A ROS-Based Framework for Simulation and Benchmarking of Multi-robot Patrolling Algorithms

David Portugal, Luca Iocchi and Alessandro Farinelli

Abstract Experiments with teams of mobile robots in the physical world often represent a challenging task due to the complexity involved. One has to make sure that the robot hardware configuration, the software integration and the interaction with the environment is thoroughly tested so that the deployment of robot teams runs smoothly. This usually requires long preparation time for experiments and takes the focus away from what is essential, i.e. the cooperative task performed by the robots. In this work, we present *patrolling_sim*, a ROS-based framework for simulation and benchmarking of multi-robot patrolling algorithms. Making use of Stage, a multi-robot simulator, we provide tools for running, comparing, analyzing and integrating new algorithms for multi-robot patrolling. With this framework, roboticists can primarily focus on the specific challenges within robotic collaborative missions, run exhaustive tests in different scenarios and with different team sizes in a fairly realistic environment, and ultimately execute quicker experiments in the real world by mimicking the setting up of simulated experiments.

Keywords Multi-robot systems · Multi-robot patrol · Simulation Benchmarking · ROS package

D. Portugal (✉)
Ingeniarius, Ltd, R. Coronel Veiga Simão, Edifício CTCV, 3°Piso, 3025-307 Coimbra, Portugal
e-mail: davidbsp@ingeniarius.pt

L. Iocchi
The Department of Computer, Control, and Management Engineering, Sapienza University of Rome, Via Ariosto 25, Rome 00185, Italy
e-mail: iocchi@diag.uniroma1.it

A. Farinelli
The Department of Computer Science, University of Verona, Strada Le Grazie 15, Ca' Vignal 2, Verona, Italy
e-mail: alessandro.farinelli@univr.it

A. Koubaa (ed.), *Robot Operating System (ROS)*, Studies in Computational Intelligence 778, https://doi.org/10.1007/978-3-319-91590-6_1

1 Introduction

The field of Robotics has witnessed significant advances recently, and the generalized use of a common middleware for robotic applications, the Robot Operating System (ROS) [1], has contributed to this phenomenon. Nowadays, researchers do not reinvent the wheel when developing robotic applications, since they often benefit and build upon the community contributions (algorithms, tools, drivers, etc.) in fundamental tasks such as interfacing with sensors, debugging, localization, etc. This also led roboticists to increasingly make their code available as open source, allowing the community to improve and leverage the existing functionality, thus fostering innovation in the field.

Multi-robot systems (MRS) are a research area within Robotics, in which a set of robots operate in a shared environment in order to accomplish a given task. The applications of MRS are vast and have been documented previously in the literature [2]. In fact, when they are efficiently deployed, MRS have several advantages over single robot solutions, such as: distributed control, increased autonomy, ability to communicate, greater fault-tolerance, redundancy, assistance by teammates when needed, space distribution, performing different tasks in parallel, and quicker mission accomplishment [3]. In general, MRS have the potential to increase the robustness and reliability of the robotic solution, remaining functional with some degree of performance degradation in case a member of team fails. However, one of the main challenges in such systems is to coordinate multiple robots so as to execute collective complex tasks efficiently, while maximizing group performance under a variety of conditions and optimizing the available resources. Thus, a coordination mechanism is necessary to select actions, solve conflicts, and establish relationships between robots so they can effectively fulfill the mission [4].

In this work, we present a ROS-based framework for simulation and benchmarking of MRS. In particular, we focus on multi-robot patrolling (MRP) as a case study. In MRP, a set of mobile robots should coordinate their movements so as to patrol an environment. This is a widely studied and challenging problem for MRS coordination with a wide range of practical application scenarios (see Sect. 2).

In more detail, by making use of Stage [5], a scalable and fairly realistic multi-robot simulator, we provide tools for running, comparing, analyzing and integrating new algorithms for the coordination of multiple robots performing patrolling missions. Our main goal is to relief researchers from the effort of setting up complex MRS experiments, shifting the focus to the coordination mechanism between robots, enabling exhaustive tests in different scenarios and with different team sizes, and bridging the gap between simulations and real world experiments.

In the next section, we provide the motivation and background behind this work, and in Sect. 3, the proposed framework for MRP simulation and benchmarking named *patrolling_sim* is described in detail. In Sect. 4, we discuss challenges, benefits of using the framework and lessons learned, and finally the chapter ends with conclusions and future work.

2 Background

Multi-robot systems and related subjects, such as design [6], communication [7], and path-finding [8] gained increased attention during the 80s. Still, early work on inspection robots [9], navigation of patrol robots [10], and robot security guards [11] focused exclusively on single robot solutions. In the end of the 80s and beginning of the 90s, the first physical multi-robot systems have been documented in pioneering research works with small populations of robots by researchers from Japan and the USA [12–15]. During the 90s, a significant boost in work on MRS has been noticeable, with a growing involvement of European researchers. In this decade, robotics competitions, especially RoboCup [16] played an important role to foster MRS research.

Patrol is generally defined as "the activity of going around or through an area at regular intervals for security purposes" [17]. For MRS, this is a somehow complex mission, requiring an arbitrary number of mobile robots to coordinate their decision-making with the ultimate goal of achieving optimal group performance by visiting all point in the environment, which require surveillance. It also aims at monitoring, protecting and supervising environments, obtaining information, searching for objects and detecting anomalies in order to guard the grounds from intrusion. Hence, a wide range of applications are possible, as exemplified in Table 1.

Employing teams of robots for active surveillance tasks has several advantages over, for instance, a camera-based passive surveillance system [18]. Robots are mobile and have the ability to travel in the field, collect environmental samples, act or trigger remote alarm systems and inspect places that can be hard for static

Table 1 Examples of applications of multi-robot patrol

Area of application	Example
Rescue operations	Monitoring trapped or unconscious
Military operations	Mine clearing
Surveillance and security	Clearing a building
Supervision of hazardous environments	Patrolling toxic environments
Safety	Preventive patrol for gas leak detection
Environmental monitoring	Sensing humidity and temperature levels inside a facility
Planetary exploration	Collecting samples
Cooperative cleaning	Household vacuum and pool cleaning
Areas with restricted access	Sewerage inspection
Vehicle routing	Transportation of elderly people
Industrial plants	Stock storage
Computer systems	War-game simulations

cameras to capture. These capabilities are highly beneficial to safeguard human lives and provide a great amount of flexibility to the deployed system [19].

Early work on applications using teams of mobile robots in surveillance contexts addressed essentially cooperative sweeping, coverage, and multi-robot coordination [20–24]. The MRP as we know it, started to receive more attention following the pioneer work of Machado et al. [25], where the environment was abstracted using a topological representation, i.e., a patrol graph, which connected the key points in the area that needed regular visits by agents. The authors proposed and compared several patrolling architectures, using different agent behaviors, different communication assumptions and decision-making methods. Criteria based on the average and maximum idleness of the vertices were proposed to evaluate the performance of each technique using a simplistic JAVA-based patrolling simulator [26]. However, conclusions were solely drawn on two scenarios, and unweighted edges were used, meaning that agents always take the same time to travel from one vertex to another, independently of the distance between them.

Since then, several different MRP strategies with increasingly less relaxed assumptions have been presented, based on a wide variety of concepts, such as: simple reactive architectures [27], game theory [28–31], task allocation [32, 33], market-based coordination [34–37], graph theory [38–42], gaussian processes theory [43, 44], Markov decision processes [45, 46], evolutionary algorithms [47], artificial forces [48], reinforcement learning [49, 50], swarm intelligence [51–53], linear programming modeling [54], bayesian heuristics [55, 56] and others with sub-optimal results, leading to several detailed dissertations and thesis on the subject [57–66]. Lately, an effort for real world validation of MRP systems has been evident [67–70].

There are slight variations to the MRP. The idleness concept, i.e. the time that a point in the environment spends without being visited by any robot, has been broadly used in the literature as a basic performance metric, while other authors proposed alternatives such as the frequency of visits to important locations [71, 72]. Additionally, different coordination methods for the team of agents have been studied, such as centralized deterministic [73] and distributed probabilistic methods [74].

Important theoretical contributions on the area patrolling problem have also been presented previously [75–79], and it has been showed that the multi-robot patrolling problem is NP-Hard, and it can also be solved optimally for the single robot situation by finding a TSP tour in the graph that describes the environment to patrol (cf. Fig. 1).

In this paper, we propose a patrolling framework focusing on intelligent strategies for coordination of the team in order to effectively visit all the surveillance points that need vigilance inside a target area. Thus, we do not address other variants of the the problem, like border/perimeter patrol [80, 81], and adversarial patrol [28, 82]. Building teams of robots has costs, and producing robots in large quantities may be expensive. Moreover, doing experiments with physical robotic teams is a long-term effort, which requires an integrated implementation, and it lacks the possibility to easily repeat an experiment with different approaches. Thus, choosing a useful, flexible, scalable and realistic multi-robot simulator is a key task [83]. Simulations are repeatable, faster to deploy, and can be fully automatic, enabling the comparison of different algorithms with different setups (e.g., robots types, fleet sizes,

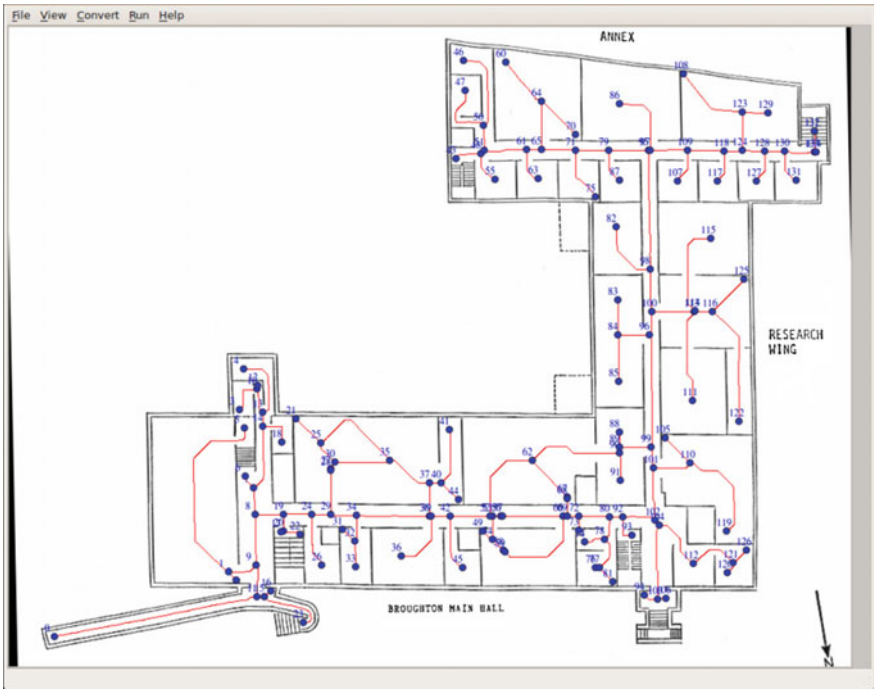

Fig. 1 A patrol graph displayed on top of a metric map to be used in multi-robot patrolling tasks. The blue dots represent the vertices of the graph that must be visited, while the red arcs represent the edges that connect pairs of vertices

environments), and simulation testbeds for MRS are nowadays crucial to rapidly reproduce experiments [84]. While ROS and Stage provide the key building blocks to develop realistic simulations of robotics systems, there is no ready-to-use framework that allows researchers to run experiments testing and validating multi-robot coordination strategies.

Against this background we present *patrolling_sim*,[1] a ROS-based framework for simulation and benchmarking of multi-robot patrolling algorithms, which has been developed and used by the authors in previous works [33, 85]. The *patrolling_sim* framework allows to run exhaustive tests in different scenarios and with different team sizes in fairly realistic environments, and ultimately to execute quicker experiments in the real world by mimicking the setting up of simulated experiments. In the next section, we describe such a framework in more details.

[1] http://wiki.ros.org/patrolling_sim.

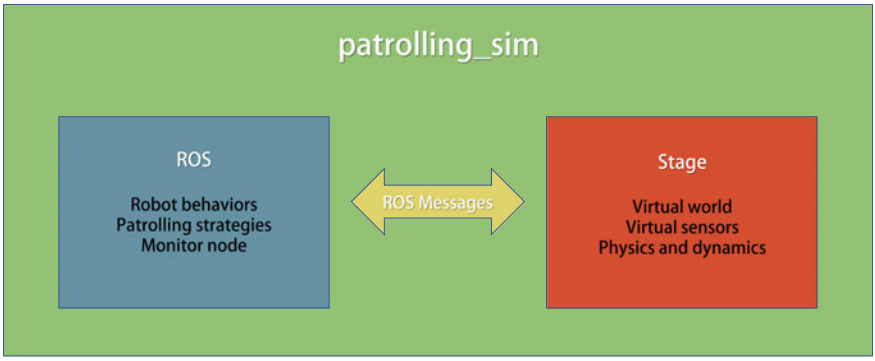

Fig. 2 High-level overview of the patrolling simulation framework

3 Patrolling Simulation Framework

Work on the *patrolling_sim* began in 2010 with the need to compare distinct multi-robot patrolling strategies [86] using a simulation environment and different team sizes. At the time, ROS CTurtle, the second official release of ROS, was used, and 5 patrolling strategies were implemented and integrated: Conscientious Reactive (CR) [25], Heuristic Conscientious Reactive (HCR) [57], Heuristic Pathfinder Conscientious Cognitive (HPCC) [57], Cyclic Algorithm for Generic Graphs (CGG) [38] and the Multilevel Subgraph Patrolling (MSP) Algorithm [38]. Over the years, several utilities, features and algorithms were progressively added, and the framework has been migrated to recent ROS distributions, being currently supported in ROS Kinetic Kame. Figure 2 illustrates the main components of *patrolling_sim*. In the next subsections, we take a deeper look into these components: ROS and Stage, and we overview and highlight some of the key design choices and features available in *patrolling_sim*.

3.1 Robot Operating System (ROS)

Despite the existence of many different robotic frameworks that were developed in the last decades, the Robot Operating System (ROS) has already become the most trending and popular robotic framework, being used worldwide due to a series of features that it encompasses, and being the closest one to become the standard that the robotics community urgently needed. The required effort to develop any robotic application can be daunting, as it must contain a deep layered structure, starting from driver-level software and continuing up through perception, abstract reasoning and beyond. Robotic software architectures must also support large-scale software integration efforts. Therefore, usually roboticists end up spending excessive time

with engineering solutions for their particular hardware setup [87]. In the past, many robotic researchers solved some of those issues by presenting a wide variety of frameworks to manage complexity and facilitate rapid prototyping of software for experiments, thus resulting in the many robotic software systems currently used in academia and industry, like YARP [88], Orocos [89], CARMEN [90] or Microsoft Robotics Studio [91], among others. Those frameworks were designed in response to perceived weaknesses of available middlewares, or to place emphasis on aspects which were seen as most important in the design process. ROS is the product of trade-offs and prioritizations made during this process [1].

The major goals of ROS are hardware abstraction, low-level device control, implementation of commonly-used functionalities, message-passing between processes and package management. ROS promotes code reuse with different hardware by providing a large amount of libraries available for the community, like laser-based 2D SLAM [92], 3D point cloud based object recognition [93], among others, as well as tools for 3D visualization (*rviz*), recording experiments and playing back data offline (*rosbag*), and much more.

Regular updates and broad community support enable the users to obtain, build, write, test and run ROS code, even across multiple computers, given its ability to run distributedly in many processors. Additionally, since it is highly flexible, with a simple and intuitive architecture, ROS allows reusing code from numerous other open-source projects such as several Player robot drivers, the Stage 2D and Gazebo 3D simulation environments, Orocos, mostly for industrial robots and machine control, vision algorithms from the Open Source Computer Vision (OpenCV) library [94], etc. As a result, integrating robots and sensors in ROS is highly beneficial.

Due to its peer-to-peer, modular, tools-based, free and open-source nature, ROS helps software developers in creating robotic applications in a quick and easy way. These applications can be programmed in C++, Python, LISP or Java, making ROS a language-independent framework. Furthermore, ROS places virtually all complexity in libraries, only creating small executables, i.e. *nodes*, which expose library functionalities to ROS. *Nodes* communicate by publishing or subscribing to messages at a given *topic*. The *topic* is a message bus, typically named so that it is easy to identify the content of the *message*. Hence, a *node* that requires a certain kind of data, subscribes to the appropriate *topic*. There may be multiple concurrent publishers and subscribers for a single *topic*, and a single *node* may publish and/or subscribe to multiple *topics*. The idea is to decouple the production of information from its consumption.

Beyond the easiness of using the available tools, ROS also provides seamless integration of new sensors without the need for hardware expertise. As a result, the overall time spent in developing software is greatly reduced due to code reuse and hardware abstraction, when using available ROS drivers to interface with the hardware (Fig. 3).

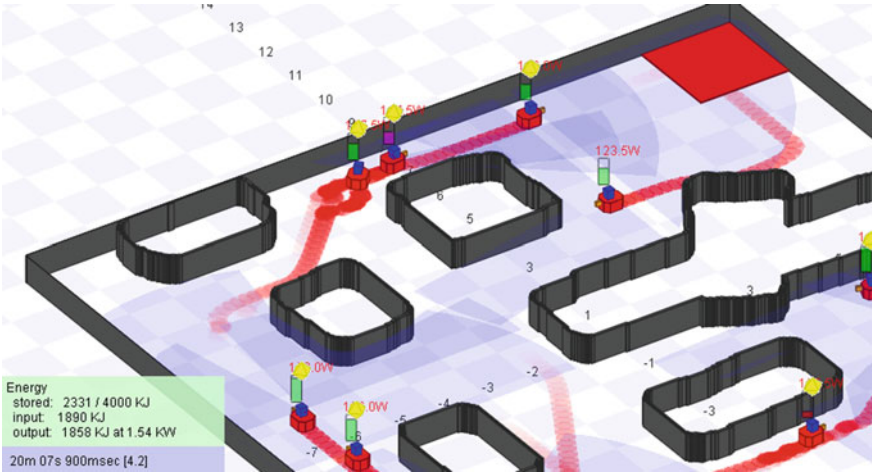

Fig. 3 Example of a simulation in Stage. Extracted from http://playerstage.sourceforge.net/?src=stage

3.2 Stage Multi-robot Simulator

The scalability of multi-robot simulators has always been a known issue. 3D simulators like Gazebo [95], MORSE [96], and V-Rep [97] normally fail to keep up the frame rate and the *simulated versus real* time ratio with teams of low number of mobile robots, such as 5 or 6, with advanced navigation and perception capabilities in modern day computers. Clearly, in order to be able to simulate at least a dozen mobile robots under the abovementioned conditions, a more lightweight simulator is necessary. Stage [98] is a C++ software library designed to support research into multi-agent autonomous systems. Stage simulates not only a population of mobile robots, but also sensors and objects in a two-dimensional (2D) bitmapped environment. It is a 2D dynamic physics simulator with some three-dimensional (3D) extensions, thus commonly being described as a 2.5D (two-and-a-half dimensional) simulator. Its graphical interface is designed using OpenGL, which takes advantage of graphics processor (GPU) hardware, being fast, easy to use, and having wide availability.

Stage was originally developed as the simulation back-end for the Player/Stage system [5]. Player clients developed using Stage usually work with little or no modification on real robots and vice-versa. Thus, Stage allows rapid prototyping of controllers destined for real robots. This is a powerful argument to support the real world validity of Stage-only experiments and a major advantage of using a well-known simulator. Stage also allows experiments with realistic robot devices that one may not happen to have. Various sensors and actuator models are provided, including range-finders (sonars, laser scanners, infrared sensors), vision (color blob detection), 3D depth-map camera, odometry (with drift error model), and differential steer robot

base. Stage is relatively easy to use, it is realistic for many purposes, yielding a useful balance between fidelity and abstraction that is different from many alternative simulators. It runs on Linux and other Unix-like platforms, including Mac OS X, which is convenient for most roboticists, and it supports multiple robots sharing a world. Moreover, Stage is also free and open-source, has an active community of users and developers worldwide, and has reached a well-known status of being a robust simulation platform.

Stage is made available for ROS, through the *stageros* node from the *stage_ros* package,[2] which wraps the Stage multi-robot simulator. Using standard ROS topics, *stageros* provides odometry data from each virtual robot and scans from the corresponding laser model. Additionally, a ROS node may interact with Stage by sending velocity commands to differentially drive the virtual robot.

3.3 Installation and Initializing Experiments

At the time of writing, the patrolling simulation framework supports the latest ROS Long-Term Support (LTS) release: ROS Kinetic Kame. Assuming one is running Ubuntu Linux OS, the installation steps are quite simple as seen below:

1. Install ROS Kinetic Kame, following the instructions at:
   http://wiki.ros.org/kinetic/Installation/Ubuntu
2. Install needed dependencies, by typing in the terminal:
   `$ sudo apt install ros-kinetic-move-base ros-kinetic-amcl ros-kinetic-map-server`
3. Setup your ROS catkin workspace, by typing in the terminal:
   ```
   $ mkdir -p ~/catkin_ws/src
   $ cd ~/catkin_ws
   $ catkin_make
   $ source devel/setup.bash
   ```
4. Add the following two lines to your bash configuration file (at /home/$USER/.bashrc):
   ```
   source ~/catkin_ws/devel/setup.bash
   export ROS_WORKSPACE=~/catkin_ws
   ```
5. Download and compile *patrolling_sim*:
   ```
   $ roscd; cd src
   $ git clone https://github.com/davidbsp/patrolling_sim
   $ roscd; catkin_make
   ```

After successfully downloading and compiling the *patrolling_sim* framework, one can easily initiate and configure multi-robot patrolling experiments by running the *start_experiment.py* script as seen in Fig. 4:

```
$ rosrun patrolling_sim start_experiment.py
```

[2] http://wiki.ros.org/stage_ros.

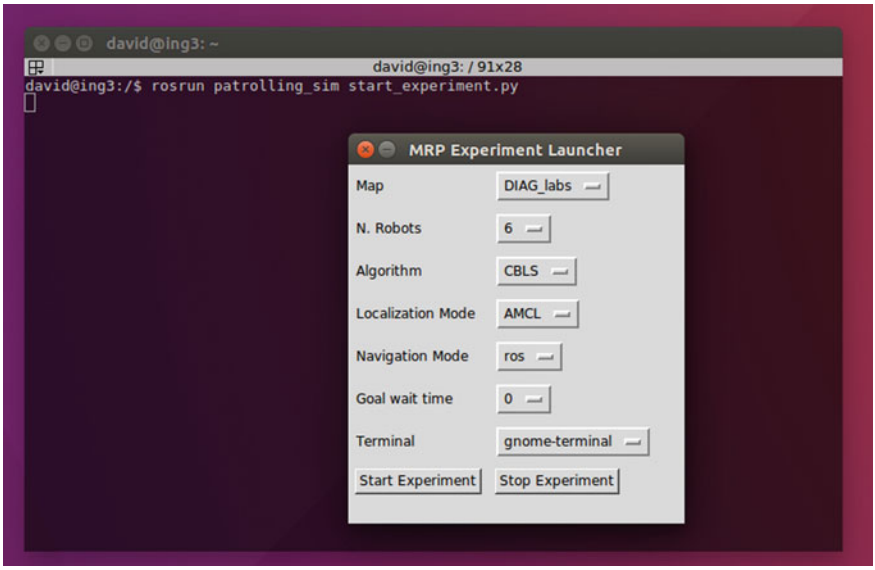

Fig. 4 User configuration interface

The script triggers a user configuration interface that has been implemented using the *TkInter* GUI Programming toolkit for Python [99]. This enables easy configuration of simulated patrolling missions using ROS and Stage. Namely, the configuration interface allows users to choose between different environment maps, robot team sizes, patrolling algorithms, localization modes, navigation modes, waiting times when reaching patrolling goals, and even different types of terminals. Due to the expandability and flexibility of the patrolling framework, the user can easily add additional maps, and patrolling algorithms beyond those referred in Sect. 3.

Currently, two localization modes are supported: Adaptive Monte Carlo Localization (AMCL) and fake localization. AMCL is a probabilistic global localization algorithm [100], which uses a particle filter to track the pose of a robot, by fusing laser scan matching with a source of odometry in order to provide the estimate of the robot's pose with respect to a known map reference frame. Fake localization is a much more lightweight localization node for simulations that provides the same interface to the robots as AMCL, and simply forwards perfect localization information reported by the Stage simulator with negligible computation cost.

Furthermore, the robots leverage from autonomous navigation by following two possible approaches: ROS navigation or *spqrel_navigation*. On one hand, ROS navigation [101] is available out of the box in ROS via the navigation metapackage. This way, given any physically reachable goal, the robot should be able to autonomously navigate to that goal, avoiding collisions with obstacles on the way by following a series of steps. The navigation system at the high level is fairly simple: it takes in a navigation goal, data from sensors, and localization information, and outputs

velocity commands that are sent to the mobile robot base via the *move_base* node. Autonomous navigation in *patrolling_sim* is achieved with a known *a priori* map. Therefore, the robot will follow informed plans considering distant obstacles. The navigation algorithm includes several interesting features. For instance, Random Sample Consensus (RANSAC) is applied to filter out Light Detection And Ranging (LIDAR) readings that are invalid due to hardware limitations in the real world, such as false positives generated by veiling effects. Also, a planar costmap, which is initialized with the static map, is used to represent obstacle data and the most recent sensor data, in order to maintain an updated view of the robots local and global environment. Inflation is performed in 2D to propagate costs from obstacles out to a specified radius in order to conservatively avoid collisions. The global planner uses an A^* algorithm that plans in configuration space computed during obstacle inflation in the costmap, not taking into account the dynamics or the kinematics of the robot, which are considered instead by the local planner, which generates velocity commands for the robot, safely moving it towards a goal. The planner cost function combines distance to obstacles, distance to the path produced by the global planner, and the speed at which the robot travels. Finally, a few recovery behaviors can be performed, e.g. due to entrapment. The robot will perform increasingly aggressive behaviors to clear out space around it, and check if the goal is feasible.

On the other hand, *spqrel_navigation*[3] [102] is a lightweight alternative for ROS navigation, which includes two ROS nodes: *srrg_localizer2d*, a lightweight variant for the AMCL node, and *spqrel_planner*, a lightweight variant for the *move_base* node. They have the same interfaces as AMCL and *move_base*, so they can be used in their replacement with minimal effort. Also *spqrel_navigation* is open source and it has been created with the goal to run on systems with limited computational resources, thus it is very suitable for multi-robot simulations on a single machine or for low-cost multi-robot systems. At the high-level, the *spqrel_navigation* package has the same interfaces of ROS navigation and therefore it can be easily used as a replacement for it. However, a significant decrease in computation load when compared to ROS navigation can be expected.

After choosing the desired configuration, and pressing the "start experiment" button, the script will dynamically trigger ROS *launch* files, which will start the necessary ROS *nodes* and *parameters* to accommodate the configurations chosen (e.g. setting the initial position for localization estimation and the actual robots' position in the stage simulator). Moreover, the script will start each different robotic agent with navigation, localization, and communication capabilities in ROS. This is illustrated in Fig. 5.

In addition, one can run a set of experiments using the `run_exp.sh` bash script. After the time defined in the TIMEOUT variable, the command terminates and more instances can be repeated for performing multiple batch experiments. The script template runs a command-line version of the `start_experiments.py` script as many times as intended.

[3] https://github.com/LCAS/spqrel_navigation/wiki.

