NoStop: An Open Source Framework for Design and Test of Coordination Protocol for Asymmetric Threats Protection in Marine Environment

Simone Nardi(✉) and Lucia Pallottino

Research Center "E. Piaggio", Università di Pisa, Pisa, Italy
nardi@mail.dm.unipi.it, lucia.pallottino@unipi.it

Abstract. *NoStop* is an open source simulator dedicated to distributed and cooperative mobile robotics systems. It has been designed as a framework to design and test multi–agent collaborative algorithms in terms of performance and robustness. The particular application scenario of a team of autonomous guards that coordinate to protect an area from asymmetric threat is considered. *NoStop* system is an integrated tool able to both evaluate the coordination protocol performance and to design the team of guards involved in the asymmetric threat protection. Moreover, *NoStop* is designed to validate robustness of coordination protocol through the use of a remote pilot that control the intruder motion to escape from the guards that monitor the area and accomplish its mission. The project core is a simulation server with a dynamic engine and a synchronization facility. Different coordination protocol can be designed and easily integrated in *NoStop*. The framework is fully integrated with the Robot Operating System (ROS) and it is completed by a control station where the remote pilot moves the intruder following the guards evolution in a $3D$ viewer.

Keywords: Asymmetric threat · Multi-agent · Simulation · Game theory · Distributed algorithm

1 Introduction

The problem of detecting and accordingly reacting to an asymmetric threat in marine environments is a challenge both from research and technological point of view [1]. The surveillance sensors currently available on naval platforms are sufficient to identify and classify asymmetric threats, in time for the gradual appropriate response to the threat, in nominal working conditions. Indeed, it is well known that adverse weather conditions lead to degradation of sensors performance. As a consequence, the time available for a possible reaction after the detection, identification and classification procedures [2] can be drastically decreased in such conditions [3]. The short time–to–reaction may increase the

J. Hodicky (Ed.): MESAS 2016, LNCS 9991, pp. 176–185, 2016.
DOI: 10.1007/978-3-319-47605-6_14

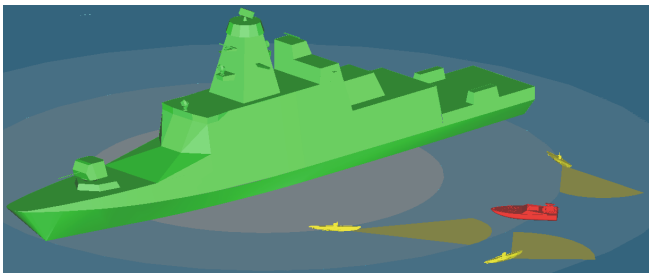

Fig. 1. Example of an asymmetric threat detected by a team of marine autonomous robots. The team of robots must efficiently monitors the area around the ship

possibility of human errors especially in stressful situations (e.g. an incorrect assessment of the necessary reaction).

The goal of innovative surveillance systems is hence to guarantee an adequate supervised area in any working conditions.

In this work, the area monitoring problem considered assumes that every region of the environment can not be under the robot sensor footprint (i.e. monitor) at every time instant. Moreover, it is supposed that the decision process of intruders is unknown to the team of robots (i.e., *guards*). For this reason, it is necessary to test the robustness of each coordination protocol to determine the applicability in real world scenario. To prove the robustness of coordination protocol we developed a novel simulator, *NoStop*, dedicated to the evaluation of coordination protocols of unmanned vehicles involved in asymmetric threats protection. *NoStop* is designed to be a generic framework for implementing and testing multi–robot collaborative algorithms in distributed frameworks. Moreover the *NoStop* framework is suitable for the performance evaluation of algorithms based on different specifications of both monitoring robots (or guards) and intruders such as velocity, probability of detection and available sensors.

The project is fully integrated with the well-known Robot Operating System (ROS) [4], and it is composed by five different modules:

1. a core engine, which manages the interaction between the world, the team of guards and the intruder;
2. a monitoring module, which manages all sensors dedicated to the detection of the intruder;
3. an agent module, which automatically handles everything a vehicle needs to be integrated in the framework, except the robot control laws that must be customized based on the considered scenario;
4. a ground control station, which allows an operator to move the intruder in the scenario and, moreover, shows the current results of the coordination protocol of the team;
5. an evaluation module, which computes performance of the selected coordination protocol.

