Integrating ROS and Android for Rescuers in a Cloud Robotics Architecture: Application to a Casualty Evacuation Exercise

Manuel Toscano-Moreno, Juan Bravo-Arrabal, Manuel Sánchez-Montero, Javier Serón Barba, Ricardo Vázquez-Martín, J.J. Fernandez-Lozano, Anthony Mandow, and Alfonso Garcia-Cerezo

***Abstract*— Cloud robotics and the Internet of robotic things (IoRT) can boost the performance of human-robot cooperative teams in demanding environments (e.g., disaster response, mining, demolition, and nuclear sites) by allowing timely information sharing between agents on the field (both human and robotic) and the mission control center. In previous works, we defined an Edge/Cloud-based IoRT and communications architecture for heterogeneous multi-agent systems that was applied to search and rescue missions (SAR-IoCA). In this paper, we address the integration of a remote mission control center, which performs path planning, teleoperation and mission supervision, into a ROS network. Furthermore, we present the UMA-ROS-Android app, which allows publishing smartphone sensor data, including audio and high definition images from the rear camera, and can be used by responders for requesting a robot to the control center from a geolocalized field position. The app works up to API 32 and has been shared for the ROS community. The paper offers a case study where the proposed framework was applied to a cooperative casualty evacuation mission with professional responders and an unmanned rover with two detachable stretchers in a high-fidelity exercise performed in Malaga (Spain) in June 2022.**

Keywords: Cloud Robotics; Disaster Robotics; Distributed Robot Systems; ROS; Android; Search and Rescue

I. Introduction

Cloud robotics and the Internet of robotic things (IoRT) can boost the performance of human-robot cooperative teams in demanding applications like disaster response [1], [2], agriculture [3] and nuclear surveillance [4], where robots need to develop their tasks while communicating among themselves and with their human operators [5]. However, no consensus has been reached on the tools and technologies for cloud robotics and use cases remain crucial [6].

The widely accepted Robot Operating System (ROS) [7] can serve as the basis for new frameworks for cloud robotics [8]. Thus, existing debugging and visualization tools for ROS, such as *RViz*, have been useful for providing situational awareness for teleoperation interfaces using streamed data from search and rescue (SAR) robots [9]. Moreover, the ROSLink custom protocol [10] aimed to integrate ROS based robots with other devices through the Internet of things (IoT). More recently, [11] proposed another open framework for ROS-based cloud integration that can be used over public Internet.

*This work has been partially funded by the Spanish Ministerio de Ciencia, Innovación y Universidades, Gobierno de España, projects RTI2018-093421-B-I00 and PID2021-122944OB-I00.

The authors are with Robotics and Mechatronics Group, Institute of Mechatronics Engineering and Cyber-physical Systems, University of Malaga, Spain.

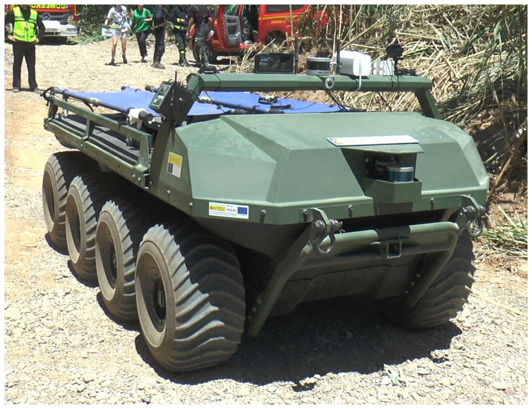

Fig. 1: All terrain Rover J8 robot with the adapted detachable onboard stretchers, two GPS and antennas, two lidars, two 180° field-of-view cameras, a 5G smartphone and a 5G router.

In this context, smartphones are becoming a powerful IoRT enabling technology [12], providing not only built-in sensing and processing, but also communications for cloud and edge computing. Experimental client libraries such as *rosjava* add ROS support for Android smartphone applications like embedding a smartphone in mobile robots [13], [14] and controlling robots with live video stream [15]. Furthermore, smartphone apps can provide effective teleoperation human interfaces for robots running on ROS [16]. In fact, a user-friendly smartphone human-robot interaction interface can improve mission performance and responder acceptability in long-distance collaborative search [17]. The ROS-Mobile Android application [18] is an alternative to *RViz* for mobile devices, offering control and visualization features.

In previous works we presented the Internet of cooperative agents (X-IoCA) architecture [2], a generic cloud robotics framework for the effective integration of heterogeneous sensor networks and robots, multi-edge computing centers (MECCs), and 5G communications in field applications with cooperative human and robot teams. In particular, we presented an implementation for SAR (SAR-IoCA) which was not fully integrated with ROS. In [19], we explored the application of SAR-IoCA with a remote mission control center through commercial networks.

In this paper, we address the full integration of the feedback information system (FIS), which performs path planning, teleoperation and mission supervision, into a ROS

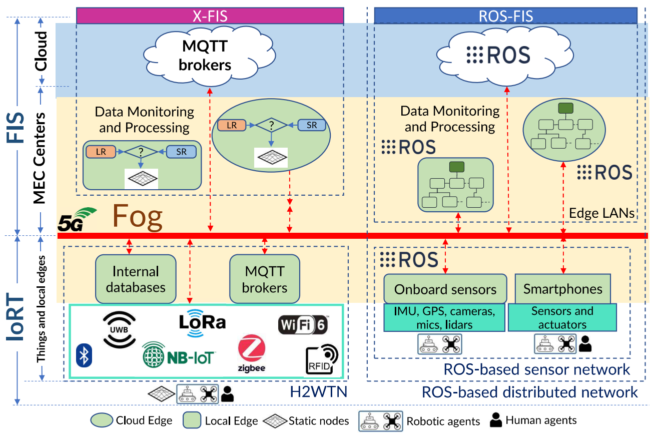

Fig. 2: The X-IoCA architecture

network. Furthermore, we present an update of UMA-ROS-Android app [20], which allows publishing smartphone sensors, including audio and high definition images from the rear camera, and can be used by responders for requesting a robot to the control center from a geolocated field position. The app has been developed for current Android versions (API 32) and has been shared for the ROS community [21]. We have implemented and tested the evolved SAR-IoCA framework for a cooperative casualty evacuation mission with professional responders and an unmanned rover with two detachable stretchers (see Fig.1) in a high-fidelity exercise performed in Malaga (Spain) in June 2022. The paper discusses lessons learned from this use case.

The rest of the paper is organized as follows: Sect. II gives a brief review of the X-IoCA architecture, Sect. III presents the evolution of the architecture integrating ROS and Android, Sect. IV is dedicated to the experiments designed to validate the new characteristics of the SAR-IoCA architecture. Finally, Sect. V is devoted to the concluding remarks.

II. Review of the X-IoCA Architecture

This section briefly reviews the generic Internet of cooperative agents (X-IoCA) architecture (see Fig. 2) [2], where agents are entities that carry or wear end-devices. Agents can be humans, different types of robots (e.g., UGVs and UAVs), vehicles, and even sensorized animals.

The X-IoCA architecture comprises sensing and computing elements, and is divided into two subsystems: an IoRT, which is the source of sensor data, and a feedback information system (FIS) for data monitoring, storing, and processing. The MECC consist of computing devices (e.g., PCs or smartphones) that can be used within both the IoRT or the FIS. Regarding to communications, each MECC can have one or more user-equipment (UE) devices (e.g., a 5G CPE or a 5G smartphone). Local edges (green rounded rectangles) correspond to local hosts in the operation area, while cloud edges (green ellipsoids) are remote MECCs. The MECC in the FIS are made up of an X-FIS and a ROS-FIS, each dedicated to a specific type of information flow coming

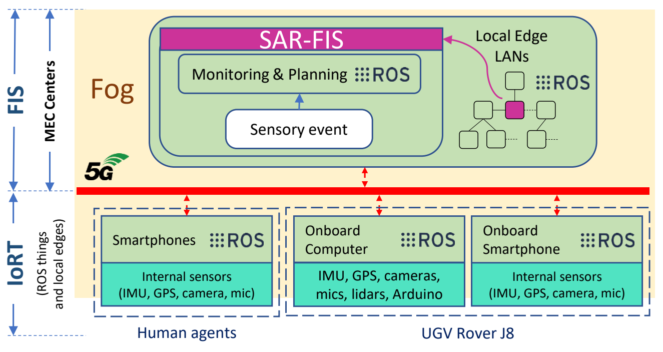
FIS
MEC Centers
IoRT
(ROS things and local edges)
Fog
SAR-FIS
Monitoring & Planning ROS
Sensory event
Local Edge LANs
ROS
5G
Smartphones ROS
Internal sensors (IMU, GPS, camera, mic)
Onboard Computer ROS
IMU, GPS, cameras, mics, lidars, Arduino
Onboard Smartphone ROS
Internal sensors (IMU, GPS, camera, mic)
Human agents
UGV Rover J8

Fig. 3: An implementation of the new SAR-IoCA system.

from the two parts of the IoRT: a hybrid and heterogeneous network of wireless transceivers (H2WTN), composed of radio-capable (LoRa, BLE, UWB, WiFi, etc.) sensor-nodes covering short and long ranges (SR and LR, respectively), and a ROS-based distributed network. Similarly, the local edges within the IoRT can manage the H2WTN and the ROS network. Thus, the information is not necessarily at the physical place where the data are acquired, and a given agent can use a resource more or less distant from it, which entails complying or not with a series of requirements [22].

III. Integration of ROS and Android in the SAR-IoCA Architecture

III-A. *The new SAR-IoCA architecture*

This section presents the evolution of the SAR-IoCA architecture, where the FIS has been fully integrated in a ROS architecture for SAR for a more efficient use of resources such as: smartphones and lidars included as sensor-nodes in the IoRT, UGV path-planning systems, and MECCs. The cloud has been omitted since non-ROS elements (e.g., H2WTN and Message Queing Telemetry Transport –MQTT– brokers) within the IoRT have not been considered for this paper (see Fig. 3). The most relevant element of the FIS is the SAR-FIS application, an *ad-hoc* MATLAB software for monitoring and processing the sensory information that includes a strategic global path planner for SAR missions.

The 5G communications are represented by a thick red line, to which all the UEs are connected. Each UE has a SIM card with a static and public IP address (SP-IPa). This ensures that communications between UEs, and thus between the MECCs and the IoRT, are point-to-point, without network address translation (NAT) issues.

There are two requirements for any PC within the architecture: bidirectionality, i.e., the possibility to publish information from any LAN and to subscribe to information in the WAN; and fluidity (symmetrical bitrates) in the existing communications in the Fog-distributed ROS network, i.e., what a node publications reaches the subscriber node with the same bandwidth. In the case of computers hosted in MECCs, it is necessary to establish port forwarding rules to ensure that ROS nodes hosted on all the PCs of the local edge LANs can play the double role of subscriber and publisher with the rest of the WAN.

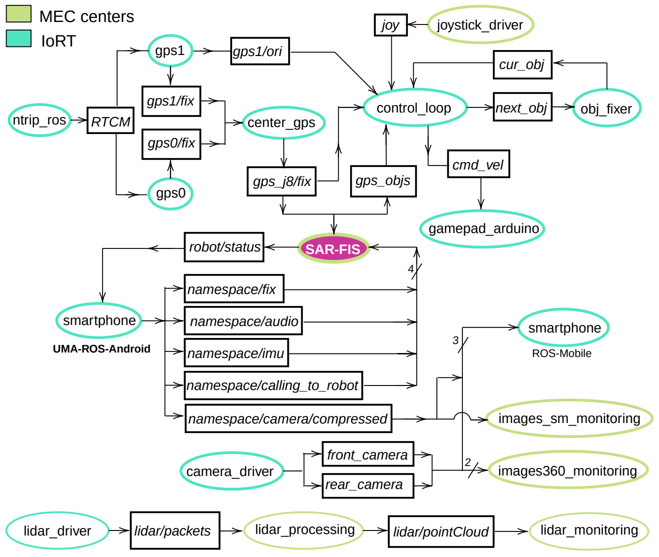

Fig. 4: ROS nodes distributed through the Fog.

Each device in any LAN of a MECC must register the SP-IPa that the rest of LANs within the Fog use to access to the Internet. Only one PC in each LAN will be able to identify itself with the SP-IPa associated with its CPE, so that only it can be seen from the outside (WAN).

For the PC onboard the UGV Rover J8 (hosting ROS nodes in the IoRT), a Demilitarized Zone (DMZ) has been established for its local IP address. In this way, all the messages published by ROS nodes hosted on the robot can flow out to the WAN. In the case of smartphones, there is no two-way problem, so ROS nodes created by the UMA-ROS-Android app can subscribe and publish through the Fog.

With regard to fluidity, it is necessary to define which ROS nodes require information and from where, and to take into account the bandwidth occupied by the messages published in each ROS topic. An excessive number of subscriptions may cause saturation in the throughput capacity of UEs.

Fig. 4 shows how the ROS nodes (ellipsoids) have been distributed between the IoRT and the MECCs, publishing and obtaining information via ROS topics (rectangles). The ROS master node can be hosted by any PC within the Fog. The nodes in the IoRT include several nodes publishing sensor-related data (camera_driver, smartphone, gps0, gps1, ntrip_ros, center_gps). UGV motion control is performed through a main control node (control_loop) and three other nodes devoted to reading and publishing location and velocity objectives (obj_fixer, joystick_driver and padsim_arduino). In the MECCs, the main node (SAR-FIS, in purple in Fig. 4) runs inside the SAR-FIS application in MATLAB, monitoring the agents' positions and generating a list of waypoints. Three other nodes read inputs from a joystick, cameras, and lidar (joystick_driver, images360_monitoring, images_sm_monitoring and lidar_monitoring). A detailed description of the nodes can be found at [21].

III-B. *The UMA Cloud Robotics app for ROS and Android*

This section presents the design (frontend) and functionality (backend) of the developed UMA-ROS-Android app, compatible with the current Android version (API 32). The main goal is to integrate a smartphone as a sensor-node, by publishing information, via ROS, from and about the SAR agent carrying it. The interaction between the SAR agent and the UMA-ROS-Android app must be user-friendly and agile (see Fig. 5), leading to a double purpose:

- To expand the ROS network, adding new sensor-nodes to the IoRT ecosystem, either through a LAN or via WAN, by means of smartphones.
- To acquire information from the agents' surroundings, and encourage cooperation between them.

III-B.1. Frontend: The app has two screens for the user interface: the setup and the connection activities. A dark, high contrast colour gamut has been chosen for proper outdoor viewing and so that devices with OLED displays can benefit from reduced battery consumption. Two user inputs are requested: the ROS master socket (local or public IP address and port) and the SAR ID, which is associated with the namespace of the ROS topics where the desired information will be published. This can be done by selecting the switches related to each of the smartphone's internal sensors (camera, microphone, IMU and GPS). Finally, the connect button activates the smartphone in the ROS network, being all the topics visible in any host of the architecture: Cloud, MECCs and IoRT.

The second screen is the connection activity, where the user can select one or more switches in order to call one or more of the robots (UGVs) available in the IoRT. The list of UGVs presented is static, and includes three UGVs from our Robotics and Mechatronics Group: Cuadriga, Rambler and Rover J8. All of them can be called from the app, thanks to the integration of SAR-FIS in the ROS network. Fig. 5 shows how the agent Garcia has performed a request for the Rover J8 to be planned to the location of the smartphone, which is taken by SAR-FIS in the ROS topic */Garcia/fix*. Finally, the bottom of the connection activity shows what information is transmitted from this smartphone and through which ROS topics.