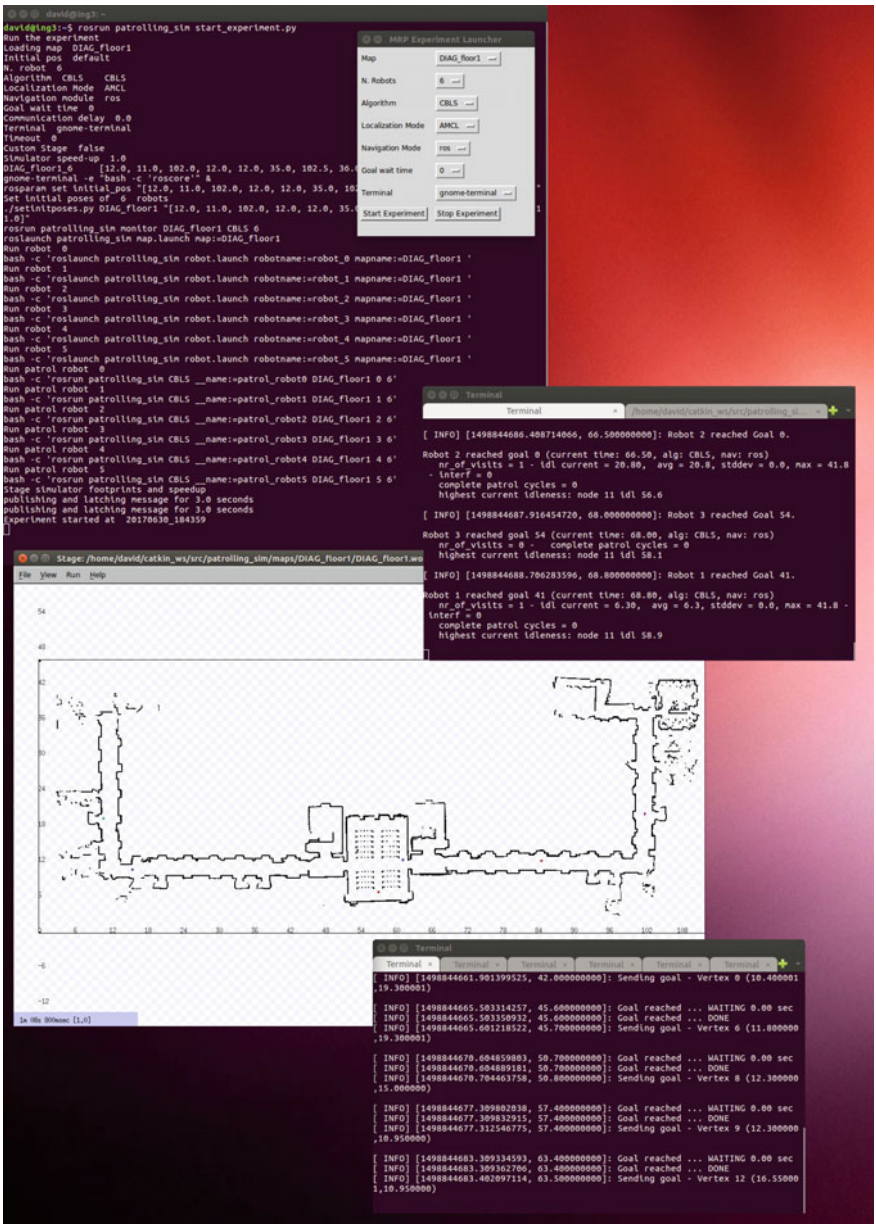

Fig. 5 An experiment running after the initial configuration

3.4 Patrol Agent and Additional Strategies

The patrolling behavior of each robot depends exclusively on the MRP strategy chosen. Typically, the mobile robots within a team follow a similar behavior, only changing their initial conditions, such as their ID and starting position in the environment. During the mission, robots are either given or compute their own waypoints, i.e. vertices of the graph that they should visit, and they continuously coordinate with teammates (for instance, by keeping distances between them, explicitly communicating, etc.) to collectively perform the patrolling task.

Considering the above description of a typical case, it becomes clear that there are several common behaviors within distinct patrolling strategies. Having this in mind, we have provided a general *PatrolAgent* foundation, which can be used for the implementation of robot behaviors. More specifically, *PatrolAgent* represents a base class with general behaviors, which can be extended in derived classes for each specific MRP algorithm that inherit its members and retain its characteristics, and in which they can add their own members.

The common properties of the *PatrolAgent* class include essentially the initialization of agents (assigning the robot ID, extracting relevant map and graph information, initializing control variables, starting positions, idleness tables, ROS publishers and subscribers, etc.); routines for announcing when the robot is ready to start the patrolling mission; actions to perform when the robot moves to a position in the environment, when it arrives there, in case of inter-robot interference and when the simulation finishes; routines for updating parameters based on events, for exchanging poses with other robots, and for saving and sending the robot's own pose.

This way, the inclusion of additional MRP strategies in the *patrolling_sim* framework becomes straightforward and is highly encouraged. One simply needs to create a derived class that inherits all the accessible members of *PatrolAgent*, and modify or add new functions and members to implement the required behaviors of the coordination strategy proposed. This flexibility to add MRP algorithms has resulted in a current total of 11 distinct approaches (cf. Table 2): the 5 original strategies referred in Sect. 3, and 6 additional strategies developed along the years, namely: Random Patrolling (RAND), Greedy Bayesian Strategy (GBS) [103], State Exchange Bayesian Strategy (SEBS) [55], Concurrent Bayesian Learning Strategy (CBLS) [69], Dynamic Task Assignment Greedy (DTAG) [33], and Dynamic Task Assignment based on sequential single item auctions (DTAP) [33].

3.5 Automatically Extracting Results for Analysis

The patrolling framework proposed is based on distributed communication, following a publish/subscribe mechanism, due to its built-in integration in ROS. In the beginning of each test, a specific ROS node is responsible for advertising the start of the mission when all robots are ready, and collecting results during the

Table 2 Overview of MRP strategies in *patrolling_sim*

MRP strategy	Short description
Conscientious reactive (CR)	Robots move locally to the neighbor vertex with higher idleness
Heuristic conscientious reactive (HCR)	Robots decide the neighbor locally, based on idleness and distance
Heuristic pathfinder conscientious cognitive (HPCC)	Robots decide the next vertex globally on the graph, based on idleness, distance, and the vertices in-between
Cyclic algorithm for generic graphs (CGG)	All robots follow the same global route, which visits all vertices in the graph
Multilevel subgraph patrolling (MSP)	Each robot patrols its own region of the graph, using a cyclic strategy in each subgraph
Random patrolling (RAND)	Robots decide randomly the next vertex
Greedy bayesian strategy (GBS)	Robots use local Bayesian decision to maximize their own gain
State exchange bayesian strategy (SEBS)	Similar to GBS, but considers their teammates in the decision to avoid interference
Concurrent bayesian learning strategy (CBLS)	Robots concurrently decide and adapt their moves, according to the system and teammates state, using a reward-based learning technique
Dynamic task sssignment greedy (DTAG)	Robots negotiate greedily the next patrol vertex to visit
Dynamic task assignment based on sequential single item auctions (DTAP)	Robots negotiate all vertices of the graph to build a partition of locations to visit

experiments. This *monitor* node is merely an observer, which analyzes the exchange of communication between robots in the network, and does not provide feedback to them whatsoever. The key objective is to collect experimental results independently, as seen in Fig. 6, which in turn allows benchmarking different MRP algorithms under the same test conditions.

During the patrolling missions, the *monitor* node (cf. Fig. 7) logs several relevant parameters, such as the current idleness of vertices in the graph and corresponding histograms, the average and standard deviation of the idleness of the vertices along time, the total and average number of visits per vertex, the number of complete patrols, the number and rate of inter-robot interference occurrences, the maximum and minimum idleness between all vertices, and the overall average, median and standard deviation of the graph idleness. All these data are saved on files in different formats for later statistical analysis. Some examples of performance metrics and results obtained with *patrolling_sim* are illustrated in [33].

Furthermore, the monitor node controls the patrol termination condition, which can be defined when reaching a given number of patrol cycles (typically a minimum number of visits to all vertices in the graph), as a time window, or any other measurable condition; thus announcing the end of the mission, and stopping the simulation.

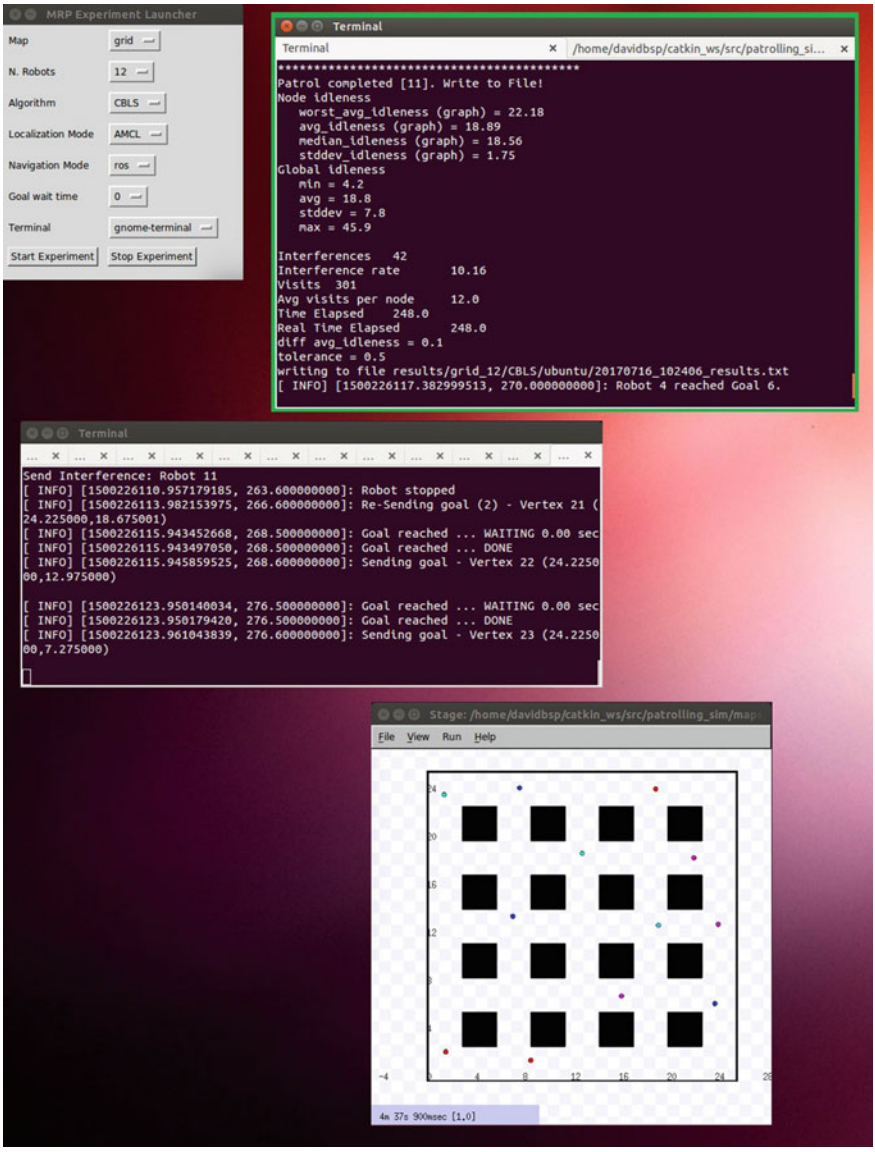

Fig. 6 The monitor node (highlighted in green) announcing the 11th patrol cycle in an experiment with 12 robots in a grid shaped map

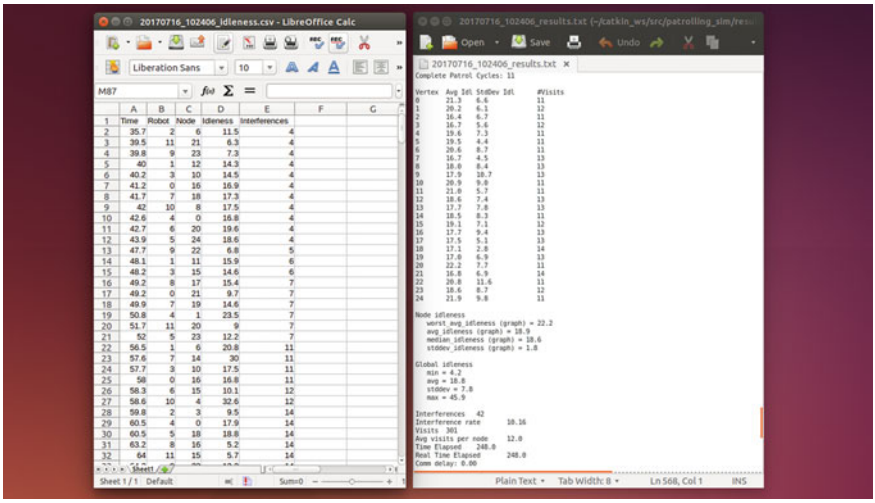

Fig. 7 Log files written by the monitor node, resulting from the experiment of Fig. 6

3.6 Transition to Other Simulators and Real World Experiments

Despite having been developed for use with the Stage multi-robot simulator, *patrolling_sim* can easily be tested with other simulators with minor modifications. To that end, one only needs to launch the framework without resorting to the User Configuration Interface, and replace the ROS launch file that starts the stage simulator with a launch file to start the alternative simulator instead. By having the simulator (environment, robots, sensors, ROS topics, tf frames, etc.) configured similarly to Stage, all the ROS nodes in the system will be able to communicate flawlessly, and simulations will run without issues. In fact, in [83], a multi-robot team on patrol employing the *patrolling_sim* framework, was used as a case study for comparing the Morse and the Gazebo 3D simulators. The quantitative analysis focused on CPU and GPU consumption, thus assessing the scalability of both simulators, and their ability to simulate a limited number of robots. This shows the flexibility and ease of use with other simulators. The patrolling framework has also been tested with the V-REP simulator, according to [104].

In addition to this, *patrolling_sim* can also be exploited for use with teams of physical mobile robots. Some research groups have tested patrolling strategies based on the proposed framework over the past few years. For instance, in [33, 34, 70], experiments with a team of three Turtlebot robots have been described in office-like and corridor-like environments. In [69], experiments with up to six Pioneer-3DX robots have been conducted in a real world large scale infrastructure, and in [68] this number was raised to a total of 8 Turtlebot robots in the experiments reported.

Fig. 8 Example of a real world experiment with 3 turtlebots. Extracted from [33]

According to [33], "*the tests with real robots have been performed by using the same implementation of the algorithms described […]. ROS infrastructure indeed allows for an easy porting from a Stage-based simulated application to a real one. In particular, Turtlebots in the real environment and robots in the Stage simulator use the same map of the environment, the same configuration of parameters for localization and navigation, and the same implementations of the MRP algorithms*". This is illustrated in Fig. 8.

By testing the execution of the developed algorithms with real robots, the portability of the software to a real environment becomes evident. However, besides the complexity involved in setting up teams of mobile robots for patrolling tasks, these experiments present an additional challenge due to the intrinsic characteristics of ROS, which is typically used for centralized applications, e.g. in single robots or architectures with a common point for processing the information. According to [105], in MRS setups, topics and parameters are often complex and may result in duplicities, high computing costs, large demand for communications (specially over Wi-Fi), delay in the processes and other problems related to system handling by an overloaded single ROS master. Therefore, to avoid this situation, robots in a multi-robot ROS architecture commonly run a dedicated master node.

For the aforementioned physical experiments for MRP, roboticists have used several different solutions to enable the communication between robots running dedicated ROS masters. A few works use external tools, such as the lightweight communications and marshalling (LCM) [106], a library independent of ROS that is used to exchange information between robots in [70], or simply UDP broadcast, as in [68]. Moreover, a set of *multimaster* solutions have been integrated as ROS packages. For instance, in [103] the *wifi_comm*,[4] a multi-robot communication and discovery package was used. This was proposed in [107], being based on *for-*

[4] http://wiki.ros.org/wifi_comm.

eign_relay[5] to register topics on foreign ROS masters, and the Optimized Link State Routing (OLSR) protocol to monitor robots connecting and disconnecting from the network, and allowing the deployment of mobile ad-hoc networks. Another solution used for multi-robot communication in [33] is *tcp_interface*,[6] which provides a ROS node that allows easy translation from ROS messages to strings sent over TCP.

Recently, a very promising solution named *multimaster_fkie*[7] [108] has been employed in the hybrid simulated and physical robot experiments reported in [83]. This package offers a set of nodes to establish and manage a multimaster network, requiring no (or minimal) configuration, and all changes in the ROS system are automatically detected and synchronized. From the developer point of view, no specific routines are necessary, which shows the flexibility of the solution, relying only on a simple configuration of the shared topics, nodes and services between different masters. For this reason, the *multimaster_fkie* package will be put in use in the STOP R&D Project[8] [109], which aims at deploying a commercial security system of distributed and cooperative robots on patrol by 2020.

Supporting communication between different ROS master nodes, allow for the exchange of messages between different robots without the need of a server, and supporting the malfunction of any part of the system without compromising the integrity of the whole system, since there is no central point of failure.

4 Discussion

In this work, we take a step towards providing a standard benchmarking framework for running and comparing different patrolling strategies with teams of mobile robots. Due to the complexity of the MRP problem, and the absence of a superior coordination strategy for any environment with any number of robots, we believe that providing a common simulation testbed will allow important advancements in this field of research. Therefore, we welcome the integration of additional strategies by the MRS community.

The simulation of new MRP strategies allows to preliminarily validate them, while enhancing the coordination of robots, the decision-making abilities, and correcting small bugs before moving on to real world experiments. On one hand, real world experiments include noisy sensor readings, localization issues and even robot failures, which may not be precisely modeled in simulation experiments. On the other hand, they may benefit from significant code reuse, tools for analysis, debugging, etc., and the features provided by *patrolling_sim* and ROS itself.

In summary, the patrolling framework proposed is based on the current standard for robotics software – ROS, allowing the easy utilization and transition of experi-

[5] http://wiki.ros.org/foreign_relay.

[6] https://github.com/gennari/tcp_interface.

[7] http://wiki.ros.org/multimaster_fkie.

[8] http://stop.ingeniarius.pt/.

ments to the real world, as well as other simulators beyond Stage, e.g. MORSE or Gazebo. It provides various useful features, such as a graphical user interface for parameterizing the simulations or data logs for performance analysis, and provides a balanced trade-off between realism versus computation load, by making use of the 2.5 D Stage simulator.

Despite the focus on MRP, we hope that this framework can be useful for many other MRS applications due to the common intersections between these, e.g. setting up of MRS teams in simulated environments, launching navigation and localization nodes on several robots in parallel, creating a common structure to deal with the simulation of teams of mobile robots, allowing the communication between multiple robots within a team, etc.

5 Conclusion

In this chapter, we have proposed *patrolling_sim*, a ROS-based framework for simulation and benchmarking of MRP algorithms, which has been used in recent years to study the patrolling problem. The framework proposed enables researchers to run, compare, analyze and integrate new algorithms in commonly adopted simulation testbeds. Thus, it places the focus on the coordination between multi-robot teams, and facilitates the preparation of MRS experiments in the physical world with ROS, by mimicking the setting up of simulated experiments and reusing the source code.

Beyond the inclusion of more algorithms and simulated environments in the framework, in the future we would like to add more features, including the full integration of an automatic method to extract patrol graphs from occupancy grids and select initial robot positions for the robotic team, as well as support for running different simulators directly from the configuration GUI.

