World Academy of Science, Engineering and Technology
International Journal of Computer and Systems Engineering
Vol:13, No:4, 2019

Development of Tools for Multi Vehicles Simulation with Robot Operating System and ArduPilot

Pierre Kancir, Jean-Philippe Diguet, Marc Sevaux

***Abstract*—**One of the main difficulties in developing multi-robot systems (MRS) is related to the simulation and testing tools available. Indeed, if the differences between simulations and real robots are too significant, the transition from the simulation to the robot won't be possible without another long development phase and won't permit to validate the simulation. Moreover, the testing of different algorithmic solutions or modifications of robots requires a strong knowledge of current tools and a significant development time. Therefore, the availability of tools for MRS, mainly with flying drones, is crucial to enable the industrial emergence of these systems. This research aims to present the most commonly used tools for MRS simulations and their main shortcomings and presents complementary tools to improve the productivity of designers in the development of multi-vehicle solutions focused on a fast learning curve and rapid transition from simulations to real usage. The proposed contributions are based on existing open source tools as Gazebo simulator combined with ROS (Robot Operating System) and the open-source multi-platform autopilot ArduPilot to bring them to a broad audience.

***Keywords*—**ROS, ArduPilot, MRS, simulation, drones, Gazebo.

I. Introduction

MULTI-ROBOT systems (MRS) are systems composed of several mobile robots. Thanks to their intrinsic robustness and modularity, they can perform complex tasks more efficiently than robots alone [1]. If MRS are studied intensively today [2], [3], there are still very few industrial uses. Indeed, the MRS are complex to analytically model. Secondly the design of hardware and software parts are also tricky [4]. The process of developing, testing and debugging a group of robots is a difficult and time-consuming task given the number and complexity of the entities used. This paper presents a set of new simulation and testing tools for MRS to reduce the complexity of their development. Section II provides an analysis of the main tools used in MRS research. Then Section III presents the autopilot for robots and why they should be used. Finally, Section IV provides details of a new simulation tool.

This works targets small mobile robots, which are robotic entities capable of moving in their environment. This definition includes mobile robots (Unmanned Ground Vehicles or UGVs), flying robots (Unmanned Aerial Vehicles or UAVs), boats, etc., but excludes robotics arms.

P. Kancir is with the Lab-Sticc, Université de Bretagne Sud, UMR CNRS 6285, Lorient, France (e-mail: pierre.kancir@univ-ubs.fr).

J-Ph. Diguet and M. Sevaux are with the Lab-Sticc, Université de Bretagne Sud, UMR CNRS 6285, Lorient, France.

II. Simulation Tools

A. Simulators

The perfect simulator does not yet exist, but there are some that are able to approach current robotic reality in order to demonstrate and compare different solutions to a robotic problem. Among the most famous are the following:

- The Player/Stage suite [5]: It's a 2D open source robotic simulator. It natively supports multi-robot systems with a small sensor database. The suite is also compliant with the ROS (Robot Operating System) [6] middleware to extend its simulation capabilities. However, the models used are aging (no drone, no wireless networks, etc.) and the project is only little maintained to keep compability with ROS.
- Gazebo [7]: It is an open source 3D simulator, only available under Linux. It is the reference for 3D open source robotic simulators.
  - Ability to accurately simulate complex robots such as Atlas or Valkyrie or drones such as the 3DR Iris.
  - Gazebo is flexible. It is possible to simulate a complex robot like a cloud of drones by simply reconfiguring it. In addition, there are a multitude of plug-ins that increase the already present possibilities (sensor, physical model, etc.).
  - Integration with ROS. Robots can be simulated with the real code that will be running in their embedded systems.
  - Simulations performed on Gazebo with ROS can be replayed using logging.
  - A strong community.
- V-Rep [8]: A simulator similar to Gazebo, but not free nor open source. It is therefore a multiplatform 3D simulator. The advantage of V-Rep over Gazebo is its professional support and better learning curve. Nevertheless, the use of V-Rep in combination with real robots can only be done through the use of ROS.
- MORSE [9]: It is a simulator built on a set of Python libraries that uses the Blender software engine to interface a virtual 3D environment with external software. Open source and cross-platform, it allows to simulate a large number of sensors and robots. Its advantage is based on its modularity and its implementation in python which allows you to quickly get to grips with it. Nevertheless, it is less used than Gazebo or V-Rep and lacks of real examples other than the documentation that make its learning curve hard.

