IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received 4 December 2024, accepted 20 December 2024, date of publication 16 January 2025, date of current version 27 January 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3530391

APPLIED RESEