2017 IEEE International Symposium on
Safety, Security and Rescue Robotics (SSRR)
Shanghai, China, October 11-13, 2017

On 3D Simulators for Multi-Robot Systems in ROS: MORSE or Gazebo?

Farzan M. Noori, David Portugal, Rui P. Rocha, and Micael S. Couceiro

***Abstract*—Realistically simulating a population of robots has been an important subject to the robotics community for the last couple of decades. Multi-robot systems are often challenging to deploy in the real world due to the complexity involved, and researchers often develop and validate coordination mechanisms and collaborative robotic behavior preliminarily in simulations. Thus, choosing a useful, flexible and realistic simulator becomes an important task. In this paper, we overview several 3D multi-robot simulators, focusing on those that support the Robot Operating System (ROS). We also provide a comparative analysis, discussing two popular open-source 3D simulators compatible with ROS – MORSE and Gazebo –, using a multi-robot patrolling application, *i.e.* a distributed security task, as a case study.**

I. INTRODUCTION

Research in multi-robot systems has been growing consistently since the eighties [1]. However, one common issue persists to the present day: setting up experiments with multiple robots, especially large populations, is usually a daunting and time-consuming task. For the ease of researchers, different simulators have been developed.

Robotics simulators play an essential part in research as tools for testing safety, efficiency, new concepts, and robustness of algorithms. Moreover, code portability from a simulated to a real platform is another desirable feature in most modern robotics software tools to ensure that the system can be connected to the real robots, with little to no changes required. Due to these needs, a developing number of open-source and commercial simulation toolkits have been outlined. A workshop held in ICRA 2009 [2] has raised the awareness for the robotics community to take interest in the need of open-source middleware and software for programming and simulating robots. A comprehensive survey of robotic simulators based on the feedback provided by the researchers was carried out in 2013 [12]. More recently, in March 2017, a cofounder of Elre Robotics, one of the many companies devoted to create both hardware and software to support robotics development, wrote an article in *Hackernoon* highlighting the importance of robotic simulators, such as the Player project (formerly known as the Player/Stage project [6]) and the Gazebo simulator[1]. The same article enumerates several drawbacks that have significantly slowed down the industrialization of robotics over the past decade, such as the use of proprietary frameworks and the lack of portability and standardized interfaces.

This paper overviews 3D simulators for multi-robot systems, focusing on those that support Robot Operating System (ROS) [3]. Additionally, two widely used simulators – MORSE [4] and Gazebo [5] – are subject to both qualitative and quantitative comparison, using a multi-robot patrolling task [11] as a benchmark. The multi-robot patrolling task is one of the main approaches explored in the ongoing STOP R&D project[2], which aims at deploying a commercial security system of distributed and cooperative robots by 2020. In spite of the targeted ground-breaking real-world application, the team, comprising both public and private entities, has been benchmarking existing realistic simulators as powerful tools for saving development time and money, as well as to mitigate common mistakes made during the design of multi-robot systems. The benchmark comprises multiple features, such as portability and CPU usage, so that forthcoming researchers may assess the most adequate 3D ROS-compatible simulator, according to the requirements of their applications.

The paper is organized as follows: Section II presents the importance of ROS and surveys available robotics simulators. In Section III, both simulations and hybrid experiments (real robot together with simulated robots) are described, comparing the MORSE and Gazebo simulators under a multi-robot patrolling task. Section IV presents and discusses results based on the comparison. The paper is concluded in Section V.

II. RELATED WORK

After introducing the Robot Operating System (ROS), this section reviews available ROS-compatible robotics simulators.

A. ROS Middleware

ROS [3] is an open-source robotics middleware that provides low-level device control, package management and messages passing between processes. ROS is based on services, topics, messages and nodes, wherein nodes can communicate through messages, topics are published and subscribed by nodes, and services are a special type of pairs of messages (one for the request and another for the reply). The ROS Master is one of the key elements of ROS, which allows to monitor all topics and services; it facilitates node registration, and a

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, and by the Institute of Systems and Robotics (ISR) – University of Coimbra (project ref. UID/EEA/00048/2013) funded by the "Fundação para a Ciência e a Tecnologia" (FCT).