According to [10], the complexity of the mission and the environment affect the design of multi-robot systems. Despite the possibility of being able to use "simple" robots in MRS, the increase of the equipment complexity and the computational capabilities of modern robots, has opened the way to new possibilities. However, it has also increased the cost of the design and evaluation processes of these systems. The main parameters of an MRS that the simulator must be able to implement are the following:

- Manage a variable number of robots
- Get access to different types of robots: rolling, flying, flying, etc.
- Represent several characteristics of robots: sizes, weights, sensors, etc.
- Can be used with several levels of realism: perfect position, with positioning disturbance, etc.
- Allow to vary the mission environment: one of the main difficulty of using an MRS relies to its organization in the environment, the simulator must be able to simulate different environments with physical characteristics: building, ground elevation, etc.
- Provide a solution for recording simulation data so that they can be replayed.
- Must be freeware and participatory to maximize reuse and share experiences on MRS.

While the simulators described above are capable of meeting these characteristics, their use remains complex and cumbersome to implement. Indeed, they have a long learning time due to the lack of concrete examples, especially in the field of multi-robot simulations and need high computational power.

B. ROS

Being the reference for robot developments, ROS is also the reference for MRS. While not all work in the field uses it, most of the works proposing experiments are based on ROS. The main reason is to take advantage of eco-system allowing the reuse of the code used in simulation on real platforms and the relative simplicity of robot design. However, ROS also has many disadvantages that do not make it the perfect choice for MRS.

Initially designed to run on Willow Garage's PR2 robot, ROS has spread to other robots and other applications whose uses had not been planned. Despite its many advantages, ROS also has some disadvantages compared to other middleware or software:

- ROS has no real-time capabilities.
- There is no follow-up of detailed source updates (API changes, protocol, etc.).
- Depending on the libraries installed, the storage space required may be large.
- No deployment on very small platforms (AVR, STM32, etc.)
- No consideration of network quality: QoS, packet loss, etc.
- No standard for the multi-robot approach
- Little interaction with other eco-systems: IoT etc. [11]-[13]

Fig. 1 ArduPilot platforms

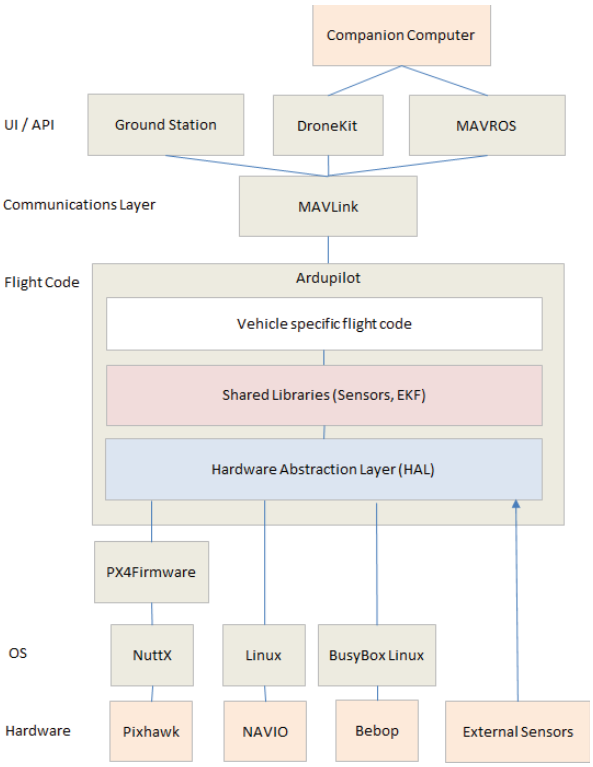

Fig. 2 ArduPilot software diagram

If the modularity and popularity of ROS have made it a reference in robotics, its overly open development also means many problems for more advanced use than prototyping. Indeed, the quality of the different tools offered by ROS is not always the same. Some tools clearly lack testing and verification that leads to operational and reliability problems, which delays development times due to debugging. In addition, many of the proposed contributions, lack quality and optimization for use in real embedded systems (use of experimental software libraries, little protection against memory overruns or divisions by zero, etc.). Initially designed for terrestrial robots, ROS also lacks an interface and standard libraries for other types of robots, especially those who are flying. Finally, it does not offer any library for security actions during problems despite the presence of a self-diagnosis tool,

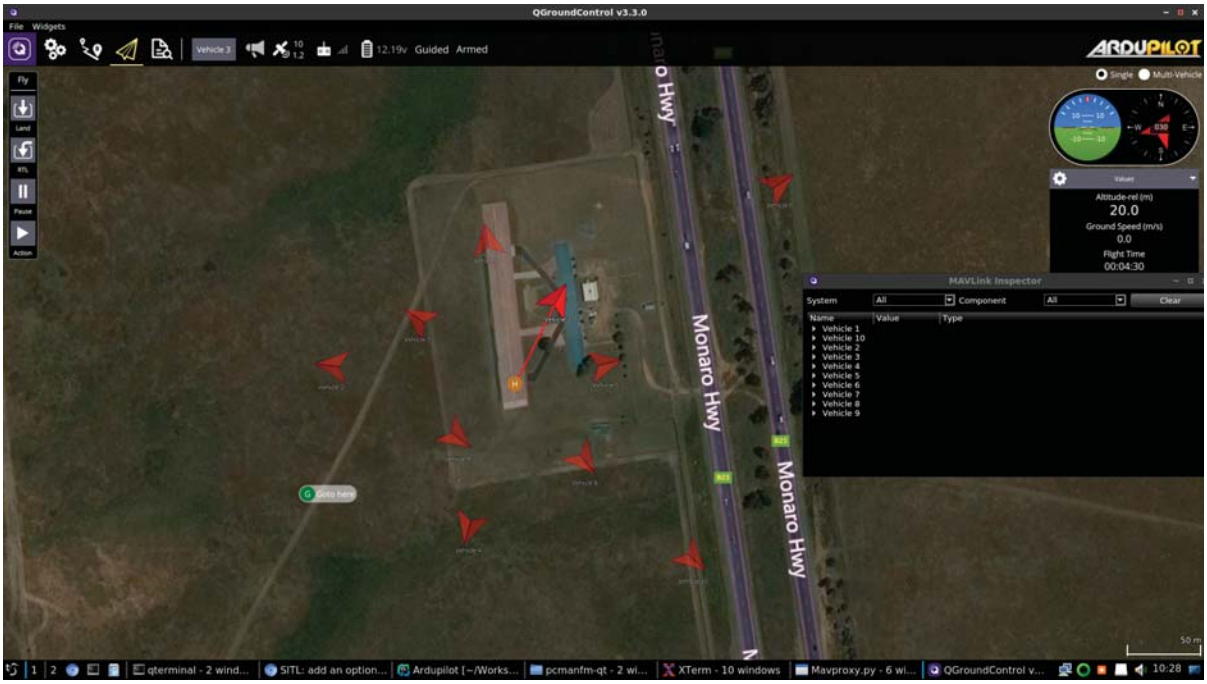

Fig. 3 Multi drone simulation with SITL

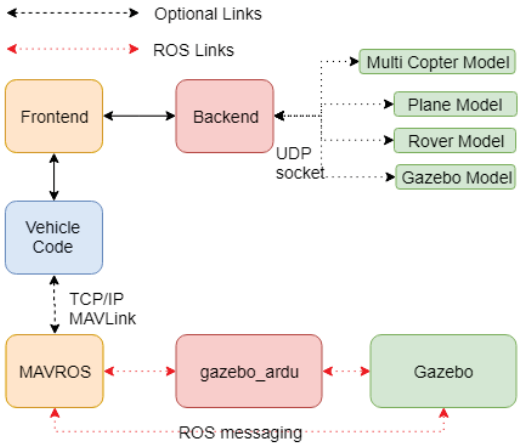

Fig. 4 SITL configurations

a vital component for large numbers of MRS demonstrations. Thus, ROS is an interesting solution for solo robots but lacks generosity for the design of MRS where the quality of robotic platforms is vital for the proper functioning of the group.

III. Autopilot for Robots

Works on autonomous vehicles are numerous today and tests in real conditions have already begun all over the world. Because of the use of these vehicles with people nearby, it is important to multiply the actual tests in as many situations as possible. While ROS has become a reference in the field of robotics, there is still no generic robotics platform at affordable prices (i.e. < 10000€). This is an important issue since the new work on MRS will have to perform complex and time-consuming platform development tasks at the expense of work on more academic issues (trajectory generation, swarm control, etc.). A solution to this problem comes from flying drones. Indeed, the current versions have more and more capabilities thanks to the autopilots that control them. The use of flying drones requires a reliable and proven flight controller (both hardware and software) capable of managing the low-level functions of the drone and safety (engine and sensor control, navigation, etc.), things that must be developed and tested on ROS. The development of a complete autopilot requires advanced knowledge of electronics and software in addition to knowledge of the type of robot required (flying, driving, etc.) and a long and costly development time. It is so preferable to start from an existing solution such as ArduPilot [14], PX4 [15], DJI [16], PaparazziUAV [17] for the most famous ones. The platform proposed in this article was built on the basis of ArduPilot because it is the only one capable of being used on any type of vehicle: driving, flying, surface vehicles, etc., something requiring strong development on ROS or Gazebo. While this autopilot is well known in the field of flying drones, its use in groups, with terrestrial robots or in MRS simulation is much less well known. It has the advantage of being low cost, open source and has a strong community and industrial partners and simply interfaces with ROS.

Unlike ROS, ArduPilot is not a middleware for robotics but an autopilot for micro autonomous vehicles. It therefore supports the 3 main robot types: ground, flying, and surface vehicles. The software architecture differs significantly from that of ROS since it is based on the model of embedded computing architectures that take into account limited resource issues, real time constraints and reliability requirements. The initial development of ArduPilot was done by amateurs and has evolved into a worldwide autopilot recognized for its reliability by both academic, industrial and government actors