Main characteristics of the *NoStop* simulator are that it is an open source software[1] and that it can be used to test the robustness and efficiency of designed coordination protocols through an operator that maneuvers the intruder trying evade from monitoring autonomous agents. Another particularly relevant aspect is the fact that for the implementation of a custom agent, a developer receives all the sensors and control actuators as an abstraction layer from the hardware. Such abstraction layer quickly enables the developer to apply the tested control law in a real robot hardware, accordingly changing only the drivers.

The *NoStop* framework has been tested and evaluated with four different known coordination protocols based on game theoretic algorithms [5–7]. Based on the *NoStop* tests results of such approaches a novel distributed coordination protocol [8,9] has been proposed. Moreover, through the *NoStop* framework potentiality and improvements of the proposed protocols have been validated.

The paper is organized as follows: in Sect. 2 the requirements which our framework was designed for, are described. The software architecture is reported in Sect. 3. The hardware integration is addressed in Sect. 4. The application and validity of the *NoStop* framework are described in Sect. 5. Finally Sect. 6 presents some conclusive remarks and the planned future work.

2 Design Requirements

NoStop has been developed with a set of requirements that were identified as main features from state of art simulators. One of the mayor features of our framework is the possibility to use the same code for both simulated and real agents. The only changes required should be the input/output drivers, not the control law or its implementation. Such features are achieved by the fully integration of the developed system with the Robot Operating System (ROS) and the dynamic simulator Gazebo. Moreover, the simulator has not hard–coded assumptions on the evaluated system, such that new modules with new features can be easily integrated into the framework. *NoStop* has been designed as modular as possible, following the Gazebo example that fully represents this approach by using interfaces and plugins. The developed system support multi–process applications where each robot controller runs in its own process or computer, while the simulation handles dynamic integration and model collisions. Such representation leads us to use hardware in the loop during the test of each protocol.

NoStop framework has been designed with the purpose to test and validate coordination protocols of team involved in asymmetric threat protection. In such scenario, an intruder must be detected and tracked by a team of coordinated autonomous guards based on local information on the surrounding (sensed) environment or information exchanged with neighbouring guards. To validate the coordination protocol a double level evaluation process is performed. The first level tests the performance of the coordination protocol with a statistical approach computing the minimum number of coordinated guards required to

[1] The entire code can be downloaded from https://github.com/SimoneNardi/nostop.

protect a given area. The second level tests robustness of the proposed solution thanks to a remote pilot that may control the intruder movements through a joystick.

In the *NoStop* system the coordination protocol is considered as an high level control loop and it is independent from control laws implemented on each agent. To separate high level control from vehicles dynamics, an abstraction layer has been developed to get information from each agents once they have reached the position assigned by the high level control loop.

3 Software Architecture

In this section, some of the main *NoStop* software components are described, focusing on the design choices. From an high level point of view, *NoStop* has a server that acts as the world (*Simulator*), a set of agents (*Clients*), a remote pilot control station (*Visualizer*) and an evaluation module (*Analysis*). The most relevant sub–modules of each component are shown in Fig. 2.

The *simulator* basic loop sends the environmental information to the agents, receives new actuator controls from them, integrates the dynamic, executes the detection update function, based on the agent communication network it sends detection information (e.g., presence of intruders in the area), and starts again.

The *client* control loop handles sensors and actuators, providing the data to the customized low level control loop of each agent. It has access to each agent state and to the inter–agent communication device (with possibly limited area) that allows the development of distributed algorithms. The low level control loop receives the environmental information from the simulator, it executes the coordination protocol under evaluation, and sends the resulting commands to the simulator. Each client has a communication area which is used by a communication filter module to filter all the messages outside the area to simulate a