Acknowledgements This work was supported by the **S**eguranças robó**T**icos co**OP**erativos (STOP) research project (ref. CENTRO-01-0247-FEDER-017562), co-funded by the "Agência Nacional de Inovação" within the Portugal2020 programme.

References

1. M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler, A. Ng, ROS: an open-source robot operating system. ICRA Workshop Open Source Softw. **3**(2), 00 (2009)
2. T. Arai, E. Pagello, L.E. Parker, Advances in multi-robot systems. IEEE Trans. Robot. Autom. **18**(5), 655–661 (2002)
3. R. Rocha, Building Volumetric Maps with Cooperative Mobile Robots and Useful Information Sharing: a Distributed Control Approach based on Entropy. Ph.D. thesis, Faculty of Engineering of University of Porto, Portugal, 2006
4. A. Farinelli, L. Iocchi, D. Nardi, Multirobot systems: a classification focused on coordination. IEEE Trans. Syst. Man Cybern. Part B: Cybern. **34**(5) (2004)

5. B. Gerkey, R. Vaughan, A. Howard, The player/stage project: tools for multi-robot and distributed sensor systems, in *Proceedings of the IEEE International Conference on Advanced Robotics (ICAR 2003)* (Coimbra, Portugal, June 30–July 3 2003), pp. 317–323
6. E. Freund, On the design of multi-robot systems, in *Proceedings of the 1984 IEEE International Conference on Robotics and Automation (ICRA 1984)*, vol. 1 (IEEE, 1984), pp. 477–490
7. K.G. Shin, M.E. Epstein, Communication primitives for a distributed multi-robot system, IN *Proceedings of the 1985 IEEE International Conference on Robotics and Automation (ICRA 1985)*, vol. 2 (IEEE, 1985), pp. 910–917
8. E. Freund, H. Hoyer, Pathfinding in multi-robot systems: soution and applications, in *Proceedings of the 1986 IEEE International Conference on Robotics and Automation (ICRA 1986)*, vol. 3 (IEEE, 1986), pp. 103–111
9. K. Takehara, Nuclear power plant facility inspection robot. Adv. Robot. **3**(4), 321–331 (1989)
10. S. Xie, T.W. Calvert, B.K. Bhattacharya, Planning viewpoints and the navigation route of a patrol robot in a known 2-D encironment, in *Cambridge Symposium on Intelligent Robotics Systems. International Society for Optics and Photonics, SPIE*, vol. 727 (1987), pp. 206–212
11. T. Kajiwara, J. Yamaguchi, J. Kanemoto, S. Yuta, A security guard robot which patrols using map information, in *Proceedings of the IEEE/RSJ International Workshop on Intelligent Robots and Systems (IROS 1989)* (Tsukuba, Japan, 4–6 Sept 1989)
12. S. Premvuti, S. Yuta, Y. Ebihara, Radio communication network on autonomous mobile robots for cooperative motions, in *Proceedings of 14th IEEE Annual Conference of the Industrial Electronics Society (IECON'88)* (Singapore, 25–27 Oct 1988), pp. 32-37
13. F.R. Noreils, Integrating multirobot coordination in a mobile-robot control system, in *Proceedings of the IEEE International Workshop on Intelligent Robots and Systems (IROS 1990)*, Towards a New Frontier of Applications (IEEE, 1993), pp. 43–49
14. A. Matsumoto, H. Asama, Y. Ishida, K. Ozaki, I. Endo, Communication in the autonomous and decentralized robot system ACTRESS, in *Proceedings IEEE International Workshop on Intelligent Robots and Systems (IROS 1990)*, Towards a New Frontier of Applications (IEEE, 1990), pp. 835–840
15. M. Matarić, Minimizing complexity in controlling a mobile robot population, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 1992)* (Nice, France, 1992), pp. 830–835
16. L. Iocchi, D. Nardi, M. Salerno, Reactivity and deliberation: a survey on multi-robot systems, in *Workshop on Balancing Reactivity and Social Deliberation in Multi-Agent Systems*, Lecture Notes in Computer Science, vol. 2103 (Springer, Berlin Heidelberg, 2001), pp. 9–32
17. Webster's Online Dictionary (2017), http://www.webster-dictionary.org
18. C. King, M. Valera, R. Grech, J. R. Mullen, P. Remagnino, L. Iocchi, L. Marchetti, D. Nardi, D. Monekosso, M. Nicolescu, Multi-robot and multi-camera patrolling, in *Handbook on Soft Computing for Video Surveillance* (CRC Press, 2012), pp. 255–286
19. D. Portugal, Effective Cooperation and Scalability in Mobile Robot Teams for Automatic Patrolling of Infrastructures. Ph.D. thesis, Faculty of Science and Technology, University of Coimbra, Portugal, 2013
20. F.R. Noreils, Multi-robot coordination for battlefield strategies, in *Proceedings of the 1992 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 1992)*, vol. 3 (Raleigh, North Carolina, USA, 7–10 July 1992), pp. 1777–1784
21. D. Kurabayashi, J. Ota, T. Arai, E. Yoshida, Cooperative sweeping by multiple mobile robots, in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation (ICRA 1996)*, vol. 2 (Minneapolis, Minesota, USA, 22–28 April 1996), pp. 1744–1749
22. L.E. Parker, B.A. Emmons, Cooperative multi-robot observation of multiple moving targets, in *Proceedings of the 1997 IEEE International Conference on Robotics and Automation (ICRA 1997)*, vol. 3 (Albuquerque, New Mexico, USA, 25–26 April 1997)
23. J. Feddema, C. Lewis, P. Klarer, Control of multiple robotic sentry vehicles, in *AeroSense'99, Proceedings of the SPIE*, Unmanned Ground Vehicle Technology, vol. 3693 (Orlando, Florida, USA, 7–8 April 1999), pp. 212–223,

24. I.A. Wagner, M. Lindenbaum, A.M. Bruckstein, Distributed covering by ant-robots using evaporating traces. IEEE Trans. Robot. Autom.n **15**(5), 918–933 (1999)
25. A. Machado, G. Ramalho, J. Zucker, A. Drogoul, Multi-agent patrolling: an empirical analysis of alternative architectures, in *Multi-Agent-Based Simulation II*, Lecture Notes in Computer Science, vol. 2581 (Springer, Berlin, 2003), pp. 155–170
26. D. Moreira, G. Ramalho, P. Tedesco, SimPatrol - towards the establishment of multi-agent patrolling as a benchmark for multi-agent systems, in *Proceedings of the 1st International Conference on Agents and Artificial Intelligence (ICAART 2009)* (Porto, Portugal), pp. 570–575
27. A. Almeida, G. Ramalho, H. Santana, P. Tedesco, T. Menezes, V. Corruble, Y. Chaveleyre, Recent advances on multi-agent patrolling, in *Advances in Artificial Intelligence (SBIA 2004)*, Lecture Notes in Computer Science, vol. 3171 (Springer, Berlin, 2004), pp. 474–483
28. N. Basilico, N. Gatti, T. Rossi, S. Ceppi, F. Amigoni, Extending algorithms for mobile robot patrolling in the presence of adversaries to more realistic settings, in *Proceedings of the International Conference on Intelligent Agent Technology (IAT09)* (Milan, Italy, 2009), pp. 557–564
29. J. Pita M. Tambe, C. Kiekintveld, S. Cullen, E. Steigerwald, GUARDS-innovative application of game theory for national airport security, in *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI11)*, vol 3 (Spain, Barcelona, 2011), pp 2710–2715
30. E. Hernández, A. Barrientos, J. del Cerro, Selective smooth fictitious play: an approach based on game theory for patrolling infrastructures with a multi-robot system. Expert Syst. Appl. **41**(6), 2897–2913 (2014). Elsevier
31. P. de Souza, C. Chanel, S. Givigi, A game theoretical formulation of a decentralized cooperative multi-agent surveillance mission, in *4th Workshop on Distributed and Multi-Agent Planning (DMAP)* (London, UK, 2016)
32. F. Sempé, A. Drogoul, Adaptive patrol for a group of robots, in *Proceedings of the International Conference on Robots and Systems (IROS 2003)* (Las Vegas, USA, 2003)
33. A. Farinelli, L. Iocchi, D. Nardi, Distributed on-line dynamic task assignment for multi-robot patrolling. Auton. Robot. J. **41**(6), 1321–1345 (2017). Springer
34. C. Pippin, H. Christensen, L. Weiss, Performance based task assignment in multi-robot patrolling, in *Proceedings of the ACM Symposium on Applied Computing (SAC 2013)* (Coimbra, Portugal, 18–22 Mar 2013)
35. K. Hwang, J. Lin, H. Huang, Cooperative patrol planning of multi-robot systems by a competitive auction system, in *Proceedings of the ICROS-SICE International Joint Conference* (Fukuoka, Japan, 18–21 Aug 2009)
36. C. Poulet, V. Corruble, A. Seghrouchini, Working as a team: using social criteria in the timed patrolling problem, in *Proceedings of the 24th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2012)* (Athens, Greece, 7–9 Nov 2012)
37. A. Sugiyama, T. Sugawara, Improvement of robustness to environmental changes by autonomous divisional cooperation in multi-agent cooperative patrol problem, in *Advances in Practical Applications of Cyber-Physical Multi-Agent Systems, 15th International Conference PAAMS 2017*, Lecture Notes in Artificial Intelligence, vol. 10349 (Springer, Berlin, 2017), pp. 259–271
38. D. Portugal, R. Rocha, MSP algorithm: multi-robot patrolling based on territory allocation using balanced graph partitioning, in *Proceedings of 25th ACM Symposium on Applied Computing (SAC 2010)*, Special Track on Intelligent Robotic Systems (Sierre, Switzerland, 22–26 Mar 2010), pp. 1271–1276
39. T. Sak, J. Wainer, S. Goldenstein, Probabilistic multiagent patrolling, in *Brazilian Symposium on Artificial Intelligence (SBIA 2008)* (Salvador, Brazil, 26–30 Oct 2008)
40. R. Stranders, E.M. de Coteb, A. Rogers, N.R. Jennings, Near-optimal continuous patrolling with teams of mobile information gathering agents, in *Artificial Intelligence* (Elsevier, 2012)
41. P. Fazli, A. Davoodi, A.K. Mackworth, Multi-robot repeated area coverage. Auton. Robot. **34**(4), 251–276 (2013)

42. A. Koubâa, O. Cheikhrouhou, H. Bennaceur, M. Sritim, Y. Javed, A. Ammar, Move and improve: a market-based mechanism for the multiple depot multiple travelling salesmen problem. J. Intell. Robot. Syst. **85**(2), 307330 (2017)
43. A. Marino, L. Parker, G. Antonelli, F. Caccavale, Behavioral control for multi-robot perimeter patrol: a finite state automata approach, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2009)* (Kobe, Japan, 2009), pp. 831–836
44. A. Marino, G. Antonelli, A.P. Aguiar, A. Pascoal, A new approach to multi-robot harbour patrolling: theory and experiments, in *Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2012)* (Vilamoura, Portugal, 7–12 Oct 2012)
45. J. Marier, C. Besse, B. Chaib-draa, Solving the continuous time multiagent patrol problem, in *Proceedings of the International Conference on Robotics and Automation (ICRA 2010)* (Anchorage, Alaska, USA, 2010)
46. X. Chen, T.S. Yum, Patrol districting and routing with security level functions, in *Proceedings of the International Conference on Systems, Man and Cybernetics (SMC2010)* (Istanbul, Turkey, Oct 2010), pp. 3555–3562,
47. O. Aguirre, H. Taboada, An evolutionary game theory approach for intelligent patrolling. Procedia Comput. Sci. Part II **12**, 140–145 (2012)
48. P. Sampaio, G. Ramalho, P. Tedesco, The gravitational strategy for the timed patrolling, in *Proceedings of the International Conference on Tools with Artificial Intelligence (ICTAI10)* (Arras, France, 27–29 Oct 2010)
49. Y. Ishiwaka, T. Sato, Y. Kakazu, An approach to the pursuit problem on a heterogeneous multiagent system using reinforcement learning. Robot. Auton. Syst. (RAS) **43**(4) (2003)
50. H. Santana, G. Ramalho, V. Corruble, B. Ratitch, Multi-agent patrolling with reinforcement learning, in *Proceedings of the International Conference on Autonomous Agents and Multiagent Systems*, vol. 3 (New York, 2004)
51. V. Yanovski, I.A. Wagner, A.M. Bruckstein, A distributed ant algorithm for efficiently patrolling a network. Algorithmica **37**, 3765–186 (2003)
52. H. Chu, A. Glad, O. Simonin, F. Sempé, A. Drogoul, F. Charpillet, Swarm approaches for the patrolling problem, information propagation vs. pheromone evaporation, in *Proceedings of the 19th IEEE International Conference on Tools with Artificial Intelligence (ICTAI 2007)*, vol. 1 (IEEE, 2007), pp. 442–449
53. H. Calvo, S. Godoy-Calderon, M.A. Moreno-Armendáriz, V.M. Marínez-Hernández, Patrolling routes optimization using ant colonies, in *Pattern Recognition, 7th Mexican Conference (MCPR 2015)*, Lecture Notes in Computer Science, vol. 9116 (Springer, Berlin, 2015), pp. 302312
54. B.B. Keskin, S. Li, D. Steil, S. Spiller, Analysis of an integrated maximum covering and patrol routing problem. Transp. Res. Part E: Logist. Transp. **48**, 215–232 (2012). Elsevier
55. D. Portugal, R.P. Rocha, Scalable, fault-tolerant and distributed multi-robot patrol in real world environments, in *Proceedings of the 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems* (Tokyo, Japan, 3–7 Nov IROS 2013)
56. H. Chen, T. Cheng, S. Wise, Developing an online cooperative police patrol routing strategy. Comput. Environ. Urban Syst. **62**, 19–29 (2017). Elsevier
57. A. Almeida, Patrulhamento Multiagente em Grafos com Pesos. M.Sc. thesis, Centro de Informtica, Univ. Federal de Pernambuco, Recife, Brazil, Oct 2003 (In Portuguese)
58. D. Moreira, SimPatrol: Um simulador de sistemas multiagentes para o patrulhamento. M.Sc. thesis, Centro de Informática, Univ. Federal de Pernambuco, Recife, Brazil, Sept 2008 (In Portuguese)
59. D. Portugal, RoboCops: A Study of Coordination Algorithms for Autonomous Mobile Robots in Patrolling Missions, Master of Science Dissertation, Faculty of Science and Technology, University of Coimbra, Portugal, Sept 2009
60. A. Franchi, Decentralized Methods for Cooperative Task Execution in Multi-robot Systems. Ph.D. thesis, Department of Computer and System Science, Sapienza University of Rome, Italy, Dec 2009

61. Y. Elmaliach, Multi-Robot Frequency-Based Patrolling. Ph.D. thesis, Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel, Jan 2009
62. N. Agmon, Multi-Robot Patrolling and Other Multi-Robot Cooperative Tasks: An Algorithmic Approach. Ph.D. thesis, Department of Computer Science, Bar-Ilan University, Ramat Gan, Israel, Feb 2009
63. F. Pasqualetti, Secure Control Systems: A Control-Theoretic Approach to Cyber-Physical Security, Ph.D. thesis, Department of Mechanical Engineering, University of California, Santa Barbara, USA, Sept 2012
64. P. Fazli, On Multi-Robot Area and Boundary Coverage, Ph.D. thesis, Department of Computer Science, University of British Columbia, Vancouver, Canada, Aug 2013
65. C.E. Pippin, Trust and Reputation for Formation and Evolution of Multi-Robot Teams. Ph.D. thesis, Georgia Institute of Technology College of Computing, Atlanta, Georgia, USA, Dec 2013
66. E.H. Serrato, Cooperative Multi-Robot Patrolling: A study of distributed approaches based on mathematical models of game theory to protect infrastructures. Ph.D. thesis, Universidade Politécnica de Madrid, Escuela Técnica Superior de Ingenieros Industriales, Madrid, Spain, Dec 2014
67. L. Iocchi, L. Marchetti, D. Nardi, Multi-Robot Patrolling with Coordinated Behaviours in Realistic Environments, in *Proceedings of the International Conference on Intelligent Robots and Systems (IROS 2011)* (San Francisco, CA, USA, 25-30 Sept 2011), pp. 2796–2801
68. C. Pippin, H. Christensen, Trust modeling in multi-robot patrolling, in *Proceedings of the 2014 IEEE International Conference on Robotics and Automation (ICRA 2014)* (Hong Kong, China, 2014), pp. 59–66
69. D. Portugal, R.P. Rocha, Cooperative multi-robot patrol with bayesian learning. Auton. Robot. J. **40**(5), 929–953 (2016). Springer
70. C. Yan, T. Zhang, Multi-robot patrol: a distributed algorithm based on expected idleness. Int. J. Adv. Robot. Syst. 1–12 (2016). SAGE
71. M. Baglietto, G. Cannata, F. Capezio, A. Sgorbissa, Multi-robot uniform frequency coverage of significant locations in the environment, in *Distributed Autonomous Robotic Systems*, vol. 8 (Springer, Berlin, 2009)
72. Y. Elmaliach, N. Agmon, G. Kaminka, Multi-robot area patrol under frequency constraints, in *Proceedings of the 2007 IEEE International Conference on Robotics and Automation (ICRA 2007)* (Rome, Italy, 10–14 April 2007), pp. 385–390
73. F. Pasqualetti, J. Durham, F. Bullo, Cooperative patrolling via weighted tours: performance analysis and distributed algorithms. IEEE Trans. Robot. **28**(5), 1181–1188 (2012)
74. D. Portugal, R.P. Rocha, Cooperative multi-robot patrol in an indoor infrastructure, in *Human Behavior Understanding in Networked Sensing, Theory and Applications of Networks of Sensors* (Springer International Publishing, 2014), pp. 339–358
75. Y. Chevaleyre, Theoretical analysis of the multi-agent patrolling problem, in *Proceedings of the 2004 International Conference on Agent Intelligent Technologies (IAT 2004)* (Beijing, China, 20–24 Sept 2004), pp. 302–308
76. F. Pasqualetti, A. Franchi, F. Bullo, On cooperative patrolling: optimal trajectories, complexity analysis and approximation algorithms. IEEE Trans. Robot. **28**(3), 592–606 (2012)
77. S. Smith, D. Rus, Multi-robot monitoring in dynamic environments with guaranteed currency of observations, in *Proceedings of the 49th IEEE Conference on Decision and Control* (Atlanta, Georgia, USA, 2010), pp. 514–521
78. S. Ruan, C. Meirina, F. Yu, K.R. Pattipati, R.L. Popp, Patrolling in a stochastic environment, in *Proceedings of the 10th International Command and Control Research and Technology Symposium (ICCRTS)* (McLean, Virginia, USA, 13–16 June 2005)
79. D. Portugal, C. Pippin, R.P. Rocha, H. Christensen, Finding optimal routes for multi-robot patrolling in generic graphs, in *Proceedings of the 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2014)* (Chicago, USA, 14–18 Sept 2014)
80. Y. Elmaliach, A. Shiloni, G.A. Kaminka, A realistic model of frequency-based multi-robot polyline patrolling, in *Proceedings of the 7th international joint conference on autonomous agents and multiagent systems (AAMAS 2008)*, vol. 1 (2008), pp. 63–70

