use image data from multiple cameras to determine a perimeter breach are used. In [84], authors use a chain of ultrasound sensors with a simple detection scheme to identify border crossings, while in [16] radar equipment is used to identify when people or vehicles come too close to runways. For spill monitoring and other dynamic perimeter scenarios, surveillance vehicles are equipped with chemical concentration sensors [71], infrared cameras [27] or standard optical cameras [37].

As stated in Section 3.1, a few robot patrolling applications have been developed, such

as MDARS [49], or the RQ-5 Hunter UAV. Recently, commercial border patrol application have been proposed as well, such as Guardium and the unmanned autonomous speed boat *Protector* manufactured by Rafael Armament Development Authority Ltd.

Clearly, the patrolling task is by nature a multi-agent task and there are a wide variety of problems that may be reformulated as particular patrolling task (such as rescuing, tracking, detecting). Recently, several architectures for multi-agent systems have been proposed and evaluated on the patrolling problem, giving encouraging results [61]. In particular, in [61] it was shown that simple strategies implemented through reactive agents with a small amount of communication can achieve impressive results. Also, some authors of these papers suggested that an approach based on partitioning the territory, such that each agent patrols in its own region, could also work well.

It is difficult to give an exact definition of robotic border patrol, in the sense that a patrolling mission may require different objectives to be fulfilled and may be subject to several constraints, depending on various conditions, such as the specific robot locomotion system, the kind and size of the border to patrol, the civilian or military applications, the number of available robots, their equipment and communication capabilities.

In [62] an analysis of the main patrolling task issues and some multi-agent-based solutions are presented. Several features, such as agents type, agents communication, coordination scheme, agents perception and decision-making, are evaluated by using different evaluation criteria. The achievements in [62] have been further extended in [6], pointing out in more detail advantages and disadvantages of each multi-agent architecture, as well as the impact of the border geometry on the performance. In [57] and [32], graph-theory is used to find the optimal solution of a mathematical problem expressing a multi-robot surveillance problem. In [5], the authors analyze non-deterministic paths for a group of homogeneous mobile robots patrolling a frontier, under the assumption of an hostile agent trying to enter the area, where the latter has full knowledge of the algorithm.
However, in most of the above mentioned works the patrolling problem is approached from an analytical perspective, having in mind centralized control solutions. Analytical approaches are not necessarily a good choice from a practical point of view [5], since they are likely to make the patrolling algorithm predictable. On the other hand, a pure random motion of the robots is unlikely to be effective [62].

3.3 Control Approaches for Performing Tasks

The control problem described above requires an high level of autonomy and several tasks to be performed such as, e.g., planning, navigation, agent recognition, eventual communication, environment awareness, or the implementation of some kind of reasoning method; the last are known in the literature with the name of Artificial Intelligence (AI) [47]. AI methods generally consist in a deliberation based on symbols under first order predicate logic or probability theory. They require a proper Knowledge Representation (KR) since the reasoner, here the mobile robot, needs a proper representation of the environment to reason about; from a general point of view this problem is still unsolved [47]; here, KR should satisfy the strict requirements of time constraints and robustness, and thus an epistemological rich formalism is left behind for a tractable and quick inference problem. Among the various kind of formalism studied, logic and probability theory are the most widely used; moreover, one appealing method is represented by Fuzzy Logic, widely applied also in low level control [40]. Fuzzy logic introduces the concept of degree of objectivity, in the sense that it is possible for a statement to be partly true and partly false. Moreover, the inference process can be written with linguistic rules not far from simple inference sentences that humans might pronounce; this characteristics is of help in coding the experience of non-expert operators. Finally, fuzzy logic is compatible with time constraints that usually force the AI methods to provide sub-optimal solutions. AI falls into the category of deliberative methods, the opposite concept is given by the reactive methods [74], i.e., the sequence Think, Then Act is modified into Do not Think, Re-Act for reactive systems. Reactive control connects the input to the output without using a Knowledge Representation, and thus, without any internal state, by taking its inspiration from the biological notion of stimulus-response. Another popular control method, described in Chapter 2, is the so-called behavioral control. It is now usefull to clarify some aspects concerning behavioral control and the supposed emergence of an overall, macroscopic, behavior by connecting several elementary behaviors. From an engineering perspective, as the border patrol approached here, the control problem is given by its macroscopic definition, the successive decomposition into elementary behaviors is aimed at decomposing a complex problem into simpler ones. Given some technological constraints, is then possible to properly define the behaviors and, e.g., allow or not communication among robots. On the other hand, in several biological or neurological studies, local interactions are first modeled and the overall, macroscopic, behavior of the dynamic system is seen as emergent

and analyzed for what it is. Somehow, there is a kind of dichotomy between a top-down and a bottom-up behavioral approaches where we do consider appropriate to follow the former. The above discussion can be easily extended to multiple robots performing a given mission. It is worth noticing that, the border patrolling is inherently distributed and that the use of several robots give to the mission the mandatory characteristics of robustness and fault tolerance not achievable with the use of a single robot. According to the literature on multiple robots [83], the problem approached here is close to what is usually defined as swarm robotics, i.e., a large group or robots that interact implicitly. The fact that the robots do not communicate explicitly does not mean that there is not information exchange; the robots may communicate indirectly, or stigmergically, leaving, intentionally or not, some traces in the environment. One example of this kind is given by the ants, it seems that ants are able to count other ants encountered during the travel and decide acordingly their direction.

3.4 A Centralized Solution to the Patrolling Problem

Before to illustrate the proposed decentralized solution to the patrolling problem, it is worth showing as the problem of surveilling a linear border can be solved by properly defining tasks functions in the centralized NSB framework. Consider N robots and a G^1 curve B (see [20]) whose parametric expression is $p_B(s) \in \mathbb{R}^2$where $s \in [s_0,\ s_f]$ is the curvilinear abscissa. Let be $\boldsymbol{p}_i \in \mathbb{R}^2$ the position of the i-th robot; for the sake of simplicity, the following dynamic for the robot will be considered:

$$\dot{\boldsymbol{p}}_i = \boldsymbol{v}_i, \tag{3.1}$$

where $\boldsymbol{v}_i \in \mathbb{R}^2$ is the control input homogeneous to a velocity. Our aim is to generate the control inputs $\boldsymbol{v}_i$ forcing the robots to approach the border, equally distribute along the border and synchronously move along it, whatever is the shape of the border. One possible solution consists in defining proper elementary task functions and then combining them in the NSB framework. To this aim, three behaviors are needed.

Primary Behavior: *Reach Frontier*

The highest priority behavior allows the robot to reach the border. The first step consists in defining a task function that takes into account the distance between thr robots and

the border:

$$\sigma_1 = \frac{1}{2N} \sum_{i=1}^{N} \| \boldsymbol{p}_i - \boldsymbol{p}_{B,i} \|^2, \tag{3.2}$$

where $\boldsymbol{p}_{B,i} \in \mathbb{R}^2$ is the closest point to the i-th robot belonging to the border B. The corresponding jacobian of task (3.2) is:

$$\boldsymbol{J}_1 = \frac{1}{N} \left[\; (\boldsymbol{p}_1 - \boldsymbol{p}_{B,1})^T, \quad (\boldsymbol{p}_2 - \boldsymbol{p}_{B,2})^T, \quad \cdots, \quad (\boldsymbol{p}_N - \boldsymbol{p}_{B,N})^T \; \right] \in \mathbb{R}^{1 \times 2N}, \tag{3.3}$$

Then, the control input:

$$\boldsymbol{v}^1 = \boldsymbol{J}_1^T \left(\boldsymbol{J}_1 \boldsymbol{J}_1^T \right)^{-1} \left[\boldsymbol{\sigma}_{d,1} + k_1 (\boldsymbol{\sigma}_{d,1} - \sigma_1) \right], \tag{3.4}$$

whith $\boldsymbol{v} = [\boldsymbol{v}_1^T, \boldsymbol{v}_2^T, \cdots, \boldsymbol{v}_2^T]^T \in \mathbb{R}^{2N}$, $k_1 > 0$,. Clearly, by choosing $\boldsymbol{\sigma}_{d,1} = 0$, (3.4) ensures the convergence to zero of the task function (3.2).

Secondary Behavior: *Spread Along the Border*

The secondary behavior ensures that the robots, once they are on the border, equally spread along the border. Let be s_i the curvilinear abscissa corresponding to the point $\boldsymbol{p}_i$, and consider the following task function:

$$\boldsymbol{\sigma}_2 = \left[\; (s_2 - s_1), \quad (s_3 - s_2), \quad \cdots, \quad (s_N - s_{N-1}) \; \right]^T \in \mathbb{R}^{N-1}. \tag{3.5}$$

The time derivative is:

$$\dot{\boldsymbol{\sigma}}_2 = \begin{bmatrix} -1 & 1 & 0 & \cdots & 0 \\ 0 & -1 & 1 & \cdots & 0 \\ & & \cdots & & \\ 0 & 0 & \cdots & -1 & 1 \end{bmatrix} \begin{bmatrix} \dot{s}_1 \\ \dot{s}_2 \\ \vdots \\ \dot{s}_N \end{bmatrix} = \boldsymbol{J}_s \dot{\boldsymbol{s}}, \tag{3.6}$$

where $\boldsymbol{J}_s \in \mathbb{R}^{(N-1) \times N}$ and $\dot{\boldsymbol{s}} \in \mathbb{R}^N$. Since $\dot{s}_i = \boldsymbol{t}(s_i)^T \dot{\boldsymbol{p}}_i$, where $\boldsymbol{t}(s_i) \in \mathbb{R}^2$ is the unit vector tangent to the curve B at the point $\boldsymbol{p}_B(s_i)$, supposed oriented in the positive sense of the border, equation (3.6) can be rewritten as:

$$\dot{\boldsymbol{\sigma}}_2 = \boldsymbol{J}_s \begin{bmatrix} \boldsymbol{t}(s_1)^T \\ \boldsymbol{t}(s_2)^T \\ \vdots \\ \boldsymbol{t}(s_N)^T \end{bmatrix} \dot{\boldsymbol{p}} = \boldsymbol{J}_2 \dot{\boldsymbol{p}}, \tag{3.7}$$

with jacobian $\boldsymbol{J}_2 \in \mathbb{R}^{(N-1)\times 2N}$ and $\boldsymbol{p} = [\boldsymbol{p}_1^T, \boldsymbol{p}_2^T, \cdots, \boldsymbol{p}_2^T]^T \in \mathbb{R}^{2N}$. Finally, to have the robots equally spread along the border, the desired value of the task function is $\boldsymbol{\sigma}_{d,2} = \frac{L}{N}\boldsymbol{I}_{N-1}$, being L lenght of the border and $\boldsymbol{I}_k$ a $(k \times k)$ square identity matrix. Then, the control input corresponding to this task is:

$$\boldsymbol{v}^2 = \boldsymbol{J}_2^T \left(\boldsymbol{J}_2 \boldsymbol{J}_2^T\right)^{-1} \left[\boldsymbol{K}_2(\boldsymbol{\sigma}_{d,2} - \boldsymbol{\sigma}_2)\right], \tag{3.8}$$

where $\boldsymbol{K}_2 \in \mathbb{R}^{(N-1)\times(N-1)}$ is a positive definite matrix and $\dot{\boldsymbol{\sigma}}_{d,2} = \boldsymbol{0}$.

3.4.1 Lowest Prioriry Behavior: *Patrol Border*

After the robot have reached and equally spread along the border, they should synchronously start patrolling the border. To this aim, it is enough the robot have the same velocity component along the tangent vector to the border. Then, the output of the third task is simply:

$$\boldsymbol{v}^3 = v_p \begin{bmatrix} \boldsymbol{t}(s_1)^T \\ \boldsymbol{t}(s_2)^T \\ \vdots \\ \boldsymbol{t}(s_N)^T \end{bmatrix}, \tag{3.9}$$

where $v_p \in \mathbb{R}^1$ is the patrolling velocity.

By combining equations (3.4),(3.8), (3.9), the overall output control input is:

$$\boldsymbol{v} = \dot{\boldsymbol{v}}^1 + \boldsymbol{N}_1 \dot{\boldsymbol{v}}^2 + \boldsymbol{N}_{12} \dot{\boldsymbol{v}}^3, \tag{3.10}$$

where $\boldsymbol{N}_i$ null projector matrix of task i, and $\boldsymbol{N}_{12}$ is the null projector of stacked jacobian $\boldsymbol{J}_{12} = \begin{bmatrix} \boldsymbol{J}_1 \\ \boldsymbol{J}_2 \end{bmatrix}$.
Figure 3.3 shows the robot positions at different time instants in a simulation case study with 15 robots, $k_1 = 5$, $\boldsymbol{K}_2 = 2\boldsymbol{I}_{15-1}$. and $v_p = 10m/s$. Robots are represented by green points surrounded by a green circle; the continuous blue line represents the border to be patrolled.

As it can be seen, the robot approaches the border (first and second frames) and equally distribute along the border, starting the patrolling phase (third and fourth frames).

In Figure 3.4, the time history of the errors on the primary (top) and secondary task (bottom) are represented; as aspected, they converge asymptotically to zero.

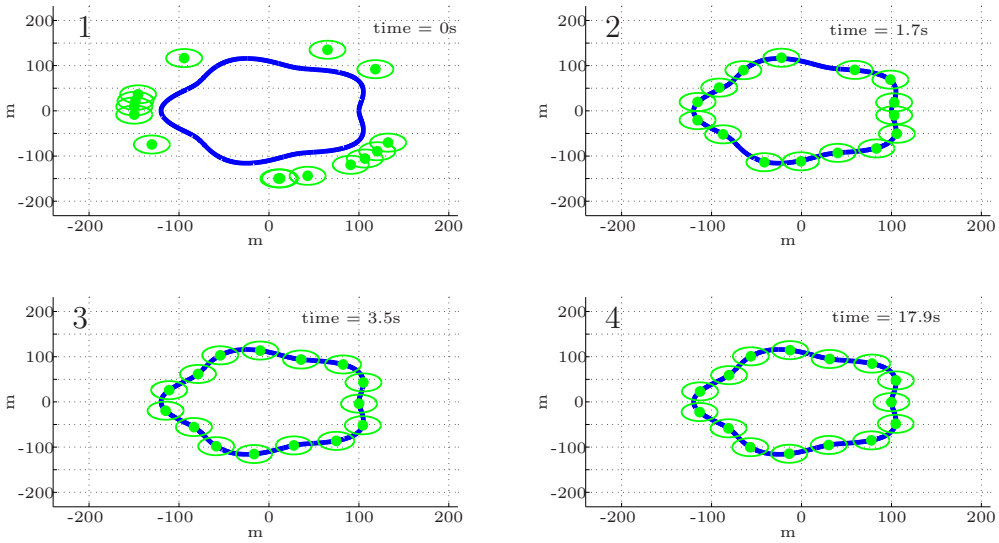

Figure 3.3: Centralized patrolling. Sample frames at different time instants of robots performing a patrolling mission.