Fig. 5 Leader-Follower simulation with Gazebo and SITL

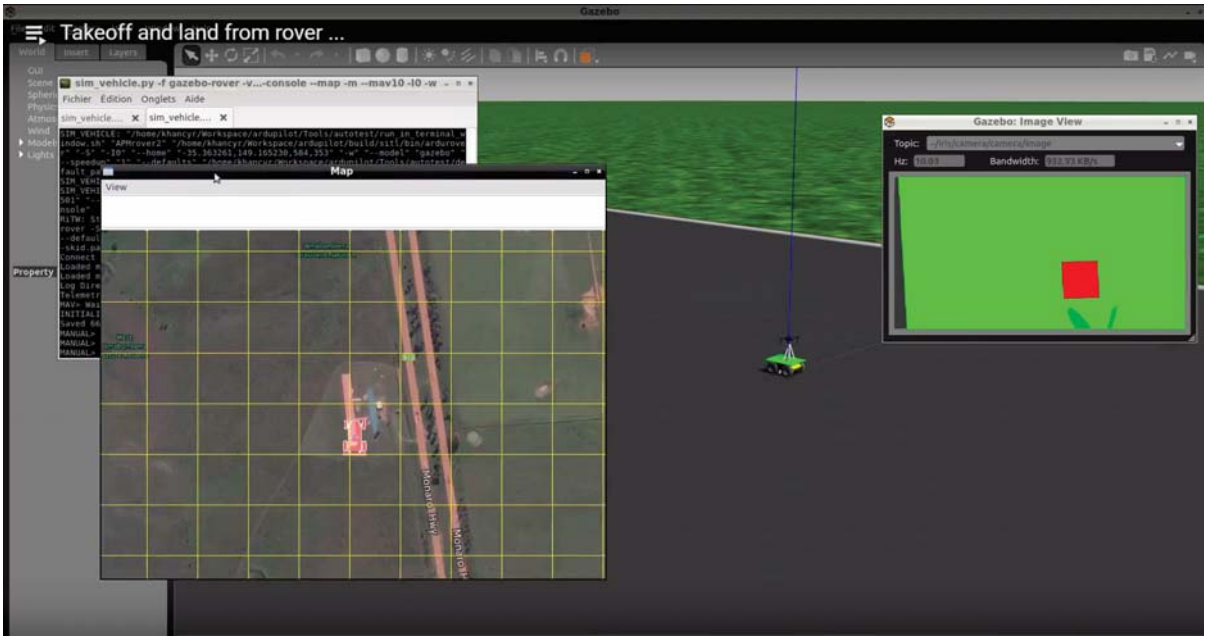

Fig. 6 Quadcopter doing precision landing on a rover in Gazebo with ArduPilot

while keeping a strong community of enthusiast and active developpers. Finally, the ArduPilot autopilot platform can be used with ROS thanks to numerous libraries, including Mavros [18]. Indeed, ArduPilot is currently only capable to understand the MAVLink protocol [19]. It is a light and open source protocol, based on binarized data structures that can be used by systems with low resources and communication links with low bandwidth. It is also based on a publish-subcribe and point-to-point mechanism. MAVLink is designed to be used with low throughput data streams: drone position, altitude, etc., transmitted in multicast. The objective is to transmit the information to those who can receive it but without any guarantee of receipt. Point to point being reserved instead for communications requiring acquiescence and a guarantee of delivery such as configurations, emergency orders, etc. On the other side, ROS is only capable to use ROS protocol. Therefore, we need some applications, like Mavros, to make a translation between MAVLink and ROS. Finally, it allows data encryption, message addressing and message re-routing by address, allowing both data security and long-distance data transmission, both of which can be important for MRS. With many standard messages and industrial support, this protocol

has established itself as a reference in autopilots for UAVs but also at the level of UGV since the recent Turtlebot 3 based on ROS uses it.

IV. New Simulation Tools

A. ArduPilot SITL

While the simulators described above are capable of meeting these characteristics from Section II, their use remains complex and cumbersome to implement. Indeed, they have a long learning time due to the lack of concrete examples, especially in the field of multi-robot simulations and need for high computational power. This is why a simpler simulation is generally used at the expense of realism. In order to overcome this defect, a new simulation solution was implemented with several levels of realism and simpler usage for simple tests (swarm group movement, etc.). This new simulation is based on the SITL (Software In The Loop) simulator of the ArduPilot project. SITL is able to simulate the different supported vehicles and a set of sensors. The simulation of the drones is complete in the sense that the simulated code is the same as the one that will be embedded in the real platforms.