Farzan M. Noori, Rui P. Rocha (corresponding author) and M. S. Couceiro are with ISR – University of Coimbra, Portugal. E-mail: {farzan,rprocha,micaelcouceiro} at isr.uc.pt

D. Portugal and M. S. Couceiro are with Ingeniarius Ltd., Portugal. E-mail: {davidbsp,micael} at ingeniarius.pt

[1]https://hackernoon.com/envisioning-the-future-of-robotics-bf529f760b45

[2]http://www.stop.ingeniarius.pt/

parameter server which permits nodes to store and retrieve parameters. The ROS master tools and server are configured with XML files.

ROS fully supports two client libraries/languages: Python *(rospy)*, and C++ *(roscpp)* [3]. Each library utilizes and provides arrangement of ROS tools to facilitate the development of new clients in ROS. The client libraries are sorted out as organized software modules called packages[3]. These packages contain a ROS-independent library and nodes, third-party software, configuration files, and/or other useful modules.

ROS is preferred due to its platform interoperability, modularity, and concurrent resource handling. Moreover, the integration of simulators with ROS allows to evaluate the many state-of-the-art useful robotics algorithms made available by the ROS community, while providing debugging tools, such as the powerful built-in visualization tool (*rviz*) which allows to "see the world through the robots' eyes".

B. Robotics Simulators

This section surveys relevant robotics simulators having in common: *i)* the capability to simulate multiple (possibly many) robots; and; *ii)* some degree of compatibility with ROS.

Starting with 2D simulators, the Simple Two-Dimensional Robot Simulator (STDR)[4] is one of the simplest multi-robot simulators, which does not require any dependencies for installation. STDR is totally ROS submissive: the measurements of every sensor or robot actions are published in appropriate ROS topics. Similarly, Stage [6] is a C++ software library that can simulate large populations of robots in 2D. Stage is a robotics toolkit based on the publish/subscribe paradigm from which ROS [3] was developed later on. Nowadays Stage is made available for ROS through the *stage* package, which wraps the Stage multi-robot simulator.

As for 3D simulators, the Virtual Robot Experimentation Platform (V-REP) is a general purpose robotic simulator developed by Coppelia Robotics[5]. The software is commercial but users can access its educational free version. V-REP is based on a distributed control architecture: each model/object can be individually controlled via a remote API, an embedded script, a ROS node, a plugin, or a custom solution. V-REP has a friendly interface running in Linux, Windows and Mac platforms. Programs can be written in Python, Lua, C/C++, Matlab, Urbi, Octave or Java.

Gazebo [5] began as a venture in the University of Southern California. Later on, John Hsu (ROS maintainer) integrated it with the ROS framework. From then, the Open Source Robotics Foundation (OSRF) has maintained Gazebo along with ROS. It relies on Open Dynamics Engine (ODE) and Object-Oriented Graphics Rendering Engine (OGRE) to provide 3D robots/environments. To achieve a high degree of realism in simulations, the simulated objects have friction, mass, and various attributes. Multiple shapes can be assembled along with different joints to make a simulated robot. Thus, the end user can build and simulate diverse robotic platforms. Gazebo is available for Linux and Mac OSX platforms.

The Modular Open Robots Simulation Engine (MORSE) [4] simulator was created in 2009 at LAAS-CNRS in France, and has now more than 20 academic environment contributors. MORSE is based on the open-source project Blender[6], a 3D game engine that comes with the bullet physics engine integrated. MORSE operates from a command line and it is purely a Python application. Blender supports almost any 3D model, so any model can be imported into MORSE.