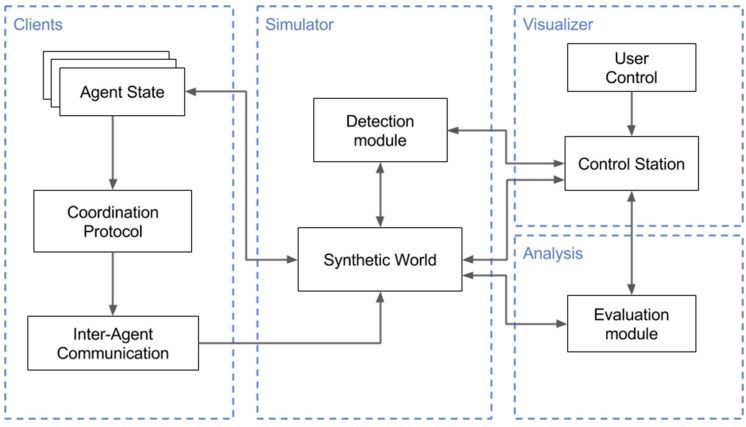

Fig. 2. The *NoStop* software architecture

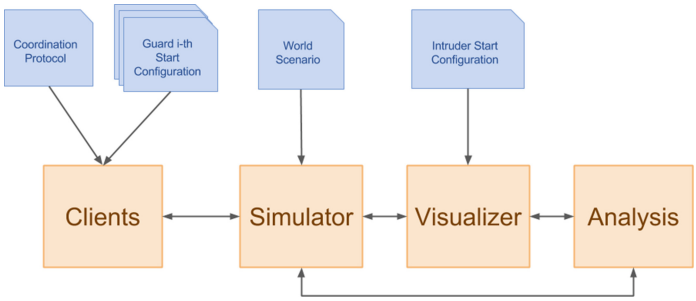

Fig. 3. The *NoStop* configuration files

local communication among agents. Similarly, each client has sensor area which is used to detect the presence of intruders in the area in the detection module.

The *analysis* component computes performance indexes of the chosen coordination protocol based on the agents' states.

The *visualizer* component shows results of the simulations in terms of performance indexes of the coordination protocol and it allows a remote pilot to control the intruder movements in order to test the robustness of the coordination protocol.

The framework parses a configuration file containing description of the world, the agents (both guards and intruder) and the coordination protocol. Such configuration files are used by each component to initialize each control loop, as shown in Fig. 3.

The work–flow of the framework is composed by two phases:

1. Statistical Evaluation phase;
2. Robustness Validation phase.

Once the scenario has been defined, the Statistical Evaluation phase is performed once to design the multi–robot system. In particular, in this phase, the procedure presented in [8] is performed to map the size of the team of guards with respect to the intruder characteristics (e.g. speed or dynamics), intruder detection sensors available, coordination protocol selected and chosen scenario. A statistical Monte Carlo evaluation is performed to identify the minimum number of guards as a function of the intruder velocity and the performance index of the coordination protocol (i.e., rate of coverage or security level of the area, refer to [8] for more details). Once this procedure is completed, robustness of the performed statistical evaluation can be tested once the characteristic of the intruder and the security level of the area have been selected, as reported in Fig. 4. The framework instantiates the correct number of guards that are necessary to control the chosen area with the required security level, and then the remote pilot can move the intruder in the environment to test the robustness of the coordination protocol. Moreover, if during the robustness test, the performance of the protocol is below a given threshold, the minimum number of guards is increased to guarantee the selected security level of the area.

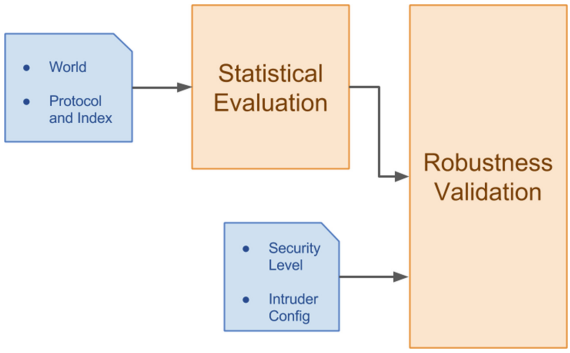

Fig. 4. The *NoStop* flowchart

Once both phases have been performed a validated tool to design the guards team for a given scenario, coordination protocol and intruder characteristic is provided.

3.1 Coordination Protocol Design

Coordination protocols consists of two main features: a local steering function and global performance index. Based on local information available (through a detection sensor or communication with neighbouring robots) each guard computes, with the local steering function, its target configuration. On the other hand, the global performance index is computed by the evaluation module (in the Analysis component) of the *NoStop* framework and it uses information from the whole environment. New coordination protocols can be integrated into the *NoStop* framework using configuration files or by implementing an abstract layer representing a generic coordination protocol. Such abstract layer helps/constraints the user in the design of new coordination protocols. In general, each coordination protocol is composed by three steps:

1. local steering computation;
2. selection of next target configuration;
3. movement towards target;

First information from the detection module is used to evaluate the next target configuration and the motion to reach it. In the end, guards perform the motion towards target configurations based on their own control laws. Once each guard has reached the target configuration, the *NoStop* framework computes the global performance index, and the protocol loop is repeated.