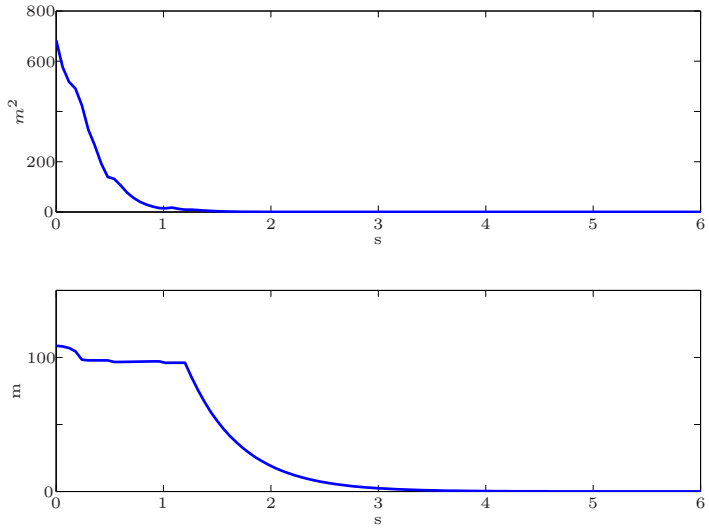

Figure 3.4: Centralized patrolling. Top: error task *Reach Frontier*. Bottom: error task *Spread Along the Border*.

While the NSB has been proven to be effective for achieving patrolling task, it suffers from some drawbacks. The first one is that it is highly centralized, i.e., a central unit is required (a leader vehicle or an external computer) that acquires the state of all vehicles, compute the velocity commands and send them back to the vehicles. Clearly, in addition to the communication problems that may arise (especially in large and/or occluded environments), a failure of the central unit causes the failure of the mission. Moreover, additional protocols are needed by the central unit to take into account the failure (addition) of one or more vehicles, in order to reduce (increase) the platoon and jacobian matrices dimension.

3.5 Decentralized Solution to the Patrolling Task

According to our terminology, the term *robot* will denote a mobile machine, while *agent* may denote both a human or a robot. The developed approach belongs to the swarm robotic class. In swarm robotics, the achievement of the mission is the result of cooperation of independent agents forming the swarm. In this approach, the overall behavior is generally *emergent*, i.e., although each unit responds to external stimuli via simple behaviors and, generally, it has not a global cognition of the assigned mission, the couples stimuli-behaviors are organized in such a way the cooperation implicitly raises from the interaction among robots and with the environment. The main advantage of this approach is its robustness to faults of the individual robots, as no predefined role is established and reorganization can be automatically achieved. Moreover, this approach is clearly modular, since new emergent behaviors at 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. For example, there is not any formal method to generate, starting from the overall mission description, a set of elementary couples stimuli-behaviors, whose interaction leads to task accomplishment, especially in the presence of dynamically changing environments. Moreover, since the architecture is decentralized and, in general, the swarm operates in dynamic environments, it is difficult to introduce performance indexes and compare these architectures with other approaches (e.g., deliberative centralized approaches). In fact, traditional benchmarking methodologies are based on the assumption of static environments. When dealing with swarms of autonomous robots, however, this concept is candidate to fail, since the environment, rather than the robots, is not repeatable (e.g., it is impossible to

have the same environment conditions, the same obstacle positions, sensor readings and, when required, the same human interaction).

3.5.1 Assumptions

The following assumptions are adopted to develop the proposed solution to the multi-robot border patrol problem.

Linear Surveillance Problem

The general perimeter surveillance problem has been reduced to the linear surveillance problem by assuming that the perimeter to be monitored is homeomorphic to a line and can therefore be represented as a single path between two points. In addition, closed perimeters are allowed. This assumption eliminates perimeters that are connected in a web-like structure. However, an arbitrary connected perimeter can be reduced to a linear perimeter by constructing a single tour that traverses all segments of the original perimeter. In particular, borders that are G^1 curve [20] has been considered; several algorithms exist to obtain such a feature starting from natural border; an example is reported in [106].

Localization

We assume that each robot is able to localize itself in the environment and that it knows a geometric description of the border or is able to detect it. Localization and, in general, Simultaneous Localization and Mapping (SLAM) [108] are well established problems in literature and will be not addressed in this thesis. To be more precise, each robot needs only to estimate its position with respect to the border to be patrolled and other agents. The way such a goal is achieved highly relies on the on-board sensors and on the practical goals to be faced.

Visibility Range

To simulate real robots sensor capability, we establish that each robot has a visibility range, where it recognizes the presence of other agents or the border itself. We assume, for simplicity, that the visibility area is a circle around the robot. This is not a unrealistic assumption, since it well approximate the case of pan-tilt-zoom camera sensors or laser range finders.

Safety Area

Each robot is characterized by its own safety area, where other agents are not allowed to enter. Also in this case, we assume that the safety area is a circle around the robot, clearly smaller than the visibility area. Such an assumption is necessary to avoid collision among robots, particularly in the case of swarm systems where a large number of robots may be present leading to conflicts and interferences.

Decentralization

Each robot is autonomous, it does not rely on a central computational unit; moreover, distributed algorithms, such as consensus, that need an explicit exchange of information are not allowed. As stated above, this is typical of swarm architectures where the overall behavior is generally *emergent*, i.e., the couples stimuli-behaviors are organized in such a way the cooperation implicitly raises from the interaction among robots and with the environment.

Communication

Between robots is forbidden any kind of explicit communications. This assumptions is devoted to supply more robustness to the algorithm. In fact, in scenarios where the perimeter is very large or terrain causes line-of-sight problems, agents may frequently be out of the communication range of the base station and neighboring vehicles. Additionally, also in the case when a UAV or UGV is in the communication range of its neighbors, the gathered data may require significant time to transmit (e.g., complete video footage).

Awareness

Each robot does not know the exact number of patrolling robot. However, it is aware of the existence of other agents, that can be other patrolling robots, friends or enemy agents. As described in the following, the awareness is implicitly taken into account by defining proper behaviors, as for example, to avoid collisions

Performance

As stated in [5] or [32], the patrolling mission might be given in analytical form, and a proper functional could be designed to be minimized by using a certain deterministic algorithm. This is, however, not necessarily the optimal solution from a practical point of view [5]. Let us consider, e.g., a functional that minimizes the time elapsed from two

visits of a certain node; the definition of a node and the definition of an optimum criterion makes the patrolling algorithm predictable. On the other hand, a pure random movement of the robots is unlikely to be effective [62]. Although the presence of performance criteria/indexes may be important, it is usually difficult to model the environment in order to perform quantitative measurements; hence, the obtained results would necessarily be confined to the specific environments considered. With these characteristics in mind, the design of the control algorithm has been driven by the aim to transfer to the robots the heuristics in performing a patrol mission, rather than approaching the problem from a pure mathematical point of view.

3.6 The overall architecture

Being the approach completely decentralized, we need to design a control architecture for the single robot. In Figure 3.5, a scheme of the overall designed architecture is given.

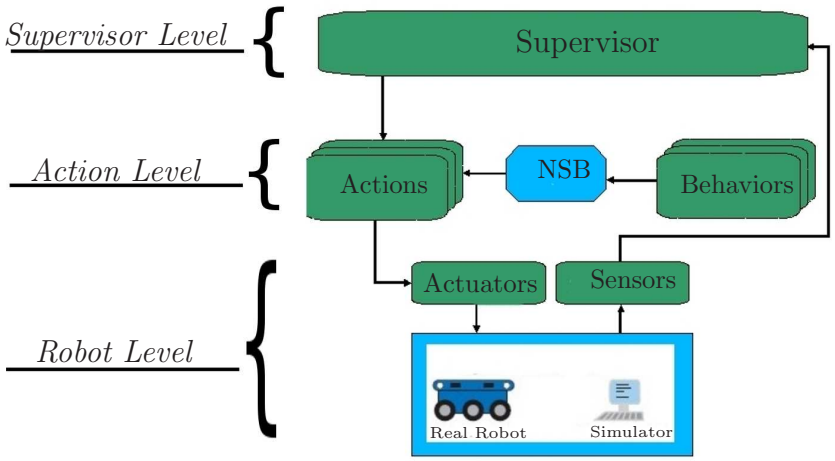

Figure 3.5: Overall Architecture.

As it can be seen, three layers have been individuated. Starting from the top, they are the *Supervisor Level*, the *Action Level* and, finally, the *Robot Level*. The first two levels are abstract. In particular, the Action Level, described in Section 3.6.1, defines the set of actions the robot can undertake. Clearly, the set of actions depends on the mission to achieve, involving different skills the robots are capable of. Once the set of actions is defined, it is important that 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. Different paradigms are possible, among them a Finite State Automata and a Fuzzy Logic Supervisor, described in Section 3.6.4, have been designed and tested. Finally, the Robot Level depends on the particular platform and mainly concerns with the available sensors and actuators. A description of the platform used to test the proposed approach will be given in Section 3.8.

3.6.1 Action Level

The basic idea is to define *elementary behaviors* and then combine them in a consistent way to form a set of high-level *actions* (see Figure 3.6). This is consistent with the ado-

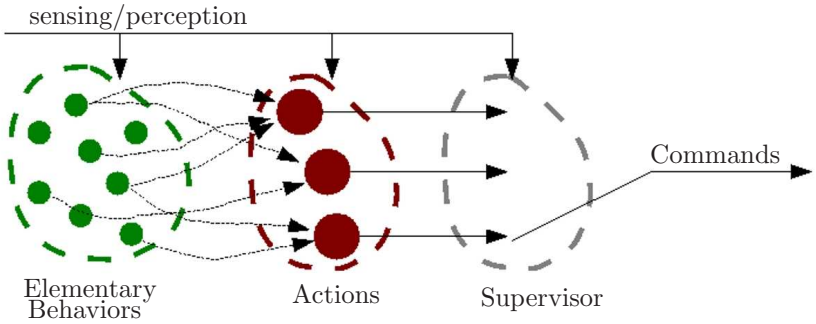

Figure 3.6: Behaviors Composition.

pted bottom-up approach in the action selection mechanism, since it allows the developer to build an intelligent system on the basis of elementary components to better handle complexity. The elementary behaviors and their combination in actions are defined in the framework of the Null-space-based approach described in Chapter 2.

3.6.2 Elementary Behaviors definition

In the following the elementary behaviors will be mathematically defined and described. They are based on the mission to achieve, identifying the elementary components required by a typical patrolling missions.

Reach Frontier

The first behavior to define in such a mission is the *Reach Frontier Behavior* (see Figure 3.7). Once activated, it allows the robot to reach the border to be patrolled. Then, given the robot position $\boldsymbol{p}_r \in R^2$ and the border B, $\boldsymbol{p}_B \in R^2$ is the closest point to $\boldsymbol{p}_r$ belonging to B. The behavior reach frontier is simply defined as the following function σ_{rf}

$$\begin{cases} \sigma_{rf} = \|\boldsymbol{p}_r - \boldsymbol{p}_B\|, \ \sigma_{rf,d} = 0, \\ \boldsymbol{J}_{rf} = \boldsymbol{r}_{rf}^T, \ \boldsymbol{J}_{rf}^\dagger = \boldsymbol{r}_{rf}, \ \boldsymbol{N}_{rf} = \boldsymbol{I}_2 - \boldsymbol{r}_{rf}\boldsymbol{r}_{rf}^{\mathrm{T}}, \\ \boldsymbol{v}_{rf} = \lambda_{rf}\boldsymbol{r}_{rf}\left(-\sigma_{rf}\right), \end{cases} \tag{3.11}$$

where the subscript "d" defined the desired value ot the behavior function, $\boldsymbol{r}_{rf} = \left(\boldsymbol{p}_r - \boldsymbol{p}_B\right)/\|\boldsymbol{p}_r - \boldsymbol{p}_B\|$, $\boldsymbol{J}_{rf}$ is the task Jacobian, $\boldsymbol{I}_2 \in R^{2\times 2}$ is the identity matrix, $\boldsymbol{N}_{rf}$ is the null-space projection matrix, λ_{rf} is a positive scalar gain and $\boldsymbol{v}_{rf}$ is the desired robot velocity.

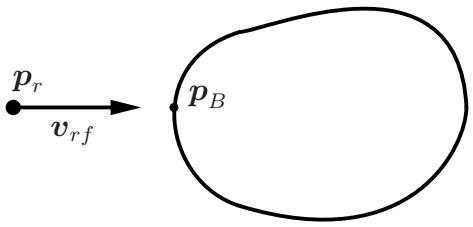

Figure 3.7: Graphical representation of the *Reach Frontier Behavior*.

It is worth noticing that the computation of the closest point $\boldsymbol{p}_B$ to the robot is needed. In particular, a discretization of the border might be required or a proper analytical approximation [5]

Patrol Frontier Clockwise

Once the robot is close to the border, it is important the robot moves 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:

$$\begin{cases} \boldsymbol{v}_{cw} = \lambda_{cw}\boldsymbol{r}_{cw}, \\ \boldsymbol{N}_{cw} = \boldsymbol{I}_2 - \boldsymbol{r}_{cw}\boldsymbol{r}_{cw}^{\mathrm{T}}, \end{cases} \tag{3.12}$$

where$\boldsymbol{v}_{cw}$ is the velocity vector encoding the behavior, $\boldsymbol{r}_{cw}$ plays the role of the task Jacobian, $\boldsymbol{N}_{cw}$ is the null-space projection matrix and λ_{cw} is a positive scalar gain. It is

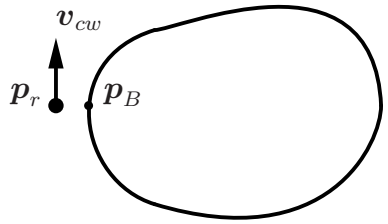

Figure 3.8: Graphical representation of the *Patrol Clockwise Behavior*.

worth noticing that this behaviors simply allow the robot to move accordingly the border tangent no matter the robot position is and, in particular if the robot is or not on the border (Figure 3.8).

Patrol Frontier Counter-Clockwise

This case is formally similar to the previous one with the obvious difference that the vector tangent to the border is oriented in the counter-clockwise direction.The behavior is then directly defined as:

$$\begin{cases} \boldsymbol{v}_{ccw} = \lambda_{ccw} \boldsymbol{r}_{ccw}, \\ \boldsymbol{N}_{ccw} = \boldsymbol{I}_2 - \boldsymbol{r}_{ccw} \boldsymbol{r}_{ccw}^{\mathrm{T}}, \end{cases} \tag{3.13}$$

where $\boldsymbol{r}_{ccw}$ plays the role of the task Jacobian, $\boldsymbol{N}_{ccw}$ is the null-space projection matrix and λ_{ccw} is a positive scalar gain.

Teammate Avoidance

In a multi-robot system, it is important to avoid collisions among robots. To this aim, let be $\boldsymbol{p}_r$ the robot position, $\boldsymbol{p}_t$ the position of the closest teammate to the robot and d_s the safety distance, i.e., the radius of the circular safety area. The behavior *Teammate Avoidance* is defined as:

$$\begin{cases} \sigma_{ta} = \|\boldsymbol{p}_r - \boldsymbol{p}_t\|, \ \sigma_{ta,d} = d_s, \\ \boldsymbol{J}_{ta} = \boldsymbol{r}_{ta}^T, \ \boldsymbol{J}_{ta}^{\dagger} = \boldsymbol{r}_{ta}, \ \boldsymbol{N}_{ta} = \boldsymbol{I}_2 - \boldsymbol{r}_{ta} \boldsymbol{r}_{ta}^{\mathrm{T}}, \\ \boldsymbol{v}_{ta} = \lambda_{ta} \boldsymbol{r}_{ta} \left(d_s - \sigma_{ta} \right), \end{cases} \tag{3.14}$$

where the subscript "d" defined 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 and λ_{ta} is a positive scalar gain and $\boldsymbol{v}_{ta}$ is the desired robot velocity.

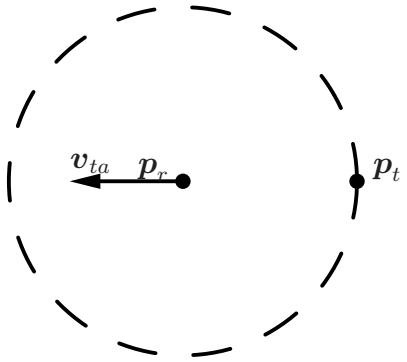

Figure 3.9: Graphical representation of the *Teammate Avoidance Behavior*.

In a few words, this behavior allows the robot to move in order to keep the other teammate on the border of its own safety area (see Figure 3.9).

Friend Avoidance

A *friend* is an agent that moves independently from other agents and that is allowed to cross the border. Therefore, when a friend tries to cross the border, patrolling agents should keep a desired distance from it without affecting its motion. This behavior is similar to the *Teammate Avoidance* behavior in Figure 3.9, then, given the robot position, $\boldsymbol{p}_r$, the friend position, $\boldsymbol{p}_f$, and a safety distance d_s, the behavior friend avoidance is defined as:

$$\begin{cases} \sigma_{fa} = \|\boldsymbol{p}_r - \boldsymbol{p}_f\|, \ \sigma_{fa,d} = d_s, \\ \boldsymbol{J}_{fa} = \boldsymbol{r}_{fa}^T, \ \boldsymbol{J}_{fa}^\dagger = \boldsymbol{r}_{fa}, \ \boldsymbol{N}_{fa} = \boldsymbol{I}_2 - \boldsymbol{r}_{fa}\boldsymbol{r}_{fa}^{\mathrm{T}}, \\ \boldsymbol{v}_{fa} = \lambda_{fa}\boldsymbol{r}_{fa}\left(d_s - \sigma_{fa}\right), \end{cases} \tag{3.15}$$

where the subscript "d" defined the desired value ot the behavior function, λ_{fa} is a positive definite diagonal matrix, N_{fa} is the null-space projector matrix and $\boldsymbol{v}_{fa}$ is the desired robot velocity. Similarly to the previous case, this behavior, keeping the robot far form the friend agent, allows it to move without being affected by patrolling agents.

3.6.3 Actions definition

The elementary behaviors defined in Section 3.6.2 are used to build the actions, i.e., more meaningful behaviors suitable for the given mission. In the following, analytical details of the actions defined for the patrolling problem are provided. As stated above, these actions

are obtained by combining the elementary behaviors. This combination is obtained via NSB in order to obtain a predictable output and, thus, requiring a priority assignment among the elementary behaviors.

Action Reach Frontier (ARF)

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*:

$$\boldsymbol{v}_{Arf} = \boldsymbol{v}_{rf}. \tag{3.16}$$

As will be clear later, this action is usefull 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.

Action Patrol Clockwise (APCW)

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*, the secondary one is *Patrol Clockwise*, leading to

$$\boldsymbol{v}_{Apcw} = \boldsymbol{v}_{rf} + \boldsymbol{N}_{rf}\boldsymbol{v}_{cw}. \tag{3.17}$$

By activating this action the robot is able to reach or keep on the border while patrolling it.
It is worth noticing, that, being the two elementary behaviors compatible in the NSB sense (since the two task velocities are orthogonal), the priority of behaviors can be exchanged. In Figure 3.10, a typical robot motion under the effect of *Action Patrol Clockwise* is shown.

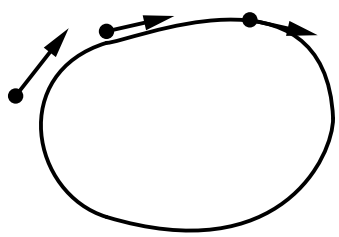

Figure 3.10: Typical robot motion under the effect of *Action Patrol Clockwise*.

Action Patrol Counter-Clockwise (APCCW)

This case is formally similar to the previous, with the obvious difference to properly consider the *Patrol Frontier Counter-Clockwise* behavior. Then, the output velocity of this action is

$$\boldsymbol{v}_{Apccw} = \boldsymbol{v}_{rf} + \boldsymbol{N}_{rf}\boldsymbol{v}_{ccw}. \tag{3.18}$$

Action Keep Going (AKG)

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* (*Patrol Frontier Counter-Clockwise*) behaviors in the NSB sense:

$$\boldsymbol{v}_{Akg} = \boldsymbol{v}_{rf} + \boldsymbol{N}_{rf}\boldsymbol{v}_{p}, \tag{3.19}$$

where $\boldsymbol{v}_p$ the vector tangent to the border at the closest point belonging to the border. The versus of the 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 to confer some unpredictability to the mission.

Action Teammate Avoidance (ATA)

When a teammate vehicle enters the safety area of the robot, it needs to avoid the teammate, while trying 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:

$$\boldsymbol{v}_{Ata} = \boldsymbol{v}_{ta} + \boldsymbol{N}_{ta}\boldsymbol{v}_{rf}. \tag{3.20}$$

In Figure 3.11, it is depicted a typical path generated by this action.

It is woth noticing, that differently from the previous actions, is not possible to ensure task compatibility in all situations. An example of such situation is represented by a teammate that is exactly between the robot and the border; in this case, the velocity component of the secondary task (*reach frontier*), will be totally cut off. In general, only the velocity components of the secondary task that do not conflict with the primary behavior will be executed, leading to the robot moving on the the border of the teammate safety area (Figure 3.11).

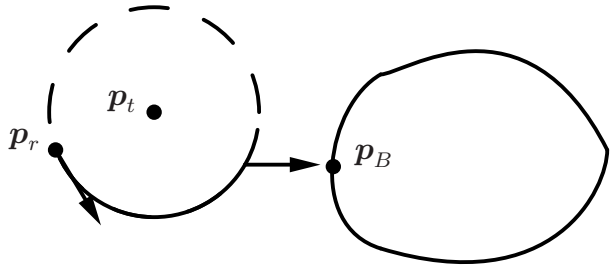

Figure 3.11: Graphical representation of the *Action Teammate Avoidance* Behavior; $\boldsymbol{p}_r$ is the poition of the vehicle, $\boldsymbol{p}_t$ the position of the teammate closest to the robot, $\boldsymbol{p}_B$ the point of the border closest to the robot

Action Friend Avoidance (AFA)

The Actio Avoid Teammate is similar to the *Action Friend Avoidance*. In the case a friend vehicle is in the safety area, the robot needs to avoid the vehicle, while trying to stay on the border, so that it can restart the patrol mission once the friend vehicle is far enough. This action can be obtained by combining the *Friend Avoidance* and *Reach Frontier* behaviors

$$\boldsymbol{v}_{Afa} = \boldsymbol{v}_{fa} + N_{fa}\boldsymbol{v}_{rf}. \tag{3.21}$$

Being *Reach Frontier* the secondary task, only its velocity components that do not conflict with the primary task will be executed. It is worth noticing that, when more than one friend is in the safety area of the robot, it can be taken into account by generalizing (3.21) in the following way:

$$\boldsymbol{v}_{Afa}(t) = \boldsymbol{v}^1_{fa} + N^1_{fa}(v^2_{fa} + N^2_{fa}\boldsymbol{v}_{rf}), \tag{3.22}$$

where the closer is the friend, the higher is the priority. Also in this case, it is not possible to ensure the task compatibility between the two behaviors in all situations.

3.6.4 The Supervisor Level

According to the assumptions reported in Section 3.5.1, for the sake of robustness and fault tolerance, each robot decides the next action to be performed based only on its sensing capabilities. Therefore, the set of actions (see Section 3.6.3) defines only the system's skills for reacting to situations encountered in its environment. Hence, the problem of selecting the action needs to be executed next has to be considered. Although several paradigms might be used, two appealing tools are represented by Finite State Machines (FSMs) and Fuzzy Logic Engines; they will be described in the following.

3.6.5 Finite State Machine Supervisor

An efficient and simple way to achieve a decisional process is based on the use of finite state automata playing the role of *Supervisor*. Finite State Machine action selection mechanisms assume that [23]:

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

While these assumptions may seem unrealistic, they have been usefully applied in sufficiently abstract models. The clearest example is probably the Conways Game of Life [46], a very early ALife system.
Using this kind of paradigm, it is better to think about agents rather than environments. To be more precise, agents tend to be simpler than there environment, i.e., they tend to have less possible behaviours than possible external situations. Thus, it is simpler to write code for actions and not for events. To build a Finite State Machine, two sets are to be determined:

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

Again, the main assumptions are that both states and transitions can be enumerated and that they are mutually exclusive. Most agents, depending on the applications, have more than two possible actions. In this case, if there are a small number of possible transitions from each action to the next, then FSMs represent a suitable selection mechanism. However, if the environment is very dynamic and unpredictable and the application very complex, then there may be transitions from every state to every other state. The number of transitions grows quadratically, since all N nodes must have $N-1$ ingoing transitions. It would be better to have a way to code action selection not growing faster than the number of possible actions. However, in many practical application not all state transitions are feasible, then, we really only need to specify the transitions that the agent can make.

A possible structure of the supervisor [68] that select the proper action is shown in Figure 3.12.
As it can be seen, it is arranged in a hierarchical way by defining states and sub-states in the following way:

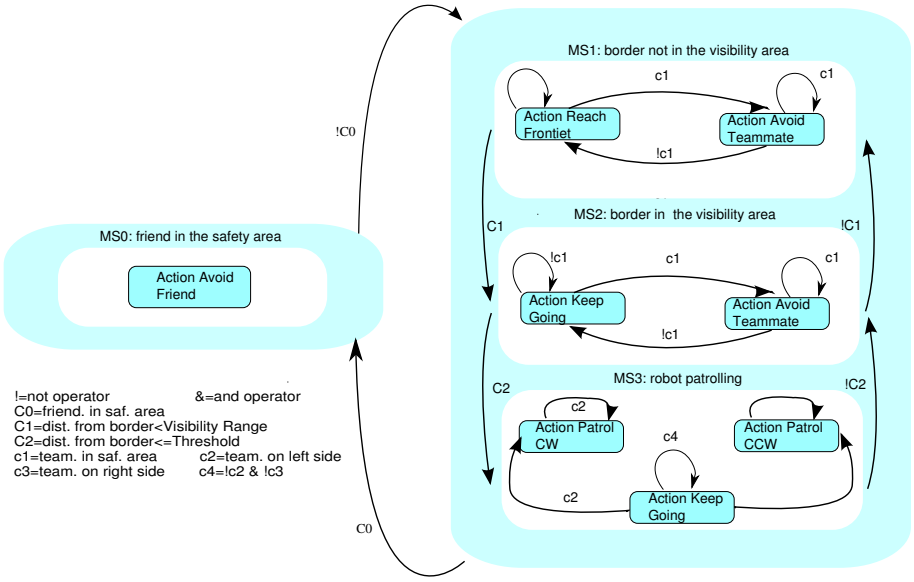

Figure 3.12: Sketch of the Supervisor.

- State MS0: **If** one or more friend agents are in the safety area **then** *Action Avoid Friend* is active **else**
  - State MS1: **If** the distance between the robot and the border is larger than the Visibility Range **and**
    - → no teammate is in the safety area **then** *Action Reach Frontier* is active,
    - → one or more teammates are in the safety area **then** *Action Avoid Teammate* is active.
  - State MS2: **If** the distance between the robot and the border is smaller than the Visibility Range and larger than a given Threshold, **and**
    - → no teammate is in the safety area **then** *Action Keep Going* is active,
    - → one or more teammates are in the safety area **then** *Action Avoid Teammate* is active.
  - State MS3: **If** the distance between the robot and the border is smaller than a given Threshold **and**
    - → no teammate is in the visibility range **then** *Action Keep Going* is active,
    - → there is a teammate on the left **then** *Action Patrol Clockwise* is active,

→ there is a teammate on the right **then** *Action Patrol Counter-Clockwise* is active.

At each time instant the robot can be in one of the *macro*-state MS0, MS1, MS2 or MS3. State MS0 corresponds to the situation where a firend agent is in the safety area of the patrolling robot. In this case Action Avoid Friend is active. On the contrary, if no friend agent is present, the robot can be in the state MS1, MS2, MS3 which correspond, respectively, to the condition in which the robot is far from the border, the robot is not far from the border (but its distance is still large) and the robot is close to the border. The main reason behind this distinction is that in the state MS1 the robots try to reach the border (and avoid the teammates), in the state MS2 the robots behave as they are patrolling the border (and avoid the teammates), in the state MS3 the robots perform the patrolling mission and do not allow robots approaching the border to influence their motion; in the last state, avoidance of other teammates patrolling the border is achieved by activating the actions *Patrol CW* and/or *Patrol CCW*.

3.6.6 Fuzzy Supervisor

In addition to the Finite State Automata described in the previous section a Fuzzy Logic Supervisor, has been designed [70]. A Fuzzy Logic Engine is in charge of encoding an expert's knowledge into a set of linguistic rules which are smoothly interpolated; the result is then defuzzified to provide a crisp actuation output. Each rule is specified as a properly shaped function and assigned to a proper range of input variable. The main advantage presented by this technique is, of course, the possibility that the solution to the problem can be cast in terms that human operators can understand, so as their experience can be incorporated into the design process. This makes it easier to automate tasks that are already successfully performed by humans, delete or add new linguistic rules. With regards to the i-th vehicle, the Fuzzy Inference System (FIS), based on the local sensors information, calculates the degree of activation $\alpha_j^i \in [0, 1]$ of j-th action ($j = 1, 2 \ldots n$), based on a classical Mamdami fuzzy type system [40]. As it may happen that sensory data match with several behavior rules (i.e., the conditional part of the fuzzy rules), more than one action might fire at the same time; hence, an arbiter should be adopted in order to deal with such a situation. However, the FIS is conceived in such a way that only one action may fire at each time step, since the arbiter selects the j-th action characterized by the maximum activation level α_j^i.