Webots [7] is a commercially available software package used to simulate, model and program mobile robots in a 3D simulation environment. The ODE library is used to simulate rigid body dynamics and provides attributes, such as mass, shape, texture, and shape. Webots supports Linux, Windows and MAC OS platforms. Different programming languages, like Java, C++, C, Python, and MATLAB, can be used to build simulations, which can also be implemented in some robot platforms, including Pioneer, iRobot, e-puck, NAO and Lego Mindstorm robots. It is also compatible with different sensors used in robots, such as light sensors, proximity sensors, GPS, touch sensors, lasers, and accelerometers. Webots can also connect to ROS using *roscpp* and *rospy* controller interfaces.

The Unified System for Automation and Robot Simulation (USARSim) [8] is a 3D simulator based on the Unreal Tournament (UT) game engine. USARSim was developed to simulate multiple robots in search and rescue environments and it has been adopted in RoboCup rescue competitions. It interfaces seamlessly with the Mobility Open Architecture Simulation and Tools framework (MOAST), which provides a fully functional modular control system. It supports sound sensors, touch sensors, lasers, odometry, and cameras. USARSim is available for Windows, MAC OS, and Linux platforms.

In 2016, a multi-UAV simulator based on ROS and Unity3D was presented [9]. The interface between ROS and the UAVs is based on the TCP/IP protocol. A particular emphasis was given in modeling the environments and sensors, especially collisions and LIDARs, as real as possible and at high frequency. Different environment modeling, such as buildings, terrains, trees, *etc.*, were introduced in the initial version.

A summary of the main features of the aforementioned simulators is presented in Table I. It also grades the support relating to graphical user interface, tutorials, and the mailing list of the simulators. Gazebo is undoubtedly the most used 3D ROS-compatible simulator, while MORSE provides a flexible interface at variable levels of abstraction and ease of programming. V-REP stays close to MORSE in the comparison table, but we preferred to focus our study in two fully open-source projects: MORSE [4] and Gazebo [5]. In the following sections of this paper, an in-depth comparison of these two simulators is carried out. ROS [3] was used along with them to allow easily migrating code between simulators, and from simulated to real robots.

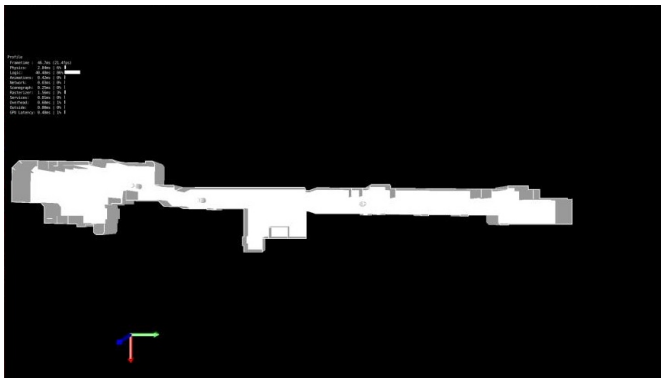
[3]http://wiki.ros.org/gazebo_ros_pkgs

[4]http://stdr-simulator-ros-pkg.github.io/

[5]http://www.coppeliarobotics.com/

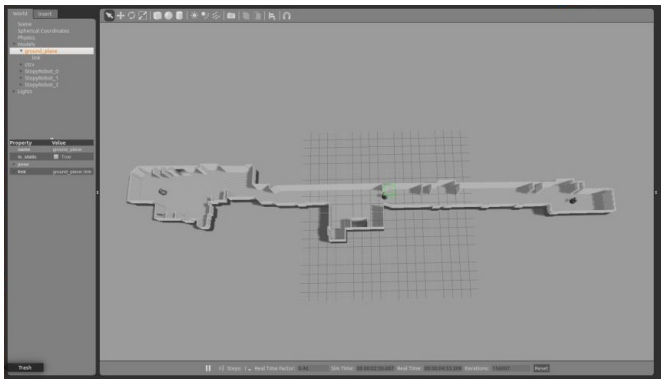
TABLE I: Characteristics of different robotics simulators.

	V-REP	Gazebo	MORSE	Webots	USARSim	STDR/Stage	Unity
Main Program. Language	C++	C++	Python	C++	C++	C++	C++
Operating System	Mac, Linux	Mac, Linux	BSD, Mac, Linux	Linux, Mac	Linux	Linux	Linux
Simulation Type	3D	3D	3D	3D	3D	2D	3D
Physics Engine	ODE, Bullet, Vortex, Newton	ODE, Bullet, Dart	Bullet	ODE	Unreal	OpenGL	Unity 3D
3D Rendering Engine	Internal, External	OGRE	Blender game	OGRE	Karma	-	OGRE
Portability	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Support	****	*****	****	****	***	****	**
ROS Compatibility	****	*****	****	***	**	****	*

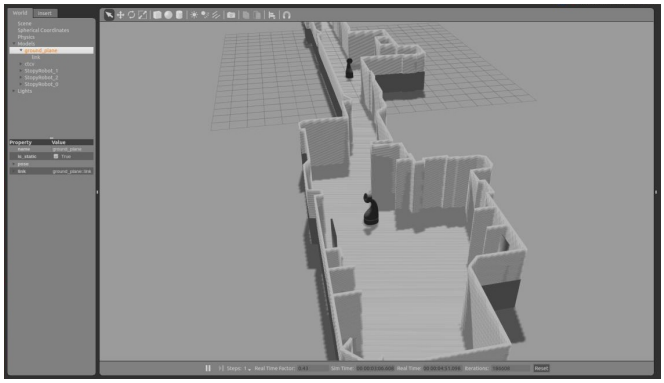


(a) MORSE simulation (top view).

(b) MORSE simulation (isometric view).



(c) Gazebo simulation (top view).



(d) Gazebo simulation (isometric view).

Fig. 1: Simulation environments in MORSE (a, b) and Gazebo (c, d) simulators.

III. A Multi-Robot Patrolling Case Study

To compare MORSE [4] and Gazebo [3], similar experiments were performed on both simulators. Initially, a single robot was used in the environment and more robots were added progressively. The case study used requires multiple robots cooperatively patrolling the whole environment with proper coordination and without collisions. The robot team uses the Concurrent Bayesian Learning Strategy (CBLS) for multi-robot patrolling and coordination [11], which is based on probabilistic reasoning for collective inspection, and distributed communication via lightweight messages. ROS [3] is used to program the behavior of robots, and MORSE [4] and Gazebo [5] are used to simulate dynamics and physics as if we were in the real world. All robots are equipped with a LIDAR and a depth camera. At the start of the simulations, all robots are sent to specific points (set previously) and then the collective patrolling task can start.

The experiments were carried out on both simulators, thus allowing for a comparison between their features. MORSE 1.4 and Gazebo 5.0 were used with the Ubuntu 14.04.4 operating system and ROS Jade. Fig. 1 depicts the simulation environments of both MORSE and Gazebo for a single robot. Fig. 1a and Fig. 1b show the top and isometric views in MORSE, whereas Fig. 1c and Fig. 1d show the top and isometric views of the environment in Gazebo. Fig. 2 depicts the *rqt_graph* and shows all the ROS nodes and topics running in a MORSE simulation. The only difference to Gazebo would be the replacement of the MORSE node with a Gazebo node, on the left-hand side of the picture. Each robot runs navigation, localization, and sensor nodes (in ROS) which consume significant CPU power.

[6]http://www.blender.org/

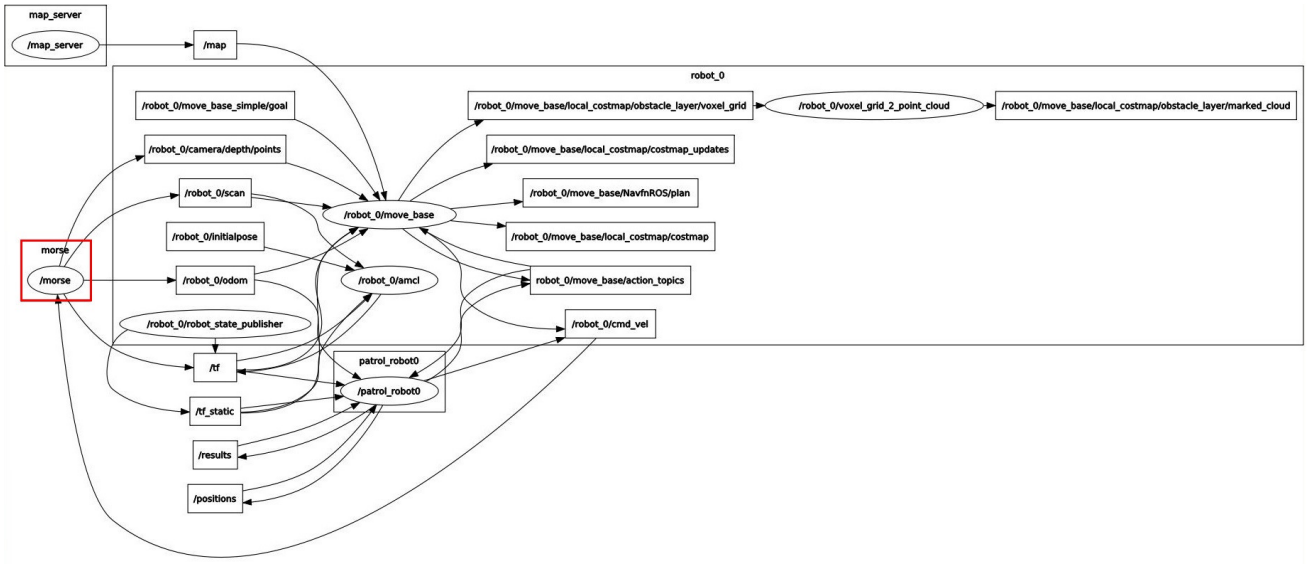


Fig. 2: Graph of ROS nodes and topics for the MORSE simulator.

The following criteria were used to evaluate and compare the two simulators for different team sizes:

i. The real-time factor allowed by the simulator;
ii. GPU load required by the simulator;
iii. CPU load required by the simulator.

As for criterion (i), the real-time ratio can be computed through Eq. (1). If this ratio equals to 1.0, it means that the simulation is running exactly at the same pace as it would in a real-world scenario.

$$Real\ time\ factor = \frac{Simulation\ time}{Real\ time} \tag{1}$$

Furthermore, to cross-validate the results, hybrid experiments were also performed. In these hybrid experiments, two robots were simulated on a computer, while a third robot – the real robot depicted in Fig. 3 – patrolled in the real world environment. In this experiment, the *multimaster_fkie* package[7] was used for communication between the different ROS systems. A long corridor with a 55 m length located in an office area in the building of CTCV, a partner in the STOP R&D project having the role of an end user, was used for both the simulation and hybrid experiments.

IV. Results and Discussion

For the benchmark, a total of twenty trials were performed on each simulator for each configuration. In each trial, the robots had to cover the whole environment at least three times. Below is the evaluation based on the set criteria.

In the case of a single robot, the real-time ratio was 1.0 for MORSE and 0.9 for Gazebo. By adding more robots in the simulated environments, the ratio decreased as simulations ran slower. MORSE's ratio also decreased with the increase of the team size but it performed slightly better than Gazebo as shown in Fig. 4. Afterwards, we added more robots to acquire the maximum number at which the simulator would be able to run with a ratio greater than 0.2. With 6 robots, the simulation time on Gazebo was 5 times slower than the real time, while MORSE had a ratio greater than 0.3, which shows the time efficiency of MORSE over Gazebo for larger team sizes.

Fig. 3: Robot patrolling in the real world.