81. A. Marino, L.E. Parker, G. Antonelli, F. Caccavale, A decentralized architecture for multi-robot systems based on the null-space-behavioral control with application to multi-robot border patrolling. J. Intell. Robot. Syst. **71**, 423–444 (2013)
82. N. Agmon, D. Urieli, P. Stone, Multiagent patrol generalized to complex environmental conditions, in *Proceedings of the 25th Conference on Artificial Intelligence (AAAI 2011)* (San Francisco, CA, 711 Aug 2011)
83. F. M. Noori, D. Portugal, R.P. Rocha, M.S. Couceiro, On 3D simulators for multi-robot systems in ROS: MORSE or Gazebo?, in *Proceedings of the 15th IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR 2017)* (Shanghai, China, 11–13 Oct 2017)
84. Z. Yan, L. Fabresse, J. Laval, N. Bouragadi, Building a ROS-based testbed for realistic multi-robot simulation: taking the exploration as an example. Robotics **6**(3), 1–21 (2017)
85. D. Portugal, R.P. Rocha, Multi-robot patrolling algorithms: examining performance and scalability. Adv. Robot. J. Spec. Issue Saf. Secur. Rescue Robot. **27**(5), 325–336 (2013). Taylor and Francis
86. D. Portugal, R.P. Rocha, On the performance and scalability of multi-robot patrolling algorithms, in *Proceedings of the 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR 2011)* (Kyoto, Japan, 1–5 Nov 2011), pp. 50–55
87. A. Araújo, D. Portugal, M.S. Couceiro, R.P. Rocha, Integrating Arduino-based Educational Mobile Robots in ROS. J. Intell. Robot. Syst. (JINT) Spec. Issue Auton. Robot. Syst. **77**(2), 281–298 (2015). Springer
88. G. Metta, P. Fitzpatrick, L. Natale, Yarp: yet another robot platform. Int. J. Adv. Robot. Syst. (IJARS) **3**(1), 43–48 (2006)
89. H. Bruyninckx, Open robot control software: the OROCOS project, in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA 2001)*, vol. 3 (Seoul, Korea Rep., 21–26 May 2001), pp. 2523–2528
90. M. Montemerlo, N. Roy, S. Thrun, Perspectives on standardization in mobile robot programming: the carneggie mellon navigation (CARMEN) toolkit, in *Proceedings of the 2003 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS2003)* (Las Vegas, Nevada, Oct 2003)
91. J. Jackson, Microsoft robotics studio: a technical introduction. IEEE Robot. Autom. Mag. **14**(4), 82–87 (2007)
92. G. Grisetti, C. Stachniss, W. Burgard, Improved techniques for grid mapping with rao-blackwellized particle filters. IEEE Trans. Robot. **23**(1), 34–46 (2006)
93. R. Rusu, S. Cousins, 3D is here: point cloud library (PCL), in *Proceeding of the IEEE International Conference on Robotics and Automation (ICRA 2011)* (Shanghai, China, 9–13 May 2011)
94. G. Bradski, A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV Library* (OReilly Media, 2008)
95. N. Koenig, A. Howard, Design and use paradigms for gazebo, an open-source multi-robot simulator, in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2004)*, vol. 3 (Sendai, Japan, Sept 28–Oct 2 2004), pp. 2149–2154
96. G. Echeverria, N. Lassabe, A. Degroote, S. Lemaignan, Modular open robots simulation engine: Morse, in *Proceedings of the 2011 IEEE International Conference on Robotics and Automation (ICRA)* (Shanghai, China, 9–13 May 2011), pp. 46–51
97. M. Freese, S. Singh, F. Ozaki, N. Matsuhira, N., Virtual robot experimentation platform v-rep: a versatile 3d robot simulator, in *The IEEE International Conference on Simulation, Modeling, and Programming for Autonomous Robots (SIMPAR 2010)* (Darmstadt, Germany, Springer, 15,18 Nov 2010), pp. 51–62
98. R. Vaughan, Massively multi-robot simulation in stage. J. Swarm Intell. **2**(2–4), 189–208 (2008). Springer
99. M.J. Conway, Python: a GUI development tool. Interact. Mag. **2**(2), 23–28 (1995)
100. S. Thrun, D. Fox, W. Burgard, F. Dellaert, Robust monte carlo localization for mobile robots. Artif. Intell. (AI) **128**(12), 99–141 (2000)

101. E. Marder-Eppstein, E. Berger, T. Foote, B. Gerkey, K. Konolige, The office marathon: Robust navigation in an indoor office environment, in *Proceedings of the 2010 IEEE International Conference on Robotics and Automation (ICRA 2010)* (Anchorage, AK, USA, May 2010), pp. 300–307
102. M.T. Lazaro, G. Grisetti, L. Iocchi, J.P. Fentanes, M. Hanheide, A lightweight navigation system for mobile robots, in *Proceedings of the Third Iberian Robotics Conference (ROBOT 2017)* (Sevilla, Spain, 22–24 Nov 2017)
103. D. Portugal, R.P. Rocha, Distributed multi-robot patrol: a scalable and fault-tolerant framework. Robot. Auton. Syst. **61**(12), 1572–1587 (2013). Elsevier
104. L. Freda, M. Gianni, F. Pirri, Deliverable 4.3: communication and knowledge flow gluing the multi-robot collaborative framework, in *TRADR: Long-Term Human-Robot Teaming for Disaster Response (EU FP7 ICT Project #609763)* (2016), http://www.tradr-project.eu/wp-content/uploads/dr.4.3.main_public.pdf
105. M. Garzón, J. Valente, J. Roldán, D. Garzón-Ramos, J. de León, A. Barrientos & J. del Cerro, Using ROS in multi-robot systems: experiences and lessons learned from real-world field tests, in *Robot Operating System (ROS) - The Complete Reference (vol. 2), Studies in Computational Intelligence*, vol. 707 (Springer, Berlin, 2017)
106. A. Huang, E. Olson, D.C. Moore DC, LCM: lightweight communications and marshalling, in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010)* (Taipei, Taiwan, Oct 1822, 2010), pp. 4057–4062
107. G. Cabrita, P. Sousa, L. Marques, A. de Almeida, Infrastructure monitoring with multi-robot teams, in *Proceedings of the 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2010), Workshop on Robotics for Environmental Monitoring* (Taipei, Taiwan, 18–22 Oct 2010)
108. A. Tiderko, F. Hoeller, T. Röhling, The ROS multimaster extension for simplified deployment of multi-robot systems, in *Robot Operating System (ROS) - The Complete Reference (Vol. 1), Studies in Computational Intelligence*, vol. 625 (Springer, Berlin, 2016), pp. 629–650
109. D. Portugal, S. Pereira, M. S. Couceiro, The role of security in human-robot shared environments: a case study in ROS-based surveillance robots, in *Proceedings of the 26th IEEE International Symposium on Robot and Human Interactive Communication (RO-MAN 2017)* (Lisbon, Portugal, Aug 28–Sept 1 2017)

David Portugal completed his Ph.D. degree on Robotics and Multi-Agent Systems at the University of Coimbra in Portugal, in March 2014. His main areas of expertise are cooperative robotics, multi-agent systems, simultaneous localization and mapping, field robotics, human-robot interaction, sensor fusion, metaheuristics, and graph theory. He is currently working as a Senior Researcher at Ingeniarius Ltd. (Portugal), where he has been involved in the STOP R&D technology transfer project on multi-robot patrolling. He has been involved in several local and EU-funded research projects in Robotics and Ambient Assisted Living, such as CHOPIN, TIRAMISU, Social-Robot, CogniWin and GrowMeUp. He has co-authored over 55 research articles included in international journals, conferences and scientific books.

Luca Iocchi is an Associate Professor at Sapienza University of Rome, Italy. His research activity is focused on methodological, theoretical and practical aspects of artificial intelligence, with applications related to cognitive mobile robots and computer vision systems operating in real environments. His main research interests include cognitive robotics, action planning, multi-robot coordination, robot perception, robot learning, sensor data fusion. He is the author of more than 150 referred papers in journals and conferences in artificial intelligence and robotics, member of the program committee of several related conferences, guest editor for journal special issues and reviewer for many journals in the field. He has coordinated national and international projects and, in particular, he has supervised the development of (teams of) mobile robots and vision systems with cognitive capabilities for applications in dynamic environments, such as RoboCup soccer, RoboCup rescue, RoboCup@Home, multi-robot surveillance, and automatic video-surveillance.

Alessandro Farinelli is an Associate Professor at University of Verona, Department of Computer Science, since December 2014. His research interests comprise theoretical and practical issues related to the development of Artificial Intelligent Systems applied to robotics. In particular, he focuses on coordination, decentralised optimisation and information integration for Multi-Agent and Multi-Robot systems, control and evaluation of autonomous mobile robots. He was the principal investigator for several national and international research projects in the broad area of Artificial Intelligence for robotic systems. He co-authored more than 80 peer-reviewed scientific contributions in top international journals (such AIJ and JAAMAS) and conferences (such as IJCAI, AAMAS, and AAAI).