The three crisp inputs of the FIS are the distance from the border d_b^i, the distance from the closest teammate d_{ct}^i, and a variable γ^i whose expression is:

$$\gamma^i = \begin{cases} 0 & if\ d_b^c > \theta_{th2} \\ \dfrac{\theta_{th2} - d_b^c}{\theta_{th2} - \theta_{th1}} \lambda^i & if\ \theta_{th1} \leq d_b^c \leq \theta_{th2} \\ 1 & if\ d_b^c < \theta_{th1} \end{cases} \tag{3.23}$$

where d_b^c is the distance from the border of the closest teammate (indicated by the subscript c) and θ_{th1} and θ_{th2} are given thresholds. Once a curvilinear abscissa on the border is established, s^i represents the value of curvilinear abscissa of the point on the border closest to the i-th vehicle. The variable λ^i is -1 if $s^c < s^i$, 1 otherwise. Therefore, γ^i is zero if the closest teammate is *far* from the border, negative if it is *close* to the border and $s_c < s^i$, positive if it is *close* to the border and $s_c \geq s^i$. In sum, γ^i represents the minimal information about the closest teammate needed to describe its status and its position with respect to the i-th vehicle. Depending on the values of γ^i and d_{ct}^i, the FIS can decide to avoid the teammate, reverse its motion direction, or keep its actual motion state.

As stated before, the outputs of the FIS are the actions degrees of activation. The linguistic variables corresponding to the crisp inputs are

- *DistanceFromBorder* = {low, medium, high},
- *DistanceFromNeighbor* = {low, medium, high},
- *NeighborState* = {PatrollingOnMyLeft, notPatrolling, PatrollingOnMyRight},

while the output linguistic variables are

- *ReachFrontierLevel (RFL)* = {low, medium, high},
- *KeepGoingLevel (KGL)* = {low, medium, high},
- *PatrolClockwiseLevel (PCWL)* = {low, medium, high},
- *PatrolCounterClockwiseLevel (PCCWL)* ={low, medium, high},
- *AvoidTeammateLevel (ATL)* = {low, medium, high},
- *AvoidFriendLevel (AFL)* = {low, high}.

It is worth noticing that, for an open border, also the distance from the left and right end of the border need to be considered, so as to properly change the patrolling direction. As membership functions, trapezoids and triangles have been chosen, together with min-max method for and-or and implication-aggregation operations, and centroid as defuzzification method. In Figure 3.13 are shown the chosen input and output membership functions.

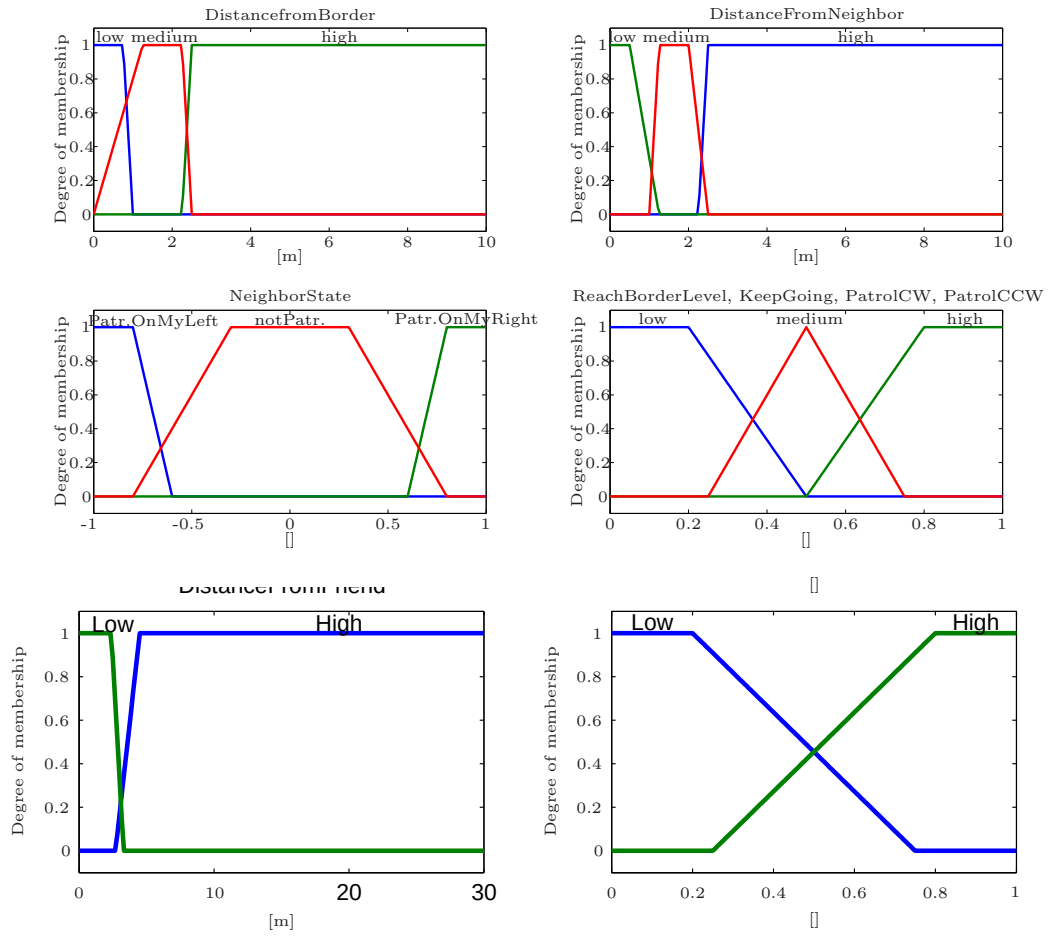

Figure 3.13: Membership functions of the input and output linguistic variables.

The number of possible states of the Supervisor is 36; the fuzzy rules depending on the state of *DistanceFromFriend* **and** *DistanceFromBorder* linguistic variable are reported in Tables 3.1-3.4.

As in the case of the FSM supervisor, the main reason of this distinction is that, when there is a friend close to the robot, then the highest priority action is Avoid Friend Action. On the contrary, when no friend is in the vicinity of the patrolling robot and the

DistanceFromFriend is low						
DistanceFromNeighbor NeighborState	low		medium		high	
PatrollingOnMyLeft	AFL	high	AFL	high	AFL	high
	ATL	low	ATL	low	ATL	low
	RFL	low	RFL	low	RFL	low
	KGL	low	KGL	low	KGL	low
	PCWL	low	PCWL	low	PCWL	low
	PCCWL	low	PCCWL	low	PCCWL	low
notPatrolling	AFL	high	AFL	high	AFL	high
	ATL	low	ATL	low	ATL	low
	RFL	low	RFL	low	RFL	low
	KGL	low	KGL	low	KGL	low
	PCWL	low	PCWL	low	PCWL	low
	PCCWL	low	PCCWL	low	PCCWL	low
PatrollingOnMyRight	AFL	high	AFL	high	AFL	high
	ATL	low	ATL	low	ATL	low
	RFL	low	RFL	low	RFL	low
	KGL	low	KGL	low	KGL	low
	PCWL	low	PCWL	low	PCWL	low
	PCCWL	low	PCCWL	low	PCCWL	low

Table 3.1: Supervisor Linguistic Rules in the case of a friend agent in the visibility range

DistanceFromFriend is high **and** DistanceFromBorder is high						
DistanceFromNeighbor NeighborState	low		medium		high	
PatrollingOnMyLeft	AFL	low	AFL	low	AFL	low
	ATL	high	ATL	high	ATL	low
	RFL	low	RFL	low	RFL	high
	KGL	low	KGL	low	KGL	low
	PCWL	low	PCWL	low	PCWL	low
	PCCWL	low	PCCWL	low	PCCWL	low
notPatrolling	AFL	low	AFL	low	AFL	low
	ATL	high	ATL	high	ATL	low
	RFL	low	RFL	low	RFL	high
	KGL	low	KGL	low	KGL	low
	PCWL	low	PCWL	low	PCWL	low
	PCCWL	low	PCCWL	low	PCCWL	low
PatrollingOnMyRight	AFL	low	AFL	low	AFL	low
	ATL	high	ATL	high	ATL	low
	RFL	low	RFL	low	RFL	high
	KGL	low	KGL	low	KGL	low
	PCWL	low	PCWL	low	PCWL	low
	PCCWL	low	PCCWL	low	PCCWL	low

Table 3.2: Supervisor Linguistic Rules in the case of a friend agent not in the visibility range and robot far from the border

DistanceFromFriend is high **and** DistanceFromBorder is medium						
DistanceFromNeighbor NeighborState	low		medium		high	
PatrollingOnMyLeft	AFL	low	AFL	low	AFL	low
	ATL	high	ATL	high	ATL	low
	RFL	low	RFL	low	RFL	low
	KGL	low	KGL	low	KGL	high
	PCWL	low	PCWL	medium	PCWL	low
	PCCWL	low	PCCWL	low	PCCWL	low
notPatrolling	AFL	low	AFL	low	AFL	low
	ATL	high	ATL	medium	ATL	low
	RFL	low	RFL	low	RFL	low
	KGL	low	KGL	high	KGL	high
	PCWL	low	PCWL	low	PCWL	low
	PCCWL	low	PCCWL	low	PCCWL	low
PatrollingOnMyRight	ATL	high	AFL	low	AFL	low
	RFL	low	ATL	high	ATL	low
	KGL	low	RFL	low	RFL	high
	PCWL	low	KGL	low	KGL	low
	PCCWL	low	PCWL	low	PCWL	low
			PCCWL	medium	PCCWL	low

Table 3.3: Supervisor Linguistic Rules in the case of a friend agent not in the visibility range and robot at medium distance from the border

DistanceFromFriend is high **and** DistanceFromBorder is low						
DistanceFromNeighbor NeighborState	low		medium		high	
PatrollingOnMyLeft	AFL	low	AFL	low	AFL	low
	ATL	high	ATL	low	ATL	low
	RFL	low	RFL	low	RFL	low
	KGL	low	KGL	low	KGL	high
	PCWL	low	PCWL	high	PCWL	low
	PCCWL	low	PCCWL	low	PCCWL	low
notPatrolling	AFL	low	AFL	low	AFL	low
	ATL	high	ATL	low	ATL	low
	RFL	low	RFL	low	RFL	low
	KGL	low	KGL	high	KGL	high
	PCWL	low	PCWL	low	PCWL	low
	PCCWL	low	PCCWL	low	PCCWL	low
PatrollingOnMyRight	AFL	low	AFL	low	AFL	low
	ATL	high	ATL	low	ATL	low
	RFL	low	RFL	low	RFL	low
	KGL	low	KGL	low	KGL	high
	PCWL	low	PCWL	low	PCWL	low
	PCCWL	low	PCCWL	high	PCCWL	low

Table 3.4: Supervisor Linguistic Rules in the case of a friend agent not in the visibility range and robot close to the border

DistanceFromBorder is *high* the robots should only try to reach the border avoiding other teammates, when the *DistanceFromBorder* is *medium* the robots can start to patrol the border clockwise or counterclockwise, but always avoiding other teammates; finally, when the *DistanceFromBorder* is *low*, the robots are performing the patrolling mission and do not allow robots approaching the border to influence their motion; this is helpful when a large number of vehicles are trying to reach the border; in this case, avoidance of other teammates patrolling the border, is achieved by activating the actions *Patrol CW* and/or *Patrol CCW*, depending on the values of the parameter γ_i and on the distance from the teammate. It can be noticed that in any case, for security reasons, if the distance between robots is *low*, the *Action Avoid Teammate* is activated.

3.7 Simulations

Several simulations on closed and open border, with different sizes and shapes, have been carried out by using both Matlab [48] and Player/Stage [114] environments. As both the supervisors present the same results, in the following no distinction will be made abou the FMS or Fuzzy supervisors.

3.7.1 Simulations in 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 area equal to 20 m. The matrix gains in eqs. (3.11)–(3.14) have been chosen as $\lambda_{rf} = 5\boldsymbol{I}$, $\lambda_{cw} = 8\boldsymbol{I}$, $\lambda_{ccw} = 8\boldsymbol{I}$, λ_{cw} and $\lambda_{ta} = 10\boldsymbol{I}$. No friends vehicles are present in this first simulation. Figure 3.14 shows the robot positions at different time instants. Robots are represented by green points surrounded by their visibility ranges and safety areas; the continuous blue line represents the border to be patrolled. In the first frame, all robots are trying to reach the border, while avoiding teammates. In the second and third frames, some robots reach the border and start patrolling. In the fourth frame the robots reach a maximum density on the border without being affected by other vehicles trying to reach the border; at the same time, by properly selecting the actions *Action Patrol CW* and *Action Patrol CCW*), they keep themselves at the safe distance from other robots on the line.

It is useful to remark that the approach prevents robots from collisions. To this aim, in Figure 3.15 the time history of the minimum value of the distances among all the possible

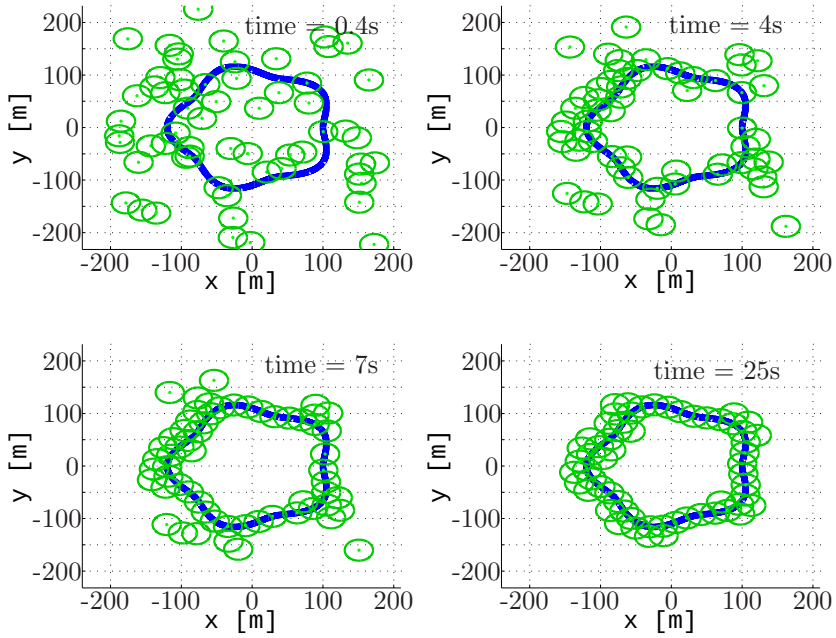

Figure 3.14: Sample frames at different time instants of robots performing a patrolling mission.

robot couples is depicted, i.e.:

$$d_{r,min}(t) = \min_{\forall i \neq j \ i,j \in Nr} \| \, p_{i,r}(t) - p_{j,r}(t) \, \|, \tag{3.24}$$

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

As it can be seen from Figure 3.15, even in the case of a large number of robots, collisions are prevented, moreover the structure of the supervisor (both the state automata and fuzzy logic) avoid interference situations. In fact, the fourth frame in Figure 3.14 shows that robots reaching the border do not affecting robots already on the line (i.e., performing the patrolling mission). On the contrary, in the case of interferences, increasing the number of robots would not cause any benefit to the overall mission, but would, in certain cases, cause the mission failure.

3.7.2 Simulations in presence of friend vehicles and a large number of faults

In the following, Matlab simulations are briefly described so as to show the algorithm behavior in the presence of large number of robot faults and presence of friend agents [69].