The laptop used for the simulations is a ThinkPad Lenovo with 8GB RAM memory and a Intel Core i7-6500 CPU@2.5GHz. The laptop has a AMD Radeon (TM) R7 M370 graphics card, which was also used to analyze the performance of the simulators. It can be seen clearly in Fig. 5 that Gazebo presents a higher GPU load than MORSE. The variability of the GPU load of both simulators increases with

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

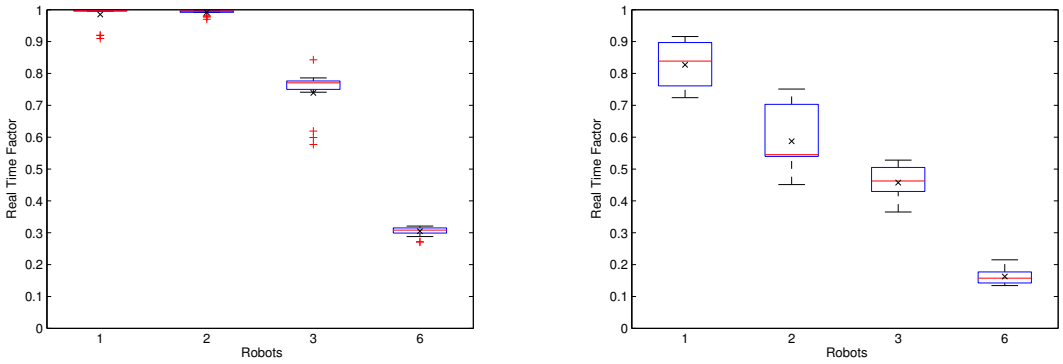


Fig. 4: Real-time ratio for different team sizes: MORSE simulator (left) and Gazebo simulator (right).

the team size. Which (either CPU or GPU) bottleneck hits first depends on the specific game engine or an application. In the case of Gazebo, for a single robot, GPU consumes more load than CPU. Adding more robots into the environment, the GPU alone was not able to handle the required computation, therefore some load was also transfered to the CPU.

We also analyzed the total CPU load of all processes, measured between 0.0 and 1.0 (0% and 100%). Gazebo presented a CPU load equal to 0.61 ± 0.08 for a single robot. As the maximum load was covered by the GPU (0.87 ± 0.09), the simulator presented a low CPU load. In the case of MORSE, both presented a fairly low load *i.e.* 0.73 ± 0.10 for GPU and 0.47 ± 0.07 for CPU. With the increase of teamsize, the CPU load also increased. With 6 robots, the CPU load for Gazebo was 0.75 ± 0.05 and 0.77 ± 0.01 for MORSE. The complete set of results is depicted in Fig. 6. The load distribution between CPU and GPU depends on the type of application. Overall, the performance of the MORSE simulator was better than Gazebo in the multi-robot patrolling simulation experiments.

In order to validate that MORSE is statistically discriminant, we applied the Student's t-test. The p value obtained using MORSE versus Gazebo was 0.0097 (less than 0.01) for 3-robots simulations, thus establishing the statistical significance of MORSE simulator. Furthermore, we increased the number of robots to perform a stress test to assess the criteria analyzed in an extreme situation for each simulator. Gazebo almost stalled when simulating a total of 10 robots. The real-time factor was 0.05 for Gazebo and 0.14 for MORSE, and p values were less than 1.0E-5. MORSE almost stalled when simulating a total of 15 robots, having a real-time factor equal to 0.06 in this experiment. Fig. 7 shows a simulation being run in Gazebo with 10 robots, as seen in *rviz*.

For cross-validation of results, a hybrid experiment was also conducted. Two robots were configured to patrol in the simulator while another robot (see Fig. 3) was moving in the real world. Using MORSE, the task (*i.e.* patrolling the whole environment at least three times) was completed in 5.8 minutes, whereas Gazebo took 13.5 minutes. Furthermore, the CPU load was 0.95 with Gazebo, whereas it was 0.75 with MORSE. The portability of the code between simulated and real robots was good for both simulators. The code written for the simulation environment was easily ported to the real robotic platform and the latter could interact seamlessly with simulated robots as if robots were either all virtual or all real.