III-B.2. Backend: The functionality of the application is structured in 10 classes written in Java, eight of which include the construction of the ROS nodes needed to exchange information with the rest of ROS nodes in the SAR-IoCA architecture. The information published includes the rear camera of the smartphone (*sensor_msgs/CompressedImage*), the GPS and IMU data (*sensor_msgs/NavSatFix* and *sensor_msgs/IMU*) and audio (*.WAV*). Three classes publish a Bool message associated with the user's choice of the robot, which triggers the path planning for the selected UGV (*calling_to_robot*, where *robot* is the name of the UGV.) The two other Java classes are associated with the two activities that make up the frontend, so that the user can switch from one to the other using objects. A detailed descriptions of the classes and their functionalities can be found at [21].

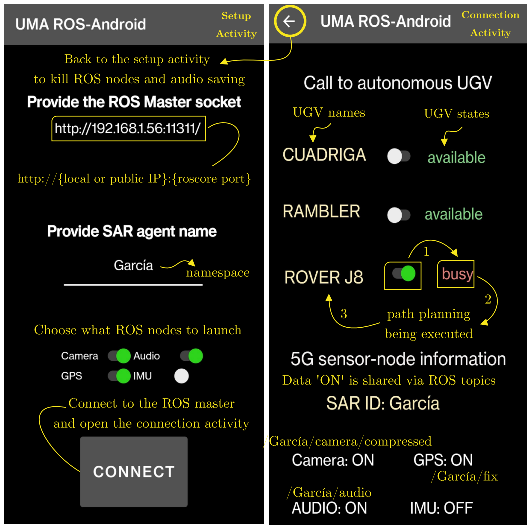
UMA ROS-Android
Setup Activity
Back to the setup activity to kill ROS nodes and audio saving
Provide the ROS Master socket
http://192.168.1.56:11311/
http://{local or public IP}:{roscore port}
Provide SAR agent name
García
namespace
Choose what ROS nodes to launch
Camera
Audio
GPS
IMU
Connect to the ROS master and open the connection activity
CONNECT
UMA ROS-Android
Connection Activity
Call to autonomous UGV
UGV names
UGV states
CUADRIGA
available
RAMBLER
available
ROVER J8
busy
1
2
3
path planning being executed
5G sensor-node information
Data 'ON' is shared via ROS topics
SAR ID: García
/García/camera/compressed
Camera: ON
GPS: ON
/García/fix
/García/audio
AUDIO: ON
IMU: OFF

Fig. 5: UMA-ROS-Android frontend: setup (left) and connection activities (right). Comments are shown in yellow.

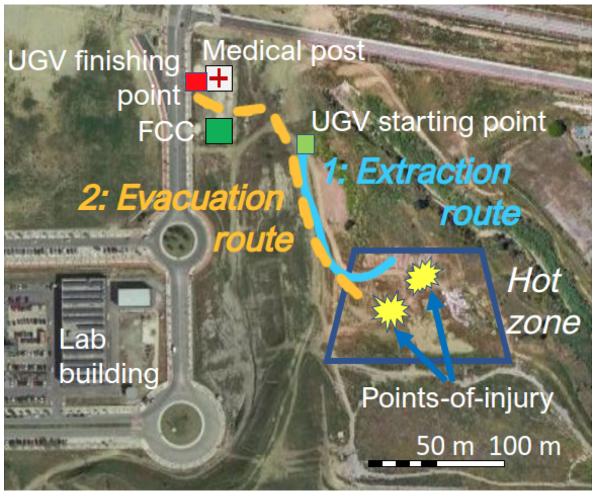

Fig. 6: Casualty extraction and evacuation scenario layout over an aerial view of the disaster response exercise site [23].

III-C. *Integration of a path planner*

SAR-FIS has been designed and implemented by the UMA Robotics and Mechatronics Group for the global planning of UGV paths over natural terrain and the monitoring of sensory devices deployed over the environment, e.g., IP cameras, LoRa devices and/or heterogeneous sensors distributed in a ROS network. SAR-FIS integrates a strategic global planner, i.e., a global path planning system for a set of UGVs towards a set of target points with different priorities. Therefore, the scope of SAR-FIS exceeds the aim proposed in this work, which is to highlight the SAR-IoCA capability to integrate a UGV path planner through a ROS network by request from a human agent. Consequently, neither the description of the path planning algorithm nor all the functionalities of SAR-FIS are within the scope of this paper.

The area where the agents are deployed is defined by a digital elevation model (DEM) and an orthophoto or zenith aerial photograph. The global path planner uses the DEM to estimate the terrain slopes and obtain a feasible path. Each agent publishes the current geolocation coming from GPS devices by a single ROS topic. SAR-FIS uses points of interest (PoI) as target positions for the path planning. A PoI is triggered by geo-referenced events (e.g., a call for a robot from a smartphone app).

Two alternative navigation modes have been defined for Rover J8: (1) planned navigation to a PoI triggered by a call for a robot from a smartphone; and (2) teleoperation.