4 Hardware Integration

All the code written in *NoStop* can be reused on the real hardware thanks to the communication facility provided by ROS. When in the configuration file an agent

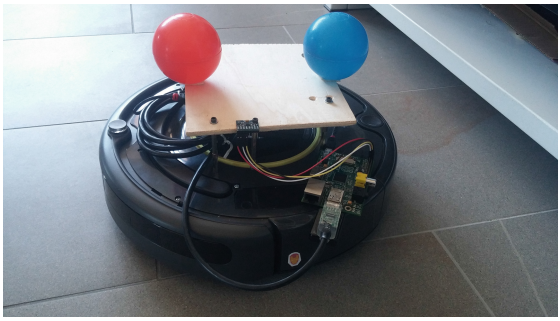

Fig. 5. A vehicle equipped with a *Raspberry Pi* and passive marker (red and blue ball) for localization (Color figure online)

has been declared as not simulated commands are sent via a serial communication directly to the hardware. This is done through a serial communication block that fully complies with the standard messages that the client/server exchanges during the simulation. With this approach only a serial bus device is necessary on the hardware to receive and execute the actuation commands. The commands for the robots currently used are *linear velocity* and *angular velocity*. The hardware used to test the coordination protocols in the laboratory, before performing experiments with marine autonomous vehicles, is shown in Fig. 5.

For the experiments, the localization framework developed in [10] has been used due to the high performance reached in the position and orientation accuracy. In the vehicle there is a *Raspberry Pi*, which is the main core of the robot. On board there are a wireless key to communicate with the other agents and a Linux OS on which the various algorithms run. Plugged to it via a serial bus there is the actuators which receive the actuation commands from the *Raspberry Pi* and sends them to the motors.

One advantage to have this architecture is that we can work with real and simulated robot simultaneously simply by selecting the appropriate type of the robots in the configuration file.

5 *NoStop* Evaluation

Thanks to *NoStop* framework, many different coordination protocols have been verified and developed, both on the software and on the hardware side. In particular, in case of asymmetric threat protection, tested coordination protocols come from the field of game theoretic algorithms.

One protocol tested with the *NoStop* framework is the one proposed in [11], where team reaches correlated equilibria. Another one verified in simulation is the one proposed in [12] where pareto equilibria are reached by the guards. Moreover, the protocols prosed in [6] and in [7] have been verified to converge to Nash equilibria. All those protocols have been designed for static scenarios, where the intruder is not supposed to move, i.e., it is a fixed threat. In case of asymmetric

threats the intruder moves in the environment toward its target hence, the algorithms reported in [8] (*DHSL*) and [9] (*T-DHSL*) have been extended to cope with dynamic scenarios and thus tested and evaluated with the *NoStop* framework. As mentioned above, for each protocol, the *NoStop* simulation computes a map between the minimum number of robots, the intruder velocity and the security level of the selected area.

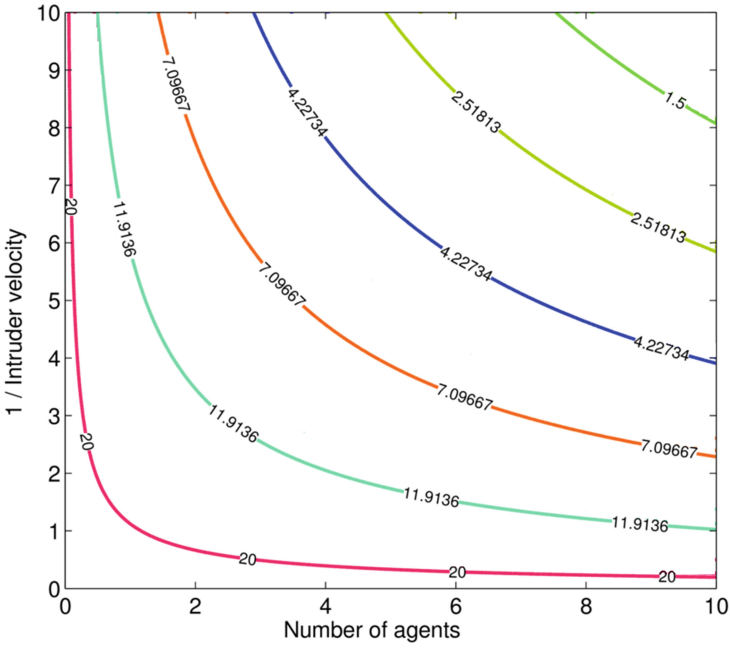

Fig. 6. Tool for design the size of the team of robots: the minimal number of robots is identified with the maximum intruder velocity and the maximal security index of the area

Indeed, in case of open sea scenario (with no close islands or land) the statistical evaluation of the *DHSL* protocol, computes the tool reported in Fig. 6. In this case, the robustness validation phase uses this tool to determine the size of the team: if the remote pilot controls an intruder with maximum velocity equals to 5 and desired security level of 7, the *NoStop* framework, proposes a team composed by 5 vehicles. In Fig. 7 are reported some snapshots from the robustness evaluation phase of the *DHSL* protocol. The figure on the left reports an initial configuration, where the intruder is located in the middle of the scenario and a team of 5 guards are randomly deployed in the environment. The figure on the right reports a *steady* configuration where the intruder is successfully tracked by the team.