Several previous studies have simulated multi-robot systems. In [10], the authors proposed a simulation-based communication system for a cooperative driving system. MORSE along with the NS-3 network simulator was used for better functionality, which shows that MORSE is an emerging simulator, and it is becoming more popular in the community for multi-robot simulations.

The ROS official repository for Gazebo is maintained by the OSRF. The package contains plugins for the interface between ROS and Gazebo. In the simulator scene, these plugins can be attached to the models and provide handy ROS topics and services. On the other hand, MORSE offers packages that can be used for creating ROS subscribers and publishers. The packages *python3-dev*, *python3-yaml*, *python3-setuptools*, *python3-rospkg*, and *python3-catkin-tools* must be installed to connect ROS with MORSE. The main disadvantage of Gazebo in comparison with MORSE is its higher CPU load for larger team sizes. The main programming language for Gazebo is C++ and MORSE strictly requires the use of Python. Gazebo has a total of 190.k documented lines of code while MORSE has just 31.k lines of code. Gazebo uses OGRE for 3D rendering, while MORSE relies on the Blender game engine.

V. Conclusion and Future Work

In this study, we compared in detail two fully open-source 3D simulators for multi-robot systems: MORSE and Gazebo. The Robot Operating System (ROS) was used as a middleware for both simulators. The quantitative analysis focused on CPU and GPU consumption. MORSE and Gazebo almost stalled for a total number of 15 and 10 robots, respectively. Overall, MORSE performed better than Gazebo in the tests conducted. The results represent a step forward in the ongoing efforts within multi-robot systems' simulations. In future work, we will extend our multi-robot patrolling system with advanced

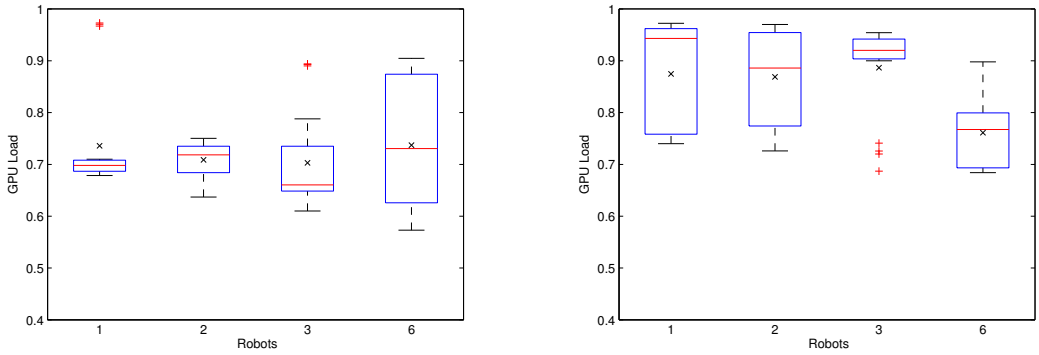


Fig. 5: GPU load required by the simulators for different team sizes: MORSE simulator (left) and Gazebo simulator (right).

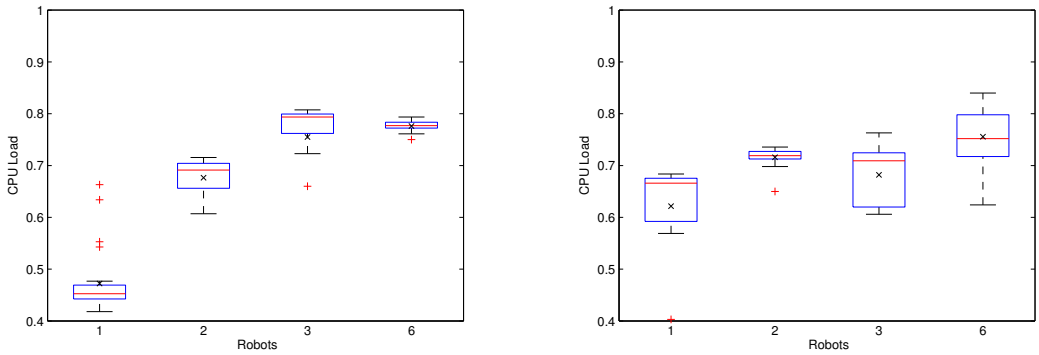


Fig. 6: CPU load required by the simulators for different team sizes: MORSE simulator (left) and Gazebo simulator (right).