- **Planned navigation**: A smartphone running the application UMA-ROS-Android (Fig. 5) allows a human agent to request a UGV. SAR-FIS receives the GPS position and request through subscriber ROS nodes associated to the respective topics. Upon detecting a change of planning request of a particular robot, SAR-FIS enables (if the request is on) or disables (otherwise) that agent for path planning. If enabled, SAR-FIS also creates a temporary PoI associated with the current location of the smartphone. Therefore, once the robot and the PoI are defined and enabled for planning, a list of waypoints is found from the current robot position to the PoI location. The resulting path is conditioned by the terrain slopes in the DEM as well as the pitch and roll thresholds admissible by the UGV. The resulting path is stored as an array of UTM coordinates. This array is sent through the ROS network by a ROS publisher associated to the corresponding topic. Finally, the ROS node control_loop running in Rover J8 subscribes to the GPS waypoint list and performs a path-following algorithm using the position and orientation data obtained from its dual GPS-receiver setup.
- **Tele-operation**: For navigation in extremely complex and dangerous environments, the SAR-IoCA implementation is compatible with teleoperation from the control center. Through the ROS network, a human operator can send velocity commands from a joystick, and get the visual feedback from the onboard cameras.

IV. Application to Victim Evacuation Exercise

IV-A. *Brief description of the Exercise*

The casualty extraction and evacuation scenario was part of an annual Workshop organized by the Chair of Security, Emergencies and Disasters at the University of Malaga (UMA) held on June 3, 2022. The robotic stretcher was tested in a cooperative training exercise with a combat medical unit of the Spanish Army (Tercio “Alejandro Farnesio” 4^o of the Spanish Legion).

An aerial view of the exercise area with a layout of the casualty extraction and evacuation robot mission is shown in Fig. 6. The initial location of the UGV is close to the forward control center (FCC). This is the starting point for the path planning of the UGV (extraction route) waiting for the call from the rescuers in the hot zone, where a terrorist attack is simulated. The finishing point of the UGV mission (evacuation route) is near the medical post tent.

The exercise site was a dedicated $90\,000\,\mathrm{m}^2$ outdoor experimental area within the UMA campus. This natural terrain zone was set up as a simulated disaster site, including rubble mounds, crushed vehicles, and tunnels [24]. This outdoor area is an unstructured natural environment with different altitudes. For instance, the points-of-injury are about eight meters above the medical post.

The objectives of this exercise were twofold: the aim of our research group was to test vehicle path-planning and smartphone calling in real scenarios, whilst the military rescue team wanted to test and get acquainted with the use of robotic vehicles in search and rescue missions, in the context of a program for testing new technologies (Force 2035).

IV-B. *Vehicle setup and hardware*

Rover J8 (see Fig. 1), developed by Argo (Kitchener, Ontario, Canada), is an electric off-road 8×8 UGV designed for outdoor navigation and driven by a dual electric motor (main and steer) working with a 46V battery that can last up to $6\,\mathrm{h}$. This UGV drives at a maximum speed of $30\,\mathrm{km\,h^{-1}}$, with a weight of $1090\,\mathrm{kg}$ and a maximum payload of $567\,\mathrm{kg}$. The vehicle includes a *follow-me mode* in which the robot can follow a person autonomously. A button in the front of the robot activates this feature.

Some hardware elements were added to the Rover J8 in order to integrate it in the described SAR-IoCA architecture. As the robot does not allow to run user made programs, a NUC8i7BEH with Ubuntu 18.04 and ROS Melodic has been installed to implement the corresponding software to the Rover J8 described in the Section III. To get the position and orientation to perform the path-following task, two GPS receivers connected to the NUC through different Universal Serial Bus (USB) ports were added, both of them getting differential correction data via the standard Networked Transport of RTCM via Internet Protocol (NTRIP) through regional public positioning networks. Rover J8 can be controlled through a Logitech Gamepad controller, directly plugging it to the robot's main PC. To be able to send velocity commands to the robot and controlling it in teleoperation and path-following mode, an Arduino Leonardo with the Xinput library [25] has been connected to the Rover's main computer to replicate the Gamepad controller. Lastly, a 5G router model HUAWEI Router 5G CPE Pro 2 H122-373 is used for wireless communication through Internet.

We used commercial stretchers with two wheels on the front end and two legs on the back, as seen in Fig. 7. According to the space available for the payload and the capacity of the UGV, two stretchers are used to transport up to two victims in each run. A reliable locking mechanism for the stretchers is needed to transport victims safely. Thus, we adapted the same locking mechanism from a previous work [26] to accommodate both the front wheels and the leg ends of the stretchers (see Fig. 7).

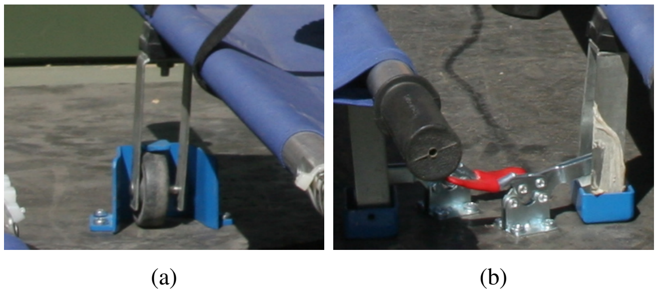

Fig. 7: Stretcher fixing details: a) front end wheel brackets and b) rear manual locking mechanism.