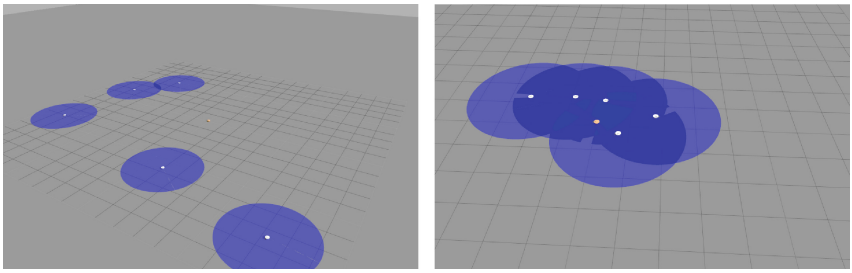

Fig. 7. On the left is reported an initial configuration of the robustness evaluation phase of the *DHSL* protocol. On the right is reported a steady configuration, where the intruder is tracked by the team

6 Conclusion

In this paper, the *NoStop* framework has been presented. The open source simulator is suited to develop and test many coordination protocols for team of guards involved in asymmetric threat protection. In particular it has been designed to be scalable and hence it is able to handle many robots. The hardware integration is straightforward and real robots and simulated one can be used simultaneously. Moreover, with the use of a remote control station, a pilot can interact with the team of guards, controlling the intruder motion in order to evade from the team and evaluate the robustness of the proposed coordination protocol.

One of the main future developments is the management of more than one intruder in the operation scenario and a graphic interface for the evaluation of the performance of the team integrated with a real time comparison with other implemented protocols.

References

1. Blank, S.J.: Rethinking asymmetric threats. Technical report, DTIC Document (2003)
2. Sutton, D.J.: Maritime force protection operations analysis methodology development. In: International Maritime Protection Symposium, USA, pp. 12–14, December 2005
3. Kessel, R.T., Strode, C., Hollett, R.D.: Nonlethal weapons for port protection: scenarios and methodology. In: 5th European Symposium on Non-lethal Weapons (2009)
4. Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., Wheeler, R., Ng, A.Y.: Ros: an open-source robot operating system. In: ICRA Workshop on Open Source Software, vol. 3, p. 5 (2009)
5. Marden, J.R.: Selecting efficient correlated equilibria through distributed learning. In: American Control Conference (ACC), pp. 4048–4053. IEEE (2015)
6. Goto, T., Hatanaka, T., Fujita, M.: Payoff-based inhomogeneous partially irrational play for potential game theoretic cooperative control: convergence analysis. In: American Control Conference (ACC), pp. 2380–2387. IEEE (2012)

7. Zhu, M., Martínez, S.: Distributed coverage games for energy-aware mobile sensor networks. SIAM J. Control Optim. **51**(1), 1–27 (2013)
8. Nardi, S., Santina, C.D., Meucci, D., Pallottino, L.: Coordination of unmanned marine vehicles for asymmetric threats protection. In: OCEANS 2015-Genova, pp. 1–7. IEEE (2015)
9. Nardi, S., Fabbri, T., Caiti, A., Pallottino, L.: A game theoretic approach for antagonistic-task coordination of underwater autonomous robots in asymmetric threats scenarios. In: OCEANS 2016-Monterey. IEEE (2016)
10. Faralli, A., Niko, G., Nardi, S., Pallottino, L.: Indoor real-time localisation for multiple autonomous vehicles fusing vision, odometry and IMU data. In: Hodicky, J. (ed.) MESAS 2016. LNCS, vol. 9991, pp. 288–297. Springer, Heidelberg (2016)
11. Borowski, H.P., Marden, J.R., Shamma, J.S.: Learning efficient correlated equilibria. In: 2014 IEEE 53rd Annual Conference on Decision and Control (CDC), pp. 6836–6841. IEEE (2014)
12. Jason, R.M., Young, H.P., Pao, L.Y.: Achieving pareto optimality through distributed learning. SIAM J. Control Optim. **52**(5), 2753–2770 (2014)