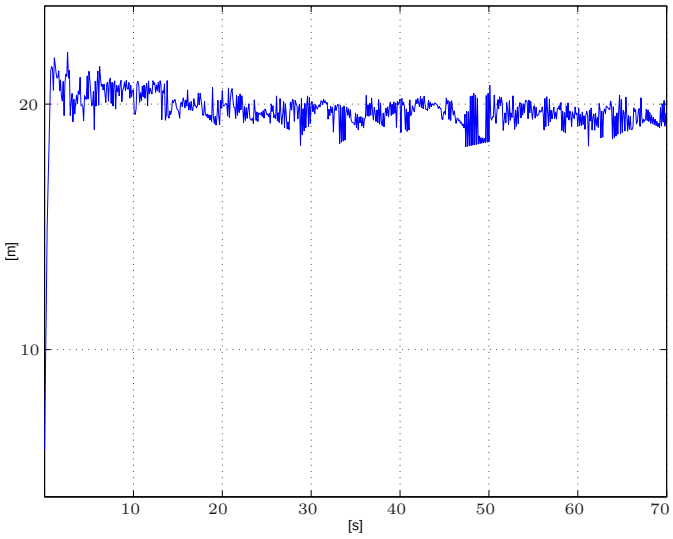

Figure 3.15: Time history of minimum of distances between all possible robot couples.

Videos of simulations are available at [3]. Also in this case, the team is composed by 60 robots, with a visibility and safety area equal to 20 m. The scalar gains are the same as in the previous case. Figure 3.16 shows the robot positions at different time instants. Robots are represented by points surrounded by their visibility range and safety area (a continuous green circle); while the continuous thick line represents the border to be patrolled. As in the previous case, in the first frame, robots are randomly distributed and they are trying to reach the border activating the *Action Reach Frontier*, while avoiding other teammates. In the second frame, some robots have reached the border and started to perform the patrolling mission activating the *Actions Keep Going* and *Patrol Clockwise* (or *Patrol Counterclockwise*). Friend agents, represented by cross markers surrounded by their safety area (a dash-dot circle of 20m radius), start to approach the border and cross it (third frame) without collisions with patrolling robots. Then, in the fourth frame friends gain the center of the bordered zone; in this situation, patrolling robots are not more affected by friends motions.

As can be noticed, the effective definition of the behaviors and actions allows collisions avoidance even in case of high robot density and friend agents. To this aim, in Figure 3.17 the minimum value over time of the distances among all the possible robot couples is depicted, i.e.:

$$d_{r,min}(t) = \min_{\forall i \neq j \ i,j \in Nr} \| \ p_{i,r}(t) - p_{j,r}(t) \ \|, \tag{3.25}$$

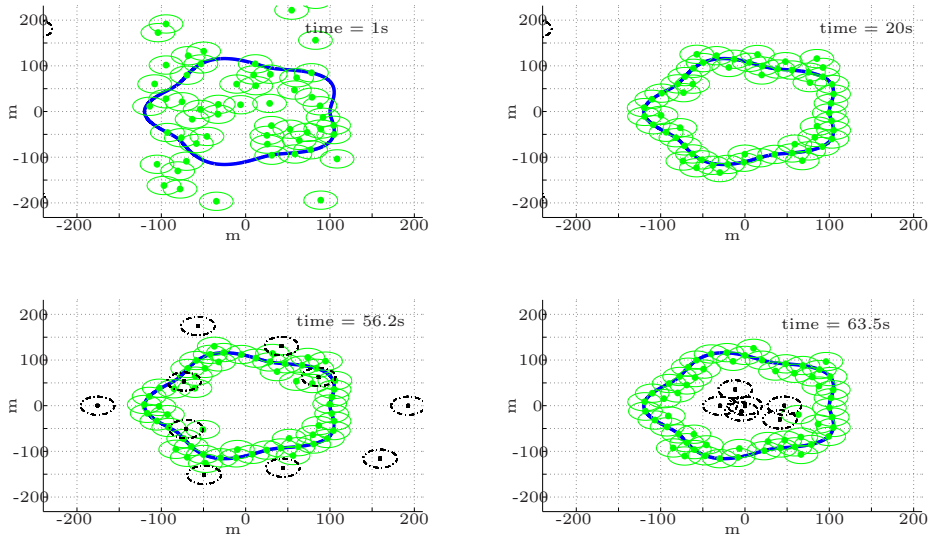

Figure 3.16: Sample frames at different time instants of robots performing a patrolling mission.

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

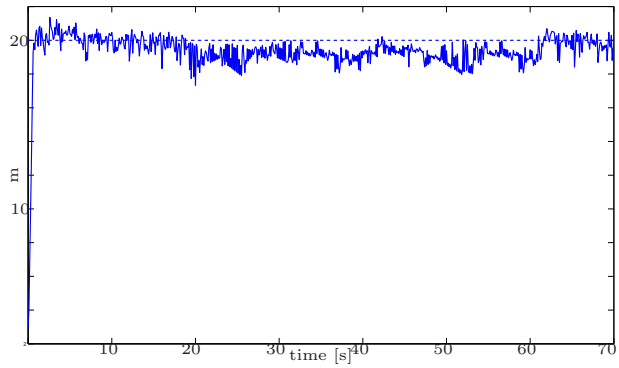

Figure 3.17: Minimum of distances between all possible robot couples.

In the same way, in Figure 3.18 it is shown the minimum distance over time of all the possible couples robot-friend, i.e.:

$$d_{f,min}(t) = \min_{\forall i \in N_r, j \in N_f} \parallel p_{i,r}(t) - p_{j,f}(t) \parallel, \tag{3.26}$$

where $p_{j,f}(t)$ is the position of j-th friend agent at instant t and N_f is the set of friend agents.

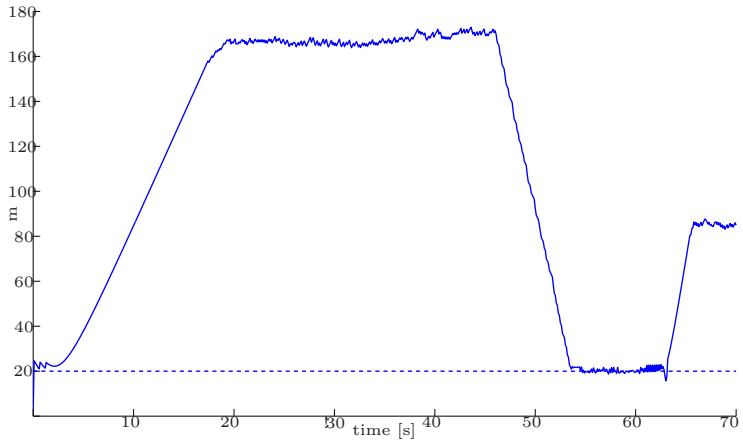

Figure 3.18: Minimum of distances between all robot-friend couples.

Figures 3.17 and 3.18 clearly show that the control is capable of avoiding collisions, and thus the mission is safely performed.

Finally, the performance in the presence of multiple robot faults are tested. Namely, the same patrolling mission is executed, but some of the patrolling robots fail (first frame of Figure 3.19). These failures are virtually represented by robot disappearances from the scene. The last frame shows the new situation, where the robots, without any external influence, redistribute along the border, thus fulfilling the mission objective.

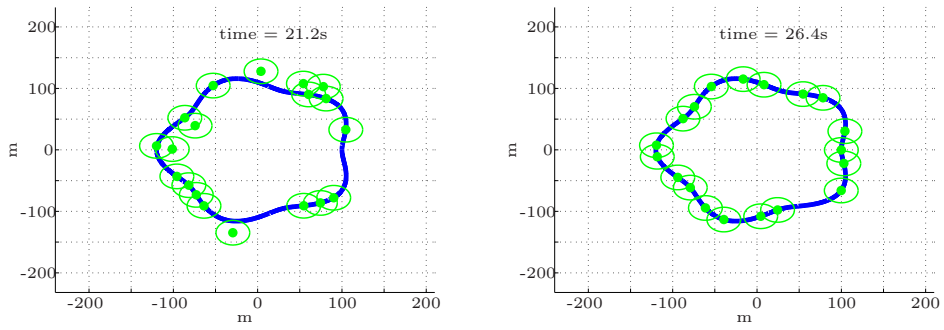

Figure 3.19: In left plot several robots fails. In the right plot remaining robots automatically redistribute around the border.

3.8 Experiments

In order to test the effectiveness of the approach, the control scheme described in Section 3.6 has been implemented on a real setup at the Distrubuted Intelligence Laboratory

of University of Tennessee, composed by three Pioneer-2DX mobile robots (in Figure 3.20).

3.8.1 The Pioneer 2DX robot

The Pioneer-2DX robot, in Figure 3.20 (left), has a 44 cm x 38 cm x 22 cm aluminum body with 16.5cm diameter drive wheels and autonomy of 18-24 hours; it is manufactured by ActivMedia Robotics [4].
Pioneer 2DX is differentially steered; the two motors use 38.3:1 gear ratios and contain 500-ticks encoders. This differential drive platform is non-holonomic and can rotate in place moving both wheels, or it can swing around a stationery wheel in a circle of 32cm radius. A rear caster balances the robot.
On flat floor, the Pioneer-2DX can move at speeds of 16 cm/s. At slower speeds it can carry payloads up to 23kg, including additional batteries and all accessories, and must be balanced appropriately for effective operation of the robot. In addition to motor encoders, the robot base includes eight ultrasonic transducers (range-finding sonar) arranged to provide 180-degree forward coverage at a sampling rate of 25Hz. They can read ranges from 15cm to approximately 7m.
The computational board includes a 32-bit RISC-based controller running Linux. On the microcontroller, there are 8 digital input and 8 digital output ports and 1 dedicated A/D port; 4 digital input ports can be reconfigured as A/D input and 4 digital output ports can be reconfigured as PWM outputs.

Additional hardware includes ethernet based communications, a wifi radio turret for remote control, a Pan-Tilt-Zoom (ptz) color camera and a laser rangefinder. Laser rangefinders contain a non-contact laser measurement system used for applications involving objects measurement, localization, monitoring, vehicle guidance and collision control. In particular, the laser rangefinder mounted on the Pioneer-2DX robots is a SICK LMS 200, characterized by a 180 degrees coverage, a 10 mm resolution, 0.25 degrees minimum angular resolution, a scanning frequency up to 75Hz, data transfer in real time, and, if properly configured, it is capable of a range up to 80 m.

3.8.2 Robot Model

In Figure 3.21, the kinematic structure of the Pioneer 2DX has been reported . Each individual wheel contributes to the robot motion and, at the same time, imposes constraints on it. Thus, the constraints of each wheel combine to form the constraints of the overall

Figure 3.20: The experimental setup at Distributed Intelligence Laboratory of University of Tennessee. On the left: a zoom of a Pioneer-2Dx mobile robot. On the right: the whole setup.

motion of the robot.

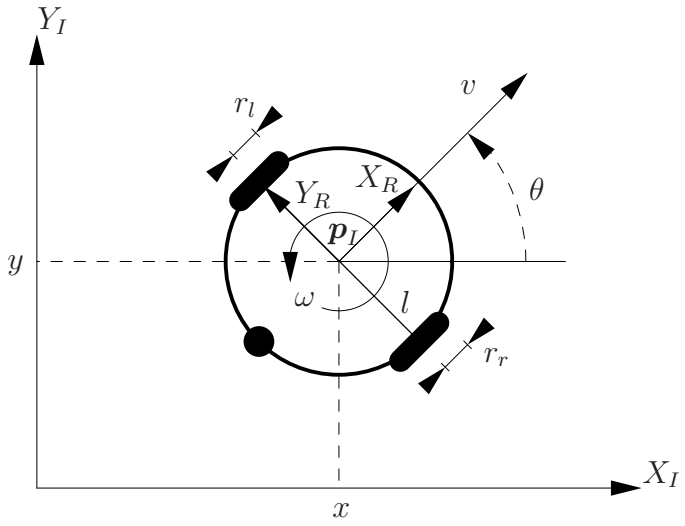

Figure 3.21: Top-view sketch of a differential-drive mobile robot with relevant variables.

To model the motion of the mobile robot, the constraints and the forces of each wheel must be expressed respect to a proper and consistent reference frame. Thus, to specify the robot position, it is possible to consider the robot as a rigid body on wheels operating on an horizontal plane; it is characterized by three Degrees Of Freedom (DOF) (two for the horizontal position in the plane and one for the orientation around a vertical axis). Then, the position of the robot can be specified by the relationship between a global

inertial reference frame and a body fixed reference frame. In Figure 3.21 the axes X_I and Y_I define the inertial reference frame, while the axes X_R and Y_R define the body-fixed reference frame. The position of the robot is specified by the coordinates x and y of the point $\boldsymbol{p}_I p$ in the $\{ O, X_I , Y_I\}$ reference frame and by the angle θ between the axes X_I and X_R

$$\boldsymbol{p}_I = \begin{bmatrix} x \\ y \\ \theta \end{bmatrix}.$$

It can be easily recognized, that the differential-drive robot is a non-holonomic system characterized by the non-integrable constraint:

$$\dot{x}\sin(\theta) - \dot{y}\cos(\theta) = \begin{bmatrix} \sin(\theta) & \cos(\theta) & 0 \end{bmatrix} \begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \boldsymbol{A}(\boldsymbol{p})\dot{\boldsymbol{p}} = 0, \tag{3.27}$$

stating that velocity components along the wheels' axis are not feasible. Feasible velocities can be obtained as a linear combination of a basis of the null space of matrix $\boldsymbol{A}(\boldsymbol{p})$:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) \\ \sin(\theta) \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} v \\ \\ \omega \end{bmatrix}, \tag{3.28}$$

where v and ω are, respectively, the linear and angular velocity of the robot. In case of differential-drive robots, assuming that the two wheels roll without slipping on the floor, linear and angular velocities are related to the wheels velocities in the following way:

$$\begin{cases} v &= (\omega_r r + \omega_l r)/2, \\ \omega &= (\omega_r r - \omega_l r)/l, \end{cases}$$

where ω_l, ω_r are the angular velocities of the left and right wheels, respectively, r is the wheels radius and l is the distance between the wheels.

It is worth noticing that the two-wheel differential-drive robot has a kinematic structure analogous to the unicycle, i.e., an homogeneous disk that rolls without slipping on an horizontal plane (see Figure 3.22). The unicycle, in fact, has a kinematic model described by the equation (3.28). The control of the unicycle poses several challenges. In fact, two main tasks are generally required: in point-to-point motion, a desired goal conguration must be reached starting from a given initial conguration, while, in trajectory following,

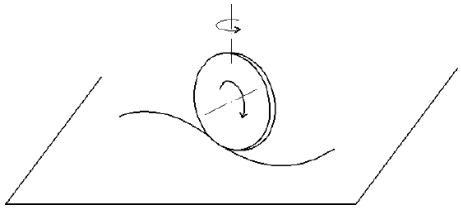

Figure 3.22: Vertically rolling disk.

a reference point on the robot must follow a trajectory in the Cartesian space (i.e., a geometric path with an associated timing law) starting from a given initial conguration. The execution of these tasks can be achieved using either feedforward or feedback control (or a combination of the two); obviously, the latter has to be preferred in view of its intrinsic degree of robustness. When executed under a feedback strategy, the point-to-point motion task leads to a regulation control problem. Instead, trajectory following leads naturally to a tracking problem, which may be asymptotic in the presence of an initial error. Remarkably, such systems cannot be stabilized at a point by smooth feedback [39]. Then, as a consequence, tracking is easier than regulation for a nonholonomic vehicle. However, several alternative approaches have been proposed for regulation of nonholonomic systems [97], [55], [89].