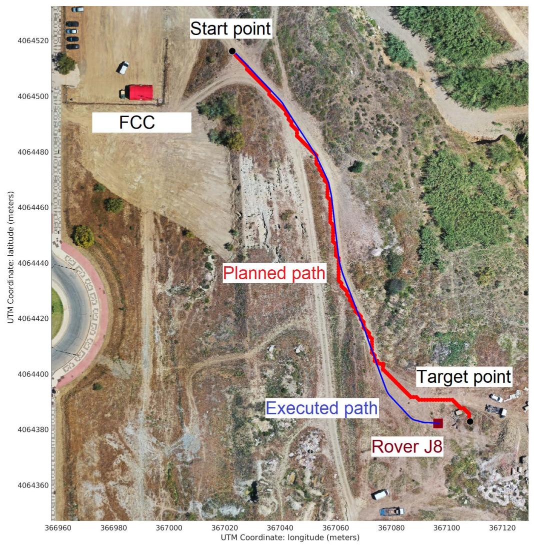

Fig. 8: Planned (red) and executed (blue) paths for Rover J8. The planned path was triggered by a human agent request for Rover J8.

IV-C. *Description of the robot mission*

The current configuration using the Rover J8 was based in a previous experience presented in [26], and it was first tested during the Workshop on October 29, 2020, but exclusively by members of our team. Rover J8 was used by actual rescue teams during the Workshop held on June 18, 2021, in a similar scenario, and an exercise conducted in the headquarters of Tercio "Alejandro Farnesio" 4° of the Spanish Army (Ronda, Malaga) on August 8, 2021.

Although the combat medical unit has participated with us three times in the last year, the members of the rescue team in this mission were completely new, including the non-commissioned officer (NCO) in charge of the operation. However, the officers planning the mission had knowledge and experience from the previous collaborations, and they were aware of the capabilities of our modified Rover J8 platform and the way it can be useful in real scenarios.

The exercise was performed under realistic conditions in a one-shot basis. Before the exercise, the rescue team was briefed about the robotic stretcher, the smartphone app to call the UGV for assistance and the *follow-me mode* of the vehicle. A demonstration of the Rover J8 path-planning to the

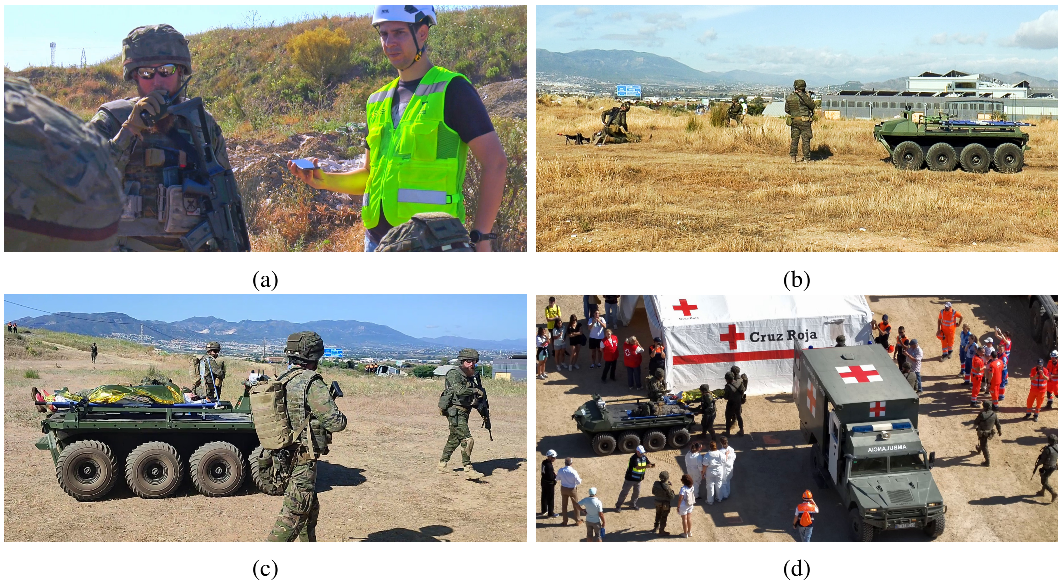

Fig. 9: Snapshots of the casualty extraction and evacuation exercise.

PoI was performed before the exercise, with the aim to show the autonomous capabilities of the robot. The autonomous demonstration trajectory was configured at a the moderate speed of $5\,\mathrm{km\,h^{-1}}$ due to safety. However, the rescue team required a higher speed in order to attend the call more quickly. Therefore, in the actual mission, the assistance to the smartphone call (the extraction route) was not autonomous but teleoperated at a speed of $10\,\mathrm{km\,h^{-1}}$. Although the extraction route was teleoperated, SAR-FIS succeeded in producing timely and effective path. In fact, the teleoperated path in the actual experiment only differs from the planned path in the final part (see Fig. 8).

Fig. 9 shows a sequence of representative moments in the rescue mission:

a) After finding the victims in the points-of-injury, the rescue team called for assistance using the smartphone.
b) Rover J8 started the teleoperated extraction route to the GPS location of the smartphone, reaching the PoI. In each point of extraction the rescuers remove the stretchers from the robot and secured the patient with straps. Victims were played by volunteer drama students simulating casualties during the attack.
c) The rescuers communicated the situation with the FCC: six casualties found, two of them successfully locked in the stretchers. The UGV in *follow-me mode* returned with casualties escorted by the rescue team.
d) The evacuation route finished at the medical post, where victims were cared for, and finally transported by an ambulance to the helicopter evacuation area.

IV-D. *Results, lessons learned, and feedback from end users*

The mission lasted approximately 25 minutes, from the beginning of the terrorist attack to the care of victims at the medical post. The evacuation and extraction of victims were successful, on both sides (research group and rescue team) but we noted some lessons learnt for future work:

- The stretcher locking mechanism is appropriate for locking and removing victims in the UGV. There were some issues in previous experiences, but they were solved with an appropriate briefing to the rescue team.
- The extraction route, using path-planning in response to the call of the rescue team was successful, but it was not used at the time of the mission since the rescue team considered the speed of the UGV inappropriate to respond to a critical situation with casualties. The UGV speed should be increased in planned navigation mode for this kind of missions.
- The rescue team leader had to use the *follow-me mode* to move the vehicle between points-of-injuries, but it was not to deactivated when the team was loading the stretchers, producing unexpected movements that may put the team and the victims under risk. The *follow-me mode* must be improved to avoid these situations, for instance including a form of automatic deactivation.
- The evacuation route was completed using the *follow-me mode* due to the operational doctrine of the rescue team: no injured personnel is left without protection. This doctrine may be different for other rescue teams, or other missions. Adaptability of the navigation modes of the UGV is key for their effective integration with rescue teams.
- The idea of requesting a UGV using a smartphone was successful, and considered useful by the rescue team. One of the goals of the exercise was to test the feasibility of the approach from the point of view of the first responders. Once the feedback is positive, the usability of the app should be specifically designed for this kind of environments.

V. Conclusions

This paper presents an implementation of the distributed X-IoCA architecture for SAR mission, integrating a remote control center, which performs path planning, teleoperation and mission supervision, into a ROS network. A specifically designed app, UMA-ROS-Android, allows publishing smartphone sensors, including audio and high definition images from the rear camera, and can be used by first responders for requesting a robot to the control center from a geolocalized field position. The app works up to API 32 and has been shared for the ROS community. The proposed framework was tested in a cooperative casualty evacuation mission with professional first responders and an unmanned rover with two detachable stretchers in a high-fidelity exercise performed in Malaga (Spain) in June 2022. The experiments showed the feasibility of the approach, allowing a rescue team to request a UGV for victim extraction, and helping the team to complete its mission. The feedback from the users was very positive regarding the role of UGVs for this kind of mission, and helped to define future improvements. Besides the lines of work mentioned in the lessons learnt, the use of this approach for logistic missions was also suggested. In this sense, the simultaneous use of several UGVs to respond to concurrent request is already possible in SAR-FIS, planning the paths of the robots under different criteria. But field experiments need to be conducted. Finally, migration of the architecture to ROS2 may improve the communications among the nodes in the fog, as well as their safety.

ACKNOWLEDGEMENTS

The authors want to thank the collaboration of the Chair for Safety, Emergencies and Disasters of the University of Malaga, led by Prof. Jesús Miranda-Páez for organizing the exercises. We are also grateful to Tercio "Alejandro Farnesio" 4^o of the Spanish Legion, for testing the robot and providing valuable user feedback. Finally, we wish to acknowledge the support from our colleagues of the UMA Robotics and Mechatronics Group.

REFERENCES

[1] A. Botta, J. Cacace, R. De Vivo, B. Siciliano, and G. Ventre, "Networking for cloud robotics: The DewROS platform and its application," *Journal of Sensor and Actuator Networks*, vol. 10, no. 2, p. 34, 2021.

[2] J. Bravo-Arrabal, M. Toscano-Moreno, J. Fernandez-Lozano, A. Mandow, J. A. Gomez-Ruiz, and A. García-Cerezo, "The internet of cooperative agents architecture (X-IoCA) for robots, hybrid sensor networks, and MEC centers in complex environments: A search and rescue case study," *Sensors*, vol. 21, no. 23, p. 7843, 2021.

[3] P. Gonzalez-De-Santos, R. Fernández, D. Sepúlveda, E. Navas, L. Emmi, and M. Armada, "Field robots for intelligent farms–inhering features from industry," *Agronomy*, vol. 10, no. 11, 2020.

[4] M. Di Castro, M. Ferre, and A. Masi, "CERNTAURO: A modular architecture for robotic inspection and telemanipulation in harsh and semi-structured environments," *IEEE Access*, vol. 6, pp. 37 506 – 37 522, 2018.

[5] D. Tardioli, L. Riazuelo, D. Sicignano, C. Rizzo, F. Lera, J. L. Villarroel, and L. Montano, "Ground robotics in tunnels: Keys and lessons learned after 10 years of research and experiments," *Journal of Field Robotics*, vol. 36, no. 6, pp. 1074 – 1101, 2019.

[6] V. Dawarka and G. Bekaroo, "Building and evaluating cloud robotic systems: A systematic review," *Robotics and Computer-Integrated Manufacturing*, vol. 73, 2022.

[7] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," in *IEEE ICRA Workshop on Open Source Software*, vol. 3, Kobe, Japan, 2009, pp. 1–6.