SITL is based on simple vehicle models (weight, dimensions, etc.) and simulated sensors. The architecture is divided into 3 parts: the vehicle, the frontend and the backend. The simulated vehicle part uses the complete ArduPilot code and simulated sensor data corresponding to those of the supported platforms: IMU, GPS, barometer, etc. Thanks to the sensor data, the code performs the vehicle's stabilization, navigation, avoidance, etc., tasks and returns the calculated actuator outputs (motors, LEDs, etc.) to the *frontend*. The *frontend* part provides the autopilot with the simulated data from the *backend*, and returns to the *backend* the autopilot outputs that will be used to update the models in the simulation. This part is responsible for managing the simulation clock. The simulation clock is separated from the clock of the hardware environment. This allows the simulated time to be accelerated or stopped for debugging, for example. The *backend* part defines the simulated environment with the technical characteristics of the vehicles and sensors (weight, dimension). Each vehicle has a set of characteristics and several default vehicles corresponding to the most used vehicles are proposed. At each simulated time step, the *frontend* updates the *backend* which sends the sensor data to the autopilot and receives the data from the actuators that will update the simulation. The separation into *frontend* and *backend* of the simulator is effective to allow the use of different *backends*. While the default *backend* remains simple, it allows you to quickly simulate robots with a simple set of sensors but does not have a collision model other than with the ground.

B. Contributions

1) SITL Improvements: In order to be able to offer both simple and complex simulations, several improvements have been added to SITL in order to turn it from a single to a multiple-robot system simulator. The aim is to be able to quickly and simply simulate groups of vehicles with a light simulator learning curve but without falling into the trap of graphs or point simulations in a matrix. But also to allow advanced simulation levels or combined with reality. An important SITL rewrite work has been undertaken to allow easier addition of *backend* and sensor from the different *backend* and a simulation launch interface to simply create multiple instances of SITL with vehicles configured to run in groups.

Fig. 3 presents a view from GCS (Ground Control Station) that command a swarm of 10 quadcopters simulated in SITL. Each drone is doing the planned tasks that were assigned by the GCS as a real will do and is able to communicate with other drones only by through the GCS, that act as the central communication node in this simulation.

The use of other *backend* allows to increase the models and simulation levels. Thus it is now possible to connect SITL to Gazebo as a simulator in order to benefit from a collision model and a more complete simulation environment (obstacles, mechanical constraints, etc.). This *backend* is presented in Section IV-B2.

The use of SITL allows simple but fast MRS simulations with the simple addition of collective behaviour or coordination mechanisms either directly from the autopilot code, as in Figure 5 or from an external controller or application like ROS. The basic simulation is simplified since the vehicle models used are basic: simple sensor and actuator models, no collision systems, no mechanical problems. But they realistically simulate the movements and behaviour of vehicles since the entire autopilot code is used. This low-resource simulation (1-2% of a 2017 i5 processor for a vehicle instance) allows rapid progress in validating collective behaviors with a simple learning of how to use the platform and associated security concepts such as battery management or EKF (*Extended Kalman Filter*) positioning issues.

2) SITL 3D Plotting with Gazebo: In order to extend the possibilities of using SITL, a simple "plotting" plugin has been developed in Gazebo [20]. This plugin allows the designers to load a 3D model of a vehicle and external sensors (Lidar, sonar, camera, etc.) into Gazebo and move the model with position and orientation data from SITL.

Thus Gazebo serves as a simulator for sensors and 3D engines for robots whose simulated data comes from SITL. This architecture is the same as that used in drones using autopilots: the autopilot manages the low level parts and another controller (usually a microcomputer) manages the high level tasks with ROS (sensor requiring Lidar 360 type treatments, path planning, etc.). For this purpose, ROS is used. A Gazebo plugin could have been developed but it was more useful and simpler to use ROS here. Indeed, thanks to the close bounding between Gazebo and ROS, all the sensors simulated in Gazebo have natively an interface with ROS and thus with ArduPilot via Mavros. Thus it is possible to combine the three software packages, to make fast simulations with advanced sensor simulations and simple obstacle and collision management at the cost of a loss of realism. Due to the use of Gazebo, this simulation is heavier than the one provided by SITL alone. However, it provides advanced collision and sensor management without the management of advanced

kinematic and dynamic models usually used with Gazebo that are difficult to implement. Looking at Gazebo model for the pioneer2dx robot [21], we can see the complexity. In fact, tasks like calculating all inertial matrix, and friction between components are not simple to do. This type of simulation is used to test localization or avoidance strategies that will later be optimized to be embedded without the disadvantages of overly complete simulation. Fig. 5 shows a visualization of a leader-follower behavior performed directly in ArduPilot, the framed drone is the leader who sends his current position and velocity vector to the two followers who must maintain a constant distance and a given position with the leader. This simulation is simpler to do than the full Gazebo version. In fact, there are few parameterization to be done on Gazebo side to only visualize the vehicle position, whereas a full simulation would include some tuning of the collision and lift-drag system. The SITL configuration consists only on launching 3 instances of quadcopter on SITL. In the configuration part, this type of simulation is faster than doing a full Gazebo simulation, but it is also less consuming in term of computation power as the model and physics involved in Gazebo are simplified. A standard laptop with i5 processor is able to play it with simulation time accelerated more than 5 times.

3) ArduPilot Plugin for Gazebo: Finally, a new Gazebo plugin has been created [22]. This time, it allows Gazebo to be used as a SITL backend in order to take advantage of a collision model and a more complete simulation environment (obstacles, mechanical constraints, etc.). That is to say, all data provided to SITL comes from the models loaded into Gazebo. This allows you to take advantage of Gazebo's advanced sensor models directly in the autopilot. It is no longer necessary to use ROS as a gateway. Indeed, even if ArduPilot is not as complete as ROS in terms of robotics algorithms (path planning, sensor interface, etc.), it is sufficiently complete to allow autonomous travel (navigation on the route by GPS waypoints and simple obstacle avoidance). This new Gazebo plugin does not include ArduPilot in this one but allows a direct connection with SITL which works on all types of vehicles and supports multi-robots configurations. This work is the basis of the simulations carried out for the Service Academies Swarm Challenge Live-Fly Competition [23] where two groups of 40 UAVs (20 flying wings and 20 quadrotors) compete in a flag capture match. Fig. 6 shows the simulation of a rover and quadcopter under ArduPilot in Gazebo. Thanks to the camera data from Gazebo, the quadcopter is able to track the rover and land accurately on it during movement. The code for precision landing is available in ArduPilot [24], Gazebo here allows to check its proper functioning with the addition of wind and different speeds of movement of the rover.

V. Conclusion

This article first provides a review of the main tools used for MRS simulations with their use and drawbacks. It also introduces new software solutions as well as new simulation methods to simplify MRS developments, especially those that are heterogeneous, i.e. with several different vehicle types. These new tools have been made available to the scientific community as open source and royalty-free software. We maintain that this platform, which is compatible with ROS, but which meets operating constraints closer to the solutions expected by manufacturers, will provide a generic framework for even more advanced MRS solutions [25]. In future work, we would like to complete the integration of the Gazebo plugin in the next versions of the simulator in order to make it even easier for the entire ROS community to benefit from its possibilities. On the ArduPilot autopilot side, new work will make it possible to simulate even more sensors natively in SITL and improve interoperability with ROS.

References

[1] Z. Yan, L. Fabresse, J. Laval, and N. Bouraqadi, "Metrics for performance benchmarking of multi-robot exploration," *IEEE International Conference on Intelligent Robots and Systems*, vol. 2015-Decem, pp. 3407–3414, 2015.

[2] G. Francesca and M. Birattari, "Automatic design of robot swarms: achievements and challenges," *Frontiers in Robotics and AI*, vol. 3, p. 29, 2016.

[3] D. Portugal and R. P. Rocha, "Distributed multi-robot patrol: A scalable and fault-tolerant framework," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1572–1587, 2013.

[4] E. Şahin and A. Winfield, "Special issue on swarm robotics," *Swarm Intelligence*, vol. 2, no. 2-4, pp. 69–72, 2008.

[5] B. P. Gerkey, R. Vaughan, and A. Howard, "The player/stage project: Tools for multi-robot and distributed sensor systems," 08 2003.