Since our aim is not to solve the control problem for non-holonomic systems, a different and simpler solution needs to be found. To this aim, it is worth considering that we are not interested in assigning the orientation θ of the robot, then, the system to be controlled becomes

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\theta} \end{bmatrix} = \begin{bmatrix} \cos(\theta) & 0 \\ \sin(\theta) & 0 \end{bmatrix} \begin{bmatrix} v \\ \\ \omega \end{bmatrix} = \boldsymbol{G}(\theta) \begin{bmatrix} v \\ \\ \omega \end{bmatrix}. \tag{3.29}$$

Matrix $\boldsymbol{G}(\theta)$ is clearly singular for any value of θ; hence, not all velocity commands generated by the actions have corresponding robot's linear and angular velocity. A simple solution to the above problem consists in considering a different robot reference frame that is not on the wheels' axis (see Figure 3.23).

By choosing as origin of the new reference frame a point $\boldsymbol{p}_b$, whose distance is b from the wheels' axis, the kinematic model of the robot becomes:

$$\dot{\boldsymbol{p}}_b = \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -b\sin(\theta) \\ \sin(\theta) & +b\cos(\theta) \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} = \boldsymbol{G}_b(\theta) \begin{bmatrix} v \\ \omega \end{bmatrix}, \tag{3.30}$$

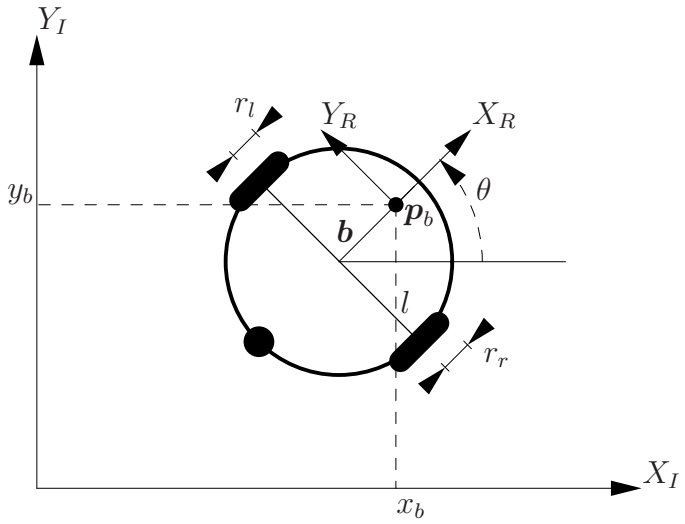

Figure 3.23: Top-view sketch of a differential-drive mobile robot with relevant variables.

in this way $\boldsymbol{G}_b(\theta)$ is a full-rank matrix independently on the the value of θ. Clearly, small value of b can result in an high value of ω, while high value imply a control point $\boldsymbol{p}_b$ too far from the robot's center of mass. Taking into account the velocities generated by the control algorithm, the dimensions and the maximum linear and angular velocities of the Pioneer 2DX robots, b has been chosen equal to $10cm$. This value prevents from saturation of the actuators and ensure a control point inside the robot chassis.

3.8.3 The Control Software

The software used to implement the proposed architecture is Player/Stage environment [114], based on a client/server architecture.

Player/Stage is an advanced robotics simulation and interface platform. It provides both the tools to simulate (Stage software), as well as to communicate to the robotics harware (player software). Player is mainly a network *server* for robot control, and communicates with on board hardware devices by using device drivers. In addition, it provides clean and simple interfaces (proxies) to the robot's sensors and actuators (see Figure 3.24). Some proxies and drivers provide complex functionalities, such as local path planning and vision, and provides to its clients standard device interfaces. For example, Player may use the SICK LMS-200 and Pioneer hardware devices, and provide to *clients* "laser" and "position", "sonar", general interfaces. This allows clients to be portable to other robots.

Player also provides transport mechanisms allowing data to be exchanged among drivers and control programs that are executing on different machines. The most common mechanism in use now is a client/server TCP socket-based transport. The drivers run inside the player server, and the user's control program runs often on an other machine as a client to that server. Client Libraries are available in various languages to facilitate the development of such control programs(C, C++, Java).

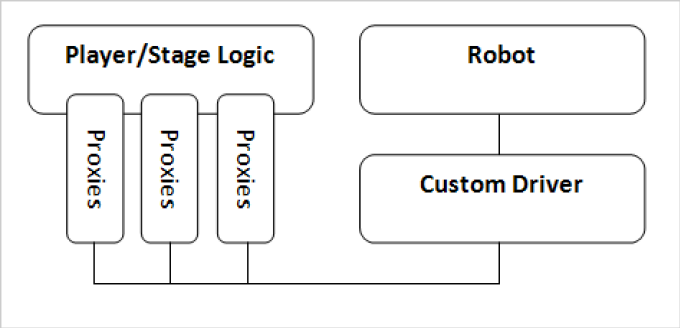

Figure 3.24: Structure of Player control sofwtare.

Stage software allows to test the control algorithms in simulation before running them on a real setup. Stage is a 2D multiple-robot simulator that simulates a population of mobile robots, sensors and objects in a two-dimensional bitmapped environment. It provides fairly simple and computationally cheap models of lots of devices, rather than attempting to reproduce any device with great fidelity. Typically, Stage is designed in such a way that the controll software designed in Player cannot *see* the difference between the real robot devices and their simulated Stage equivalents. Generally, very little modifications are needed to transform simulated controllers to the their experimental counterparts.

3.8.4 Robot Localization

As stated in Section 3.5.1, the robot needs to localize in the environment, in order to achieve the mission. To achieve such a goal, a localization algorithm needs to be adopted, provided that a map of the environment is available. Among the different approaches available in the literature, one of the most effective has been developed by Fox *et al.* by using particle filters [45].

In the used setup, in fact, localization is achieved by means of odometry and laser range readings combined in a particle filter algorithm. Although an extensive literature exists about this kind of algorithms, a brief description is given in the following. Even the most

straightforward implementation of particle filters exhibits excellent results for the position tracking and the global localization problem. Extensions of the basic algorithm have led to excellent results on the kidnapped robot and the multi-robot localization problem.
The power of particle filters is that, in contrast to the widely used Kalman filters, they can approximate a large range of probability distributions, not just normal distributions. Once a robot's belief is focused on a subspace of the space of all poses, particle filters are computationally efficient, since they focus their resources on regions in state space with high likelihood. At the basis of particle filters, there are the Bayes filters. They assume that the environment is Markovian, i.e., past and future data are conditionally independent if the current state is known. In particular, the key idea is to estimate the posterior probability density over the state space conditioned on the data. In robotics this posterior density is typically called the *belief* (*Bel*). Two types of data are distinguished: perceptual data, such as laser range measurements, and odometry data, or controls, which take into account information about robot motion. Denoting the former by $\boldsymbol{y}$ and the latter by $\boldsymbol{u}$, the belief on a generic state $\boldsymbol{x}_k \in \mathbb{R}^n$, at sample k, is:

$$Bel(\boldsymbol{x}_k) = p(\boldsymbol{x}_k|\boldsymbol{y}_k, \boldsymbol{u}_{k-1}, \boldsymbol{y}_{k-1}, \cdots, \boldsymbol{u}_0, \boldsymbol{y}_0), \tag{3.31}$$

where $p(\cdot)$ is the probability density operator. In the localization problem, $\boldsymbol{x}_k$ contains the robot's position, hence, equation (3.31) states that the propability of the robot's position depends on the time history of perceptual and control data. Pratically, Bayes filters estimate the belief recursively, and the initial belief characterizes the initial knowledge about the system state. Observing that (3.31) can be rewritten as

$$Bel(\boldsymbol{x}_k) = \frac{p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{u}_{k-1}, \cdots, \boldsymbol{u}_0, \boldsymbol{y}_0)p(\boldsymbol{x}_k|\boldsymbol{u}_{k-1}, \cdots, \boldsymbol{u}_0, \boldsymbol{y}_0)}{p(\boldsymbol{y}_k|\boldsymbol{u}_{k-1}, \cdots, \boldsymbol{u}_0, \boldsymbol{y}_0)}, \tag{3.32}$$

using the Markov assumption $p(\boldsymbol{y}_k|\boldsymbol{x}_k, \boldsymbol{u}_{k-1}, \cdots, \boldsymbol{u}_0, \boldsymbol{y}_0) = p(\boldsymbol{y}_k|\boldsymbol{x}_k)$ (this is the sensor model) and posing $\eta = 1/p(\boldsymbol{y}_k|\boldsymbol{u}_{k-1}, \cdots, \boldsymbol{u}_0, \boldsymbol{y}_0)$ (normalizing factor), equation (3.31) is equivalent to

$$Bel(\boldsymbol{x}_k) = \eta\, p(\boldsymbol{y}_k|\boldsymbol{x}_k) \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \cdots, \boldsymbol{y}_0)p(\boldsymbol{x}_{k-1}|\boldsymbol{u}_{k-1}, \cdots, \boldsymbol{y}_0)d\boldsymbol{x}_{k-1}. \tag{3.33}$$

Finally, since for the Markov assumption is $p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1}, \cdots, \boldsymbol{y}_0) = p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1})$ (vehicle model), equation (3.33) becomes:

$$Bel(\boldsymbol{x}_k) = \eta\, p(\boldsymbol{y}_k|\boldsymbol{x}_k) \int p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}, \boldsymbol{u}_{k-1})Bel(\boldsymbol{x}_{k-1})d\boldsymbol{x}_{k-1}, \tag{3.34}$$

that, clearly, has a recursive structure.
As the state $\boldsymbol{x}$ is generally continuous, the idea of particle filters algorithms is to approximate the belief $Bel(\boldsymbol{x})$ by a set of n samples, called *particles*:

$$Bel(\boldsymbol{x}_k) \approx \sum_{i=1}^{n} f_{k-1}^i \delta(\boldsymbol{x}_k - \boldsymbol{x}_{k-1}^i), \tag{3.35}$$

where δ is the Dirac function, $\boldsymbol{x}^i$ is a particular sample (of the state) and f^i is a positive number named *importance factor* that take into account the importance of each sample ($\sum_{i=1}^{n} f^i = 1$). Two problems need to be solved: the generation of the particles, and the update of the importance factors. Different approaches exist to solve these problems, one of them consists in the following three steps to be recursively exectuted [12]:

- uniformly extract n samples $\boldsymbol{x}_k^i$ from the distribution $\sum_{i=1}^{n} f_{k-1}^i p(\boldsymbol{x}_k|\boldsymbol{x}_{k-1}^i)$,
- set the importance factor as $f_k^i = p(\boldsymbol{y}_k|\boldsymbol{x}_k^i)$,
- normalize importance factors as: $f_k^i = f_k^i / \sum_{i=1}^{n} f_k^i$

The most important property of the particle filter is its ability to handle complex, multimodal (non-Gaussian) posterior distributions. However, it has difficulties when $\boldsymbol{x}$ is high-dimensional. Namely, the number of particles n required to adequately approximate the distribution grows exponentially with the dimensionality of the state space. This poses difficulties in applications such as articulated human body tracking or SLAM.

After having described the experimental setup, experiments to validated the designed architecture in an indoor environment will be shown. Videos of the experiments are available at [1], [2]. Figure 3.25 shows a portion of the patrolled area and its representation in Player/Stage. The blue line represents the border, the red, green and cyan polygons represent the robots together with their visibility range. In particular the border is a closed line composed by segments joined by arcs; and its overall length is 51 m. The robots know the exact description of the border and they approach it at a speed of 0.5 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 ($\lambda_{ta} = 0.3$). The localization in the environment is achieved by a pre-built map and the localization driver based on an adaptive particle filter ([45]). Visibility range and safety area are equal to 2.5 m, the threshold value for the state transitions described in Section 3.6.5 is 0.6 m. Moreover, if a robot is in the patrolling state (see Sect. 3.6.5), every 27 s it can decide to invert its motion direction, according to a random variable.
Figure 3.25 shows a portion of the patrolled area and its Player/Stage representation.

Figure 3.25: On the left a portion of the environment. On the right its representation obtained by Player/Stage software, the path and the three patrolling robots.

Figure 3.26 shows the distance from the border for the three robots. The distances are within 0.1 m, this value it's acceptable for the experimental conditions and requirements. Moreover, peaks are reached during rotation movements due to sensor noise and, above all, to the neglected robot dynamics in the control law.

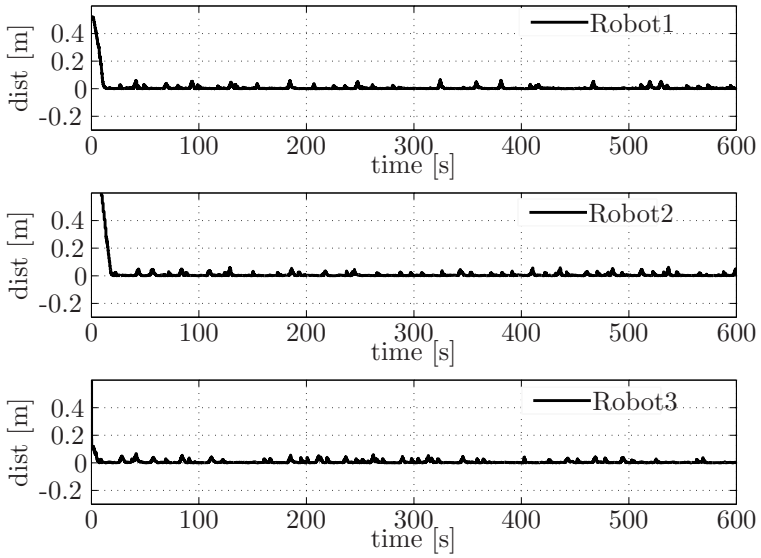

Figure 3.26: Distance from the border.

Figure 3.27 shows the time histories of the robots action selection. In their initial positions, robots 1 and 2 are closer than the safety range, in this way both of them will activate the *Avoid Teammate Action*. After a few seconds, all robots enter in the patrolling state. While patrolling, robots are usually in the state *KeepGoing* (*CW* or *CCW*). Spikes in this graph correspond to motion direction changes (when two robots encounter each other), where a sudden transition from *KeepGoing* to *PatrolCW* for one robot always

comes with a sudden transition to *PatrolCCW* for the other robot. After the interaction they go back to the *Keep Going* state proceeding in opposite directions. A sequence of the movements occurring in this situation is shown in Figure 3.28.