[8] G. Toffetti and T. M. Bohnert, "Cloud robotics with ROS," in *Robot Operating System (ROS): The Complete Reference (Volume 4)*, 2020, pp. 119–146.

[9] F. Perez-Grau, R. Ragel, F. Caballero, A. Viguria, and A. Ollero, "Semi-autonomous teleoperation of UAVs in search and rescue scenarios," in *International Conference on Unmanned Aircraft Systems*, 2017, pp. 1066 – 1074.

[10] A. Koubaa, M. Alajlan, and B. Qureshi, "ROSLink: Bridging ROS with the Internet-of-Things for cloud robotics."

[11] R. C. Mello, S. D. Sierra M., W. M. Scheidegger, M. C. Múnera, C. A. Cifuentes, M. R. Ribeiro, and A. Frizera-Neto, "The PoundCloud framework for ROS-based cloud robotics: Case studies on autonomous navigation and human-robot interaction," *Robotics and Autonomous Systems*, vol. 150, 2022.

[12] A. Jiménez-González, J. R. Martinez-De Dios, and A. Ollero, "Testbeds for ubiquitous robotics: A survey," *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1487 – 1501, 2013.

[13] J. De A Barbosa, F. Do P De C Lima, L. Dos S Coutinho, J. R Rodrigues Leite, J. Barbosa Machado, C. Henrique Valerio, and G. Sousa Bastos, "Ros, android and cloud robotics: How to make a powerful low cost robot," in *International Conference on Advanced Robotics*, 2015, pp. 158–163.

[14] H. M. Do, C. J. Mouser, Y. Gu, W. Sheng, S. Honarvar, and T. Chen, "An open platform telepresence robot with natural human interface," in *2013 IEEE International Conference on Cyber Technology in Automation, Control and Intelligent Systems*, 2013, pp. 81–86.

[15] B. Gökcen, F. Baygül, F. Çakmak, E. Uslu, M. F. Amasyalı, and S. Yavuz, "Android application for simultaneously control of multiple land robots which have different drive strategy," in *2017 International Conference on Computer Science and Engineering (UBMK)*, 2017, pp. 724–728.

[16] E. Szymanska, L. Petrovic, I. Markovic, and I. Petrovic, "Mobile robot teleoperation via Android mobile device with UDP communication," in *International Convention on Information, Communication and Electronic Technology*, 2021, pp. 1143 –1148.

[17] J. Dominguez-Vidal, I. J. Torres-Rodriguez, A. Garrell, and A. Sanfeliu, "User-friendly smartphone interface to share knowledge in human-robot collaborative search tasks," in *IEEE Int. Conf. on Robot and Human Interactive Communication*, 2021, pp. 913 – 918.

[18] N. Rottmann, N. Studt, F. Ernst, and E. Rueckert, "ROS-Mobile: An Android application for the Robot Operating System," *arXiv*, 2020.

[19] M. Sánchez-Montero, M. Toscano-Moreno, J. Bravo-Arrabal, J. S. Barba, P. Vera-Ortega, R. Vázquez-Martín, J. Fernandez-Lozano, A. Mandow, and A. García-Cerezo, "Remote planning and operation of a UGV through ROS and commercial mobile networks," in *Fifth Iberian Robotics Conference*, 2022, p. Submitted to.

[20] G. Ruiz Mudarra, J. Bravo-Arrabal, J. Fernández-Lozano, and A. García-Cerezo, "Integración de smartphones 5G en redes de sensores distribuidas para robótica de exteriores mediante ROS y Android," in *Jornadas de Robótica, Educación y Biongeniería del Comité Español de Automática*, 2022, pp. 91–99.

[21] G. Ruiz-Mudarra, J. Bravo-Arrabal, and J. J. Fernández-Lozano, "UMA-ROS-Android repository," accessed: Jul 6, 2022. [Online]. Available: https://github.com/jjflozano/uma-ros-android

[22] A. Yousefpour, C. Fung, T. Nguyen, K. Kadiyala, F. Jalali, A. Niakanlahiji, J. Kong, and J. P. Jue, "All one needs to know about fog computing and related edge computing paradigms: A complete survey," *Journal of Systems Architecture*, vol. 98, pp. 289–330, 2019.

[23] Google, "Aerial view of the UMA search and rescue experimental area," accessed: Jul 6, 2022. [Online]. Available: https://goo.gl/maps/EC2v2y1LtbRvBu4M7

[24] J. Morales, R. Vázquez-Martín, A. Mandow, D. Morilla-Cabello, and A. García-Cerezo, "The UMA-SAR Dataset: Multimodal data collection from a ground vehicle during outdoor disaster response training exercises," *The International Journal of Robotics Research*, vol. 40, no. 6-7, pp. 835–847, 2021.

[25] "Xinput: Library for emulating an Xbox controller over USB," 2022, accessed: Jul 6, 2022. [Online]. Available: https://www.arduino.cc/reference/en/libraries/xinput/

[26] A. Mandow, J. Seron, F. Pastor, and A. Garcia-Cerezo, "Experimental validation of a robotic stretcher for casualty evacuation in a man-made disaster exercise," *2020 IEEE International Symposium on Safety, Security, and Rescue Robotics, SSRR 2020*, pp. 241–245, 11 2020.