[6] M. Quigley, K. Conley, B. Gerkey, J. FAust, T. Foote, J. Leibs, E. Berger, R. Wheeler, and A. Mg, "ROS: an open-source Robot Operating System," *Icra*, vol. 3, no. Figure 1, p. 5, 2009.

[7] N. Koenig and a. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 3, pp. 2149–2154, 2004.

[8] C. Robotics, "V-rep," 2018. (Online). Available: "http://www.coppeliarobotics.com/".

[9] G. Echeverria, N. Lassabe, A. Degroote, and S. Lemaignan, "Modular openrobots simulation engine: Morse," in *Proceedings of the IEEE ICRA*, 2011.

[10] L. E. L. Parker, "Current research in multirobot systems," *Artificial Life and Robotics*, pp. 1–5, 2003. (Online). Available: http://link.springer.com/article/10.1007/BF02480877.

[11] R. Arumugam, V. R. Enti, L. Bingbing, W. Xiaojun, K. Baskaran, F. F. Kong, A. S. Kumar, K. D. Meng, and G. W. Kit, "DAvinCi: A cloud computing framework for service robots," *2010 IEEE International Conference on Robotics and Automation, ICRA 2010*, pp. 3084–3089, may 2010. (Online). Available: http://www.scopus.com/inward/record.url?eid=2-s2.0-77955831617{&}partnerID=40{&}md5=475cec250602b92879b9751beb70f40b.

[12] D. Hunziker, M. Gajamohan, M. Waibel, and R. D'Andrea, "Rapyuta: The RoboEarth cloud engine," *Proceedings - IEEE International Conference on Robotics and Automation*, pp. 438–444, 2013.

[13] R. Doriya, P. Chakraborty, and G. C. Nandi, "'Robot-Cloud': A framework to assist heterogeneous low cost robots," *Proceedings - 2012 International Conference on Communication, Information and Computing Technology, ICCICT 2012*, pp. 1–5, oct 2012. (Online). Available: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6398208.

[14] Ardupilot, "Ardupilot," 2018. (Online). Available: http://ardupilot.org/ardupilot/index.html.

[15] L. Meier, D. Honegger, and M. Pollefeys, "PX4: A node-based multithreaded open source robotics framework for deeply embedded platforms," *2015 IEEE International Conference on Robotics and Automation*, pp. 6235–6240, 2015. (Online). Available: http://www.inf.ethz.ch/personal/lomeier/publications/px4{_}autopilot{_}icra2015.pdf.

[16] DJI, "DJI," 2017. (Online). Available: http://www.dji.com/fr.

[17] PaparazziUAV, "PaparazziUAV," 2018. (Online). Available: https://github.com/paparazzi/paparazzi.

[18] Mavros, "Mavros," 2018. (Online). Available: https://github.com/mavlink/mavros.

[19] MAVLink, "MAVLink," 2017. (Online). Available: https://mavlink.io/en/.

[20] P. Kancir, "gazebo_ardu," 2017. (Online). Available: https://github.com/khancyr/gazebo_ardu.
[21] OSRF, "Comparison of single-board computers," 2018. (Online). Available: "https://bitbucket.org/osrf/gazebo_models/src/9533d55593096e7ebdfb539e99d2bf9cb1bff347/pioneer2dx/model.sdf?at=default&fileviewer=file-view-default".
[22] P. Kancir, "ardupilot_gazebo," 2017. (Online). Available: https://github.com/khancyr/ardupilot_gazebo.
[23] T. H. Chung, M. R. Clement, M. A. Day, K. D. Jones, D. Davis, and M. Jones, "Live-fly, large-scale field experimentation for large numbers of fixed-wing uavs," in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, May 2016, pp. 1255–1262.
[24] ArduPilot, "Ardupilot precision landing," 2018. (Online). Available: "http://ardupilot.org/copter/docs/precision-landing-with-irlock.html".
[25] A. Sinisterra, M. Dhanak, and N. Kouvaras, "A usv platform for surface autonomy," in *OCEANS 2017 - Anchorage*, Sept 2017, pp. 1–8.