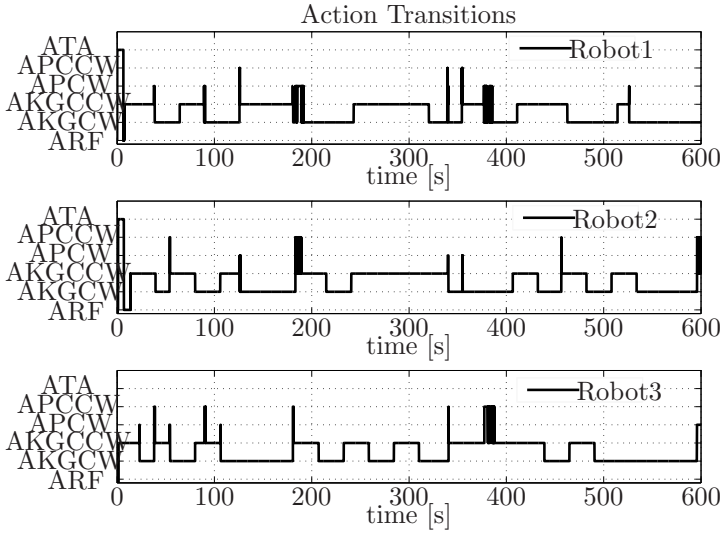

Figure 3.27: Time history of actions selection. ARF: Action Reach Frontier. AKGCW: Action Keep-Going (CW). AKGCCW: Action Keep-Going (CCW). APCW: Actions Patrol CW. APCCW: Action Patrol CCW. ATA: Action Avoid Teammate.

Finally, it is useful to show how the distance between two consecutive robots varies over time during the patrolling mission (Figure 3.29). According to the deterministic optimal policy in [32], in every time instant, the distance between two consecutive robots should be, in our case, one third of the overall border length; also, all robots should move in the same direction. On the contrary, according to [5], robots should move synchronously but in a nondeterministic way, in order to maximize the probability of intercepting an intruder. Both the cases require a centralized supervisor or information exchanging between robots, so it is straightforward to imagine that the approach proposed can have better performance only increasing the number of vehicles or removing some constraints.

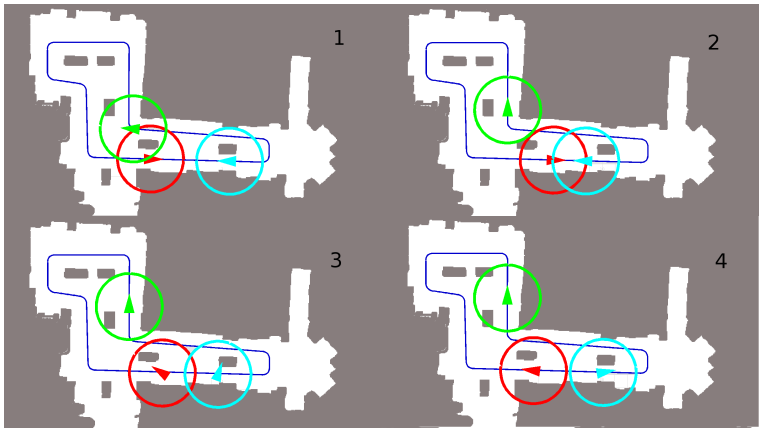

Figure 3.28: Three robots team performing the patrol mission. The lower ones (in red and cyan) meet along the path and invert their motion directions. The arrows represent forward motion direction.

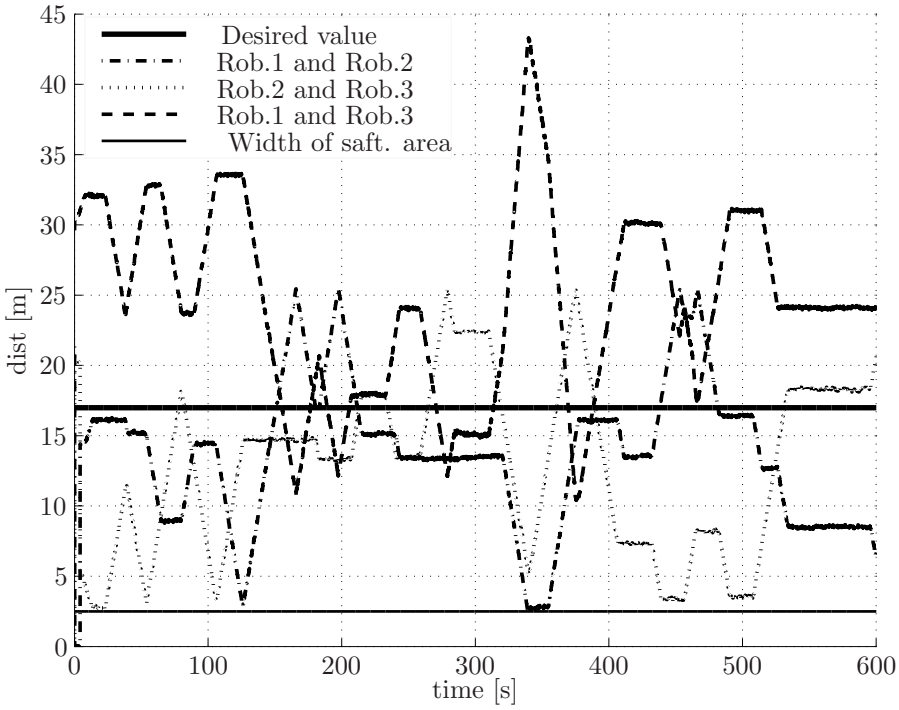

Figure 3.29: Distance between two consecutive robots.

Conclusions and future work

The research described in this thesis has concerned the development of a high level decentralized architecture for the control of multi-robot systems.
The architecture has been developed in the framework of the Null-Space-Behavioral approach, a behavior-based approach to control generic multi-robot systems. In particular, the NSB has been introduced in comparison with the main behavioral approaches known in literature (the layered control system as a significant case of competitive approach and the motor schema control as a significant case of cooperative approach) to underline its novelty and advantages. The NSB, in fact, differs from the other approaches in the way it composes the single task outputs to build the motion command to the robots, and, properly managing the tasks organized in a hierarchical structure, it allows to simultaneously achieve multiple, even-conflicting, tasks.
The proposed architecture has been developed in the framework of the swarm robotics and, among the several assumptions, no communication is allowed among robots, this means that each robot acts only according its sensing capabilities. At the basis of the developed architecture, there is the concept of action, obtained by combining in the NSB framework simple behaviors. These behaviors are elementary motion command not yet suitable for the given mission, while actions are high level commands allowing the fulfilment of given tasks. Once defined the set of actions, it is mandatory to design a Supervisor that chooses the next action to activate. Although several paradigms exist in literature, two supervisor have been developed: a Finite State Automata Supervisor and a Fuzzy Logic Supervisor.
The overall architecture has been applied to the multi-robot patrolling mission. In this mission, a group of robots is in charge of surveillance of a given area avoiding collisions and interference situations. The approach has been tested in simulation in the case of large number of robots, typical in swarm robotics, and experimentally at the Distributed Intelligence Laboratory of University of Tennessee on a setup composed by three Pioneer 2DX robots. The approach hes been shown to be effective and robust to sensor noise

and robot failures. However, several extensions and improvements can be taken into consideration. First, other types of Supervisor can be designed, in particular an emotional supervisor based on the Alliance architecture developed by Parker can be adapted to the action selection case. The use of such a supervisor, can be useful to deal with highly dynamic environments, where more behaviors and actions could be required making difficult to code the action selection mechanism by means of a state finite automata or a fuzzy logic engines. Moreover, the developed architecture can be tested in different missions like coverage, flocking, etc., properly redefining the behavior and action sets. It would be also interesting to test the approach in the case of unmanned underwater vehicles (UUVs), unmanned aerial vehicles (UAVs) and in heterogeneous team of robots. Finally, introducing some form of stigmercic communication, will be possible to introduce some performance indexes and compare the approach with existing decentralized solutions.

Bibliography

[1] http://www.cs.utk.edu/dilab/movies/Marino-Video1c.avi.

[2] http://www.cs.utk.edu/dilab/movies/Marino-Video2c.avi.

[3] http://www.cs.utk.edu/dilab/movies/Videosimulation.avi.

[4] `www.activrobots.com/`. K-Team.

[5] N. Agmon, S. Kraus, and G. A. Kaminka. Multi-robot perimeter patrol in adversarial settings. In *Proceedings 2008 IEEE International Conference on Robotics and Automation*, pages 2339–2345, Pasaena, CA, May 2008.

[6] A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, and V. Corruble. Recent advances on multi-agent patrolling. *Proceedings of the Brazilian Symposium on Artificial Intelligence*, 2004.

[7] G. Antonelli. Stability analysis for prioritized closed-loop inverse kinematic algorithms for redundant robotic systems. *IEEE Transactions on Robotics*, 25(5):985–994, October 2009.

[8] G. Antonelli, F. Arrichiello, and S. Chiaverini. Experimental kinematic comparison of behavioral approaches for mobile robots. In *Proceedings 16th IFAC World Congress*, Prague, CZ, July 2005.

[9] G. Antonelli, F. Arrichiello, and S. Chiaverini. Experiments of formation control with multirobot systems using the null-space-based behavioral control. *IEEE Transactions on Control Systems Technology*, 17(5):1173–1182, Sept. 2009.

[10] R.C. Arkin. Motor schema based mobile robot navigation. *The International Journal of Robotics Research*, 8(4):92–112, 1989.

[11] F. Arrichiello. *Coordination Control of Multiple Mobile Robots*. PhD thesis, Università degli Studi di Cassino, Italy, 2007.

[12] M. S. Arulampalam, S. Maskell, and N. Gordon. A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. *IEEE Transactions on Signal Processing*, 50:174–188, 2002.

[13] J. Baillieul. Kinematic programming alternatives for redundant manipulators. In *Proceedings of the 1995 International Conference Proc. IEEE International Conference on Robotics and Automation*, pages 722–728, 1985.

[14] T. Balch and R. C. Arkin. Communication in reactive multiagent robotic systems. *Autonomous Robots*, 1:27–52, 1994.

[15] G. Baldassarre, V. Trianni, M. Bonani, F. Mondada, M. Dorigo, and S. Nolfi. Self-organized coordinated motion in groups of physically connected robots. *IEEE Transactions on Systems, Man, and Cybernetics, Part B*, 37(1):224–239, 2007.

[16] A. S. Barry and J. Czechanski. Ground surveillance radar for perimeter intrusion detection. *Proceedings of the Digital Avionics Systems Conference*, 7-13, October 2000.

[17] R.W. Beard, J. Lawton, and F.Y. Hadaegh. A coordination architecture for spacecraft formation control. *IEEE Transactions on Control Systems Technology*, 9(6):777–790, 2001.

[18] M. Benda, V. Jagannathan, and R. Dodhiawalla. On optimal cooperation of knowledge sources. Technical report, August 1985.

[19] A. L. Bertozzi, M. Kemp, and D. Marthaler. Determining environmental boundaries: Asynchronous communication and physical scales. volume 309, pages 25–42. 2004.

[20] C. G.. Lo Bianco, A. Piazzi, and M. Romano. Smooth motion generation for unicycle mobile robots via dynamic path inversion. *IEEE Transactions on Robotics*, 20, October 2004.

[21] B.E. Bishop. On the use of redundant manipulator techniques for control of platoons of cooperating robotic vehicles. *IEEE Transactions on Systems, Man and Cybernetics*, 33(5):608–615, Sept. 2003.

[22] R. Brooks. A robust layered control system for a mobile robot. 2(1):14–23, 1986.

[23] Joanna J. Bryson. Action selection and individuation in agent based modelling. In *Proceedings of Agent 2003: Challenges of Social Simulation*, pages 317–330, 2003.

[24] Y. Cao, A.S. Fukunaga, and A.B. Kanhg. Cooperative mobile robotics: Antecedents and directions. *Autonomous Robots*, 4:7–27, 1997.

[25] S. Carpin and L. E. Parker. Cooperative motion coordination amidst dynamic obstacles. In *Distributed Autonomous Robotic Systems*, pages 145–154, 2002.

[26] J. Carvalho, E. Paiva, J. Ramos, A. Elfes, M. Bergerman, and S. Bueno. Air-ground robotic ensembles for cooperative applications: concepts and preliminary results. In *Proceedings of the International Conference on Field and Service Robotics*, pages 75–80, 1999.

[27] D. W. Casbeer, D. B. Kingston, A. W. Beard, T. W. Mclain, S. Li, and R. Mehra. Cooperative forest fire surveillance using a team of small unmanned air vehicles. *International Journal of Systems Sciences*, 37:360, 2006.

[28] A. Castano, R. Chokkalingam, and P. Will. Autonomous and self-sufficient CONRO modules for reconfigurable robots. In *Proceedings of 5th International Symposium on Distributed Autonomous RoboticSystems*, pages 155–164, Knoxville, TN, 2000. Springer-Verlag.

[29] L. Chaimowicz, B. Grocholsky, J. F. Keller, V. K., and C. J. Taylor. Experiments in multirobot air-ground coordination. In *Proceedings of the 2004 International Conference on Robotics and Automation*, pages 4053–4058, 2004.

[30] K. Chang, D. Ruspini, R. Holmberg, A. Casal, A. Baader, O. Khatib, and K. Yokoi. Force strategies for cooperative tasks in multiple mobile manipulation systems, 1996.

[31] V. Chankong and Y. Y. Haimes. *Multiobjective Decision Making Theory and Methodology*. Elsevier Science, New York, 1983.

[32] Y. Chevaleyre. Theoretical analysis of the multi-agent patrolling problem. *Procedings of the IEEE/WIC/ACM International Conference on Intelligent Agent Technology*, pages 302–308, 2004.

[33] Y. Chevaleyre, F. Sempé, and G. Ramalho. A theoretical analysis of multi-agent patrolling strategies. In *Autonomous Agents and Multi-Agent Systems*, pages 1524–1525, 2004.

[34] P. Chiacchio, S. Chiaverini, L. Sciavicco, and B. Siciliano. Closed-loop inverse kinematics schemes for constrained redundant manipulators with task space augmentation and task priority strategy. *The International Journal Robotics Research*, 10(4):410–425, 1991.

[35] S. Chiaverini. Singularity-robust task-priority redundancy resolution for real-time kinematic control of robot manipulators. *IEEE Transactions on Robotics and Automation*, 13(3):398–410, 1997.

[36] I. C. Christopher. Applications of harmonic functions to robotics. *Journal of Robotic Systems*, 10:931–946, 1992.

[37] J. Clarka and R. Fierro. Mobile robotic sensors for perimeter detection and tracking. *ISA Transaction*, 28:3–13, 2007.

[38] A.K. Das, R. Fierro, V. Kumar, J.P. Ostrowski, J. Spletzer, and C.J. Taylor. A vision-based formation control framework. *IEEE Transactions on Robotics and Automation*, 18(5):813–825, 2002.

[39] A. De Luca and G. Oriolo. *Modelling and control of nonholonomic mechanical systems*, volume 360 of *CISM Courses and Lectures*, pages 277–342. Springer, 1995.