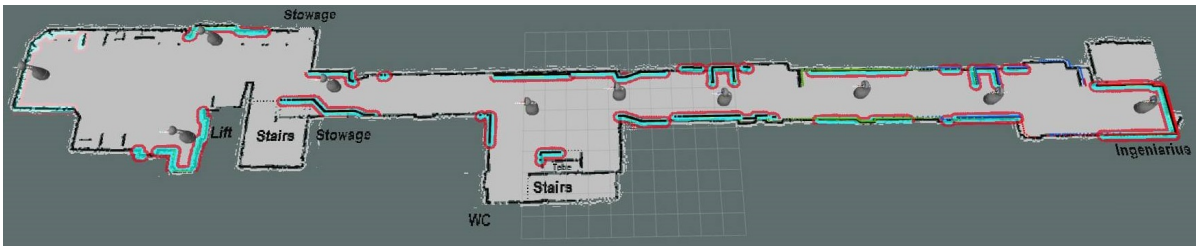


Fig. 7: Visualization of 10 robots patrolling the environment in *rviz*.

artificial perception capabilities for detecting the presence of unauthorized people and other abnormal security situations.

REFERENCES

[1] E. Freund. "On the design of multi-robot systems". In Proc. of IEEE Int. Conf. on Robotics and Automation, vol. 1, pp. 477-490, 1984.

[2] H. Hirukawa and A. Knoll. "Workshop on open source software in robotics". In Proc. of IEEE Int. Conf. on Robotics & Automation, 2009.

[3] M. Quigley, K. Conley, B.P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A.Y. Ng. "ROS: an open-source robot operating system". In Prof. of IEEE Int. Conf. on Robotics and Automation Workshop on Open Source Software in Robotics, 2009.

[4] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan. "Modular open robots simulation engine: MORSE". In Proc. of IEEE Int. Conf. on Robotics and Automation, pp. 46-51, 2011.

[5] N. Koenig and A. Howard. "Design and Use Paradigms for Gazebo, An Open-Source Multi-Robot Simulator". In Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, vol 3, pp 2149-2154, 2004.

[6] R. Vaughan. "Massively multi-robot simulation in Stage". Swarm Intelligence, vol. 2(2), pp. 189-208, 2008.

[7] O. Michel. "Webots: Symbiosis between virtual and real mobile robots". In Proc. of Int. Conf. on Virtual Worlds, pp. 254-263, 1998.

[8] S. Carpin, M. Lewis, J. Wang, S. Balakirsky and C. Scrapper. "USARSim: a robot simulator for research and education". In Proc. of IEEE Int. Conf. on Robotics Automation, pp. 1400-1405, 2007.

[9] Y. Hu, and W. Meng. "ROSUnitySim: Development and experimentation of a real-time simulator for multi-UAV local planning". Simulation, vol. 92(10), pp. 931-944, 2016.

[10] A. E. Gomez, T. C. dos Santos, C. Massera Filho, D. Gomes, J. C. Perafan, and D. F. Wolf. "Simulation platform for cooperative vehicle systems". In Proc. of IEEE 17th Int. Conf. on Intelligent Transportation Systems, pp. 1347-1352, 2014.

[11] D. Portugal, and R. P. Rocha. "Cooperative multi-robot patrol with Bayesian learning". Autonomous Robots, vol. 40(5), pp. 929-953, 2016.

[12] S. Ivaldi, J. Peters, V. Padois, and F. Nori. "Tools for simulating humanoid robot dynamics: A survey based on user feedback". In Proc. of 14th IEEE-RAS Int. Conf. on Humanoids, pp. 842-849, 2014.