[40] D. Driankov, H. Hellendoorn, and M. Reinfrank. *An Introduction to Fuzzy Control*. Springer-Verlag, Berlin, D, 1995.

[41] G. Dudek, M.R.M. Jenkin, E. Milios, and D. Wilkes. A taxonomy for multi-agent robotics. *Autonomous Robots*, 3(4):375–397, 1996.

[42] H. R. Everett, R. T. Laird, G. Gilbreath, T. A. Heath-pastore, R. S. Inderieden, K. Grant, and D. M. Jaffee. Multiple resource host architecture for the mobile detection assessment and response system. In *Technical Document 3026, Space and Naval Warfare Systems*, 1998.

[43] J.T. Feddema, C. Lewis, and D.A. Schoenwald. Decentralized control of cooperative robotic vehicles: theory and application. *IEEE Transactions on Robotics and Automation*, 18(5):852–864, 2002.

[44] J. W. Fenwick, P. M. Newman, and J. J. Leonard. Cooperative concurrent mapping and localization. In *Proocedings of the 2002 IEEE International Conference on Robotics and Automation*, pages 1810–1817, Washington, DC, USA, 2002.

[45] D. Fox, W. Burgard, F. Dellaert, and S. Thrun. Monte carlo localization: Efficient position estimation for mobile robots. In *Proc. of the National Conference on Artificial Intelligence*, 1999.

[46] M. Gardner. The fantastic combinations of john conway's new solitaire gamelife. *Scientific American*, 223:120–123, October 1970.

[47] J. Hertzberg and R. Chatila. *Springer Handbook of Robotics*, chapter AI Reasoning Methods for Robotics, pages 207–223. B. Siciliano, O. Khatib, (Eds.), Springer-Verlag, Heidelberg, D, 2008.

[48] http://www.mathworks.com.

[49] R.S. Inderieden, H.R. Everett, T.A. Heath-Pastore, and R.P. Smurlo. Overview of the mobile detection assessment and response system. In *DND/CSA Robotics and KBS Workshop*, St. Hubert, Quebec, October 1995.

[50] L. Iocchi, D. Nardi, and M. Salerno. Reactivity and deliberation: A survey on multi-robot systems. In *Balancing Reactivity and Social Deliberation in Multi-Agent Systems, From RoboCup to Real-World Applications (selected papers from the ECAI 2000 Workshop and additional contributions)*, pages 9–34, London, UK, 2001. Springer-Verlag.

[51] H. A. Jacobsen. A generic architecture for hybrid intelligent systems. In *IEEE World Congress on Computational Intelligence*, pages 709–714, 1998.

[52] D. Jung and A. Zelinsky. Grounded symbolic communication between heterogeneous cooperating robots. *Autonomous Robots*, 8:269–292, 2000.

[53] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotic Research*, 5(1):90–98, 1986.

[54] B. Khoshnevis and G. Bekey. Centralized sensing and control of multiple mobile robots. *Computer and Industrial Engineering*, 35(3-4):503–506, 1998.

[55] B. M. Kim and P. Tsiotras. Controllers for unicycle-type wheeled robots: theoretical results and experimental validation. *IEEE Transactions on Robotics and Automation*, 18(3):294–307, June 2002.

[56] H. Kitano and S. Tadokoro. Robocup-rescue: A grand challenge for multi-agent and intelligent systems. 22(1):9–52, 2001.

[57] A. Kolling and S. Carpin. Multi-robot surveillance: an improved algorithm for the graph-clear problem. In *Proceedings 2008 IEEE International Conference on Robotics and Automation*, pages 2360–2365, Pasaena, CA, May 2008.

[58] J. Kosecka and R. Bajcsy. Discrete event systems for autonomous mobile agents. *Robotics and Autonomous Systems*, 12:187–198, 1993.

[59] J. Kosecka and H. I. Christensen. Experiments in behaviors composition. *Robotics and Autonomous Systems*, 19:287–298, 1997.

[60] C. R. Kube and E. Bonabeau. Cooperative transport by ants and robots. *Robotics and Autonomous Systems*, 30:85–101, 1998.

[61] A. Machado, G. Ramalho, J. D. Zucker, and A. Drogoul. Multi-agent patrolling: An empirical analysis of alternative architectures. In *Third International Workshop on Multi-Agent-Based Simulation*, pages 155–170, 2002.

[62] A. Machado, G. Ramalho, J.D. Zucker, and A. Drogoul. Multi-agent patrolling: an empirical analysis of alternative architectures. In *Multi-Agent Based Simulation*, pages 155–170, 2002.

[63] A.A. Maciejewski and C.A. Klein. Obstacle avoidance for kinematically redundant manipulators in dynamically varying environments. *The International Journal of Robotics Research*, 4(3):109–117, 1985.

[64] B. Maclennan. Synthetic ethology: An approach to the study of communication. In *Proceedings of the 2nd Interdisciplinary Workshop on Synthesis and Simulation of Living Systems*, pages 631–658, Los Alamos, Addison-Wesky, 1991. Addison-Wesley.

[65] P. Maes. How to do the right thing. *Technical Report. AI-Laboratory, Massachusetts Institute of Technology. Cambridge*, 1989.

[66] S. Mahadevan, J. Connell, C. Sammut, R. Sutton, and Temporal Phd. Automatic programming of behavior-based robots using reinforcement learning, 1992.

[67] A. Mallet, S. Lacroix, I.-K. Jung, and R. Chatila. Towards cooperative air/ground robotics: issues related to environment modeling. In *10th International Conference on Advanced Robotics*, pages 75–80, 2001.

[68] A. Marino, L. Parker, G. Antonelli, and F. Caccavale. Behavioral control for multi-robot perimeter patrol: A finite state automata approach. In *Proceedings 2009 IEEE International Conference on Robotics and Automation*, Kobe, J, May 2009.

[69] A. Marino, L. Parker, G. Antonelli, and F. Caccavale. A fault-tolerant modular control approach to multi-robot perimeter patrol. In *IEEE International Conference on Robotics and Biomimetics (ROBIO 2009)*, Guilin, China, December 2009.

[70] A. Marino, L. Parker, G. Antonelli, and F. Caccavale. A fuzzy-based multiple robots border patrol. In *Proceedings 2009 17th Mediterranean Conference on Control and Automation.*, Thessaloniki, Greece, June 2009.

[71] D. Marthaler and A. L. Bertozzi. Tracking environmental level sets with autonomous vehicles. *Recent Developments in Cooperative Control and Optimization*, 1, 2004.

[72] M. J. Mataric. Issues and approaches in design of collective autonomous agents. *Robotics and Autonomous Systems*, 16:321–331, 1994.

[73] M.J. Mataric. Behavior-based control: Examples from navigation, learning, and group behavior. *Journal of Experimental and Theoretical Artificial Intelligence*, 9(2-3):323–336, 1997.

[74] M.J. Mataric and F. Michaud. *Springer Handbook of Robotics*, chapter Behavior-Based Systems, pages 891–909. B. Siciliano, O. Khatib, (Eds.), Springer-Verlag, Heidelberg, D, 2008.

[75] M. Minsky. The society of mind. *Simon and Schuster*, 1985.

[76] Y. Nakamura, H. Hanafusa, and T. Yoshikawa. Task-priority based redundancy control of robot manipulators. *The International Journal Robotics Research*, 6(2):3–15, 1987.

[77] Esha Nerurkar, Stergios Roumeliotis, and Agostino Martinelli. Distributed Maximum A Posteriori Estimation for Multi-robot Cooperative Localization. In *International Conference on Robotics and Automation*, Kobe, Japan, 05 2009.

[78] F. R. Noreils. Toward a robot architecture integrating cooperation between mobile robots: application to indoor environment. *Int. J. Rob. Res.*, 12(1):79–98, 1993.

[79] L. E. Parker. Cooperative robotics for multi-target observation. *Intelligent Automation and Soft Computing*, 5:5–19, 1999.

[80] L. E. Parker. Current state of the art in distributed autonomous mobile robotics. In *Distributed Autonomous Robotic Systems*, pages 3–12. Springer, 2000.

[81] L. E. Parker. Lifelong adaptation in heterogeneous multi-robot teams: Response to continual variation in individual robot performance. *Auton. Robots*, 8(3):239–267, 2000.

[82] L.E. Parker. Alliance: an architecture for fault tolerant multirobot cooperation. *IEEE Transactions on Robotics and Automation*, 14(2):220–240, 1998.

[83] L.E. Parker. *Springer Handbook of Robotics*, chapter Multiple Mobile Robot Systems, pages 921–941. B. Siciliano, O. Khatib, (Eds.), Springer-Verlag, Heidelberg, D, 2008.

[84] J. O. Peralta and M. T. C. de Peralta. Security pids with physical sensors, real-time pattern recognition, and continuous patrol. *IEEE Transactions on Systems, Man, and Cybernetics, Part C*, 32(4):340–346, 2002.

[85] D. Perugini, D. Lambert, L. Sterling, and A. Pearce. Agent-based global transportation scheduling in military logistics. *Autonomous Agents and Multiagent Systems, International Joint Conference on*, 3:1278–1279, 2004.

[86] P. Pirjanian. Behavior coordination mechanisms – state-of-the-art. *Technical Report IRIS-99-375, Institute of Robotics and Intelligent Systems, School of Engineering, University of Southern California. October 1999.*

[87] P. Pirjanian and J. Perram. Multiple objective action selection and behavior fusion using voting. Technical report, Department of Medical, 1998.

[88] J. Bryan Pletta and John Sackos. An advanced unmanned vehicle for remote applications. In *Sandia National Laboratory Report*, 1998.

[89] J.B. Pomet, B. Thuilot, G. Bastin, and G. Campion. A hybrid strategy for the feedback stabilization of nonholonomicmobile robots. *Proceedings of 1992 IEEE International Conference on Robotics and Automation*, pages 129–134, 1992.

[90] S. Premvuti and S. Yuta. Consideration on the cooperation of multiple autonomous mobile robots. In *Proceedings IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1990.

[91] Dr. Robin, R. Murphy, R. Murphy, Robin R. Murphy, Christine Lisetti, Christine Lisetti, Russ Tardif, Russ Tardif, Liam Irish, Liam Irish, Aaron Gage, and Aaron Gage. Emotion-based control of cooperating heterogeneous mobile robots. *IEEE Transactions on Robotics and Automation*, 18:744–757, 2002.

[92] J. Rosenblatt. Damn: A distributed architecture for mobile navigation - thesis summary. In *Journal of Experimental and Theoretical Artificial Intelligence*, pages 339–360. AAAI Press, 1995.

[93] J. K. Rosenblatt and C. E. Thorpe. Combining multiple goals in a behavior-based architecture. In *Proceedings of the 1995 International Conference on Intelligent Robots and Systems (IROS-95*, pages 7–9, 1995.

[94] M. Forshaw S. Young and Image comparison methods for perimeter surveillance M. Hodgetts. Manchester, UK, 13-15, July 1999.

[95] A. Saffiotti. The uses of fuzzy logic in autonomous robot navigation. *Soft Computing*, pages 180–197, 1997.

[96] A. Saffiotti, K. Konolige, and E. H. Ruspini. A multivalued logic approach to integrating planning and control. *Artificial Intelligence*, 76:481–526, 1995.

[97] C. Samson. Time-varying feedback stabilization of car-like wheeled mobile robots. *International Journal of Robotics Research*, 12(1):55–64, 1993.

[98] T. Schnberg, M. Ojala, J. Suomela, A. Torpo, and A. Halme. Positioning an autonomous off-road vehicle by using fused dgps and inertial navigation. In *Second IFAC Conference on Intelligent Autonomous Vehicles*, pages 226–231, 1995.

[99] L. Sciavicco and B. Siciliano. *Modeling and Control of Robot Manipulators*. Springer-Verlag, London, UK, 2000.

[100] B. Siciliano. Kinematic control of redundant robot manipulators: A tutorial. *Journal of Intelligent Robotic Systems*, 3:201–212, 1990.

[101] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith. First results in the coordination of heterogeneous robots for large-scale assembly. In *Proceedings of the International Symposium on Experimental Robotics (ISER*, 2000.

[102] R. Simmons, S. Singh, D. Hershberger, J. Ramos, and T. Smith. First results in the coordination of heterogeneous robots for large-scale assembly. In *Proceedings of the International Symposium on Experimental Robotics (ISER*, 2000.

[103] D.J. Stilwell and B.E. Bishop. Redundant manipulator techniques for path planning and control of a platoon of autonomous vehicles. In *Proceedings* 41 *st IEEE Conference on Decision and Control*, pages 2093–2098, Las Vegas, NE, Dec. 2002.

[104] D.J. Stilwell, B.E. Bishop, and C.A. Sylvester. Redundant manipulator techniques for partially decentralized path planning and control of a platoon of autonomous vehicles. *IEEE Transactions on Systems, Man and Cybernetics*, 35(4):842–848, Aug. 2005.

[105] G. S. Sukhatme, J. F. Montgomery, and R. T. Vaughan. Experiments with cooperative aerial-ground robots. In *Robot Teams: From Diversity to Polymorphism. AK Peters*, pages 345–367, 2001.

[106] S. Susca, S. Martinez, and F. Bullo. Monitoring environmental boundaries with a robotic sensor network. In *American Control Conference*, pages 2072–2077, Minneapolis, MN, June 2006.

[107] H.G. Tanner, G.J. Pappas, and V. Kumar. Leader-to-formation stability. *IEEE Transactions on Robotics and Automation*, 20(3):443–455, 2004.

[108] S. Thrun and J. J. Leonard. *Springer Handbook of Robotics*, chapter 37. Simultaneous Localization and Mapping, pages 871–889. B. Siciliano, O. Khatib, (Eds.), Springer-Verlag, Heidelberg, D, 2008.

[109] F. Tiche, C. Facchinetti, and H. Hgli. Multi-layered hybrid architecture to solve complex tasks of an autonomous mobile robot. In *International Journal on Articial Intelligence Tool*, pages 6–8, 1998.

[110] C. F. Touzet. Robot awareness in cooperative mobile robot learning. *Auton. Robots*, 8(1):87–97, 2000.

[111] J. P. Trevelya, S. C. Kang, and W. R. Hamel. *Springer Handbook of Robotics*, chapter 48. Robotics in Hazardous Applications, pages 1101–1126. B. Siciliano, O. Khatib, (Eds.), Springer-Verlag, Heidelberg, D, 2008.

[112] J. K. Tsotsos. Behaviorist intelligence and the scaling problem. *Artificial Intelligence*, 75(1):135–160, 1995.

[113] T. Tyrrell. Computational mechanisms for action selection. *PhD thesis. University of Edinburgh.*, 1993.

[114] R. T. Vaughan, B. Gerkey, and A. Howard. The player/stage project: Tools for multi-robot and distributed sensor systems. In *Proceedings of the 11th International Conference on Advanced Robotics, (ICAR)*, pages 317–323, Coimbra, Portugal, June 2003.

[115] M. Yim, D. G. Duff, and K. D. Roufas. Polybot: a modular reconfigurable robot. volume 1, pages 514–520 vol.1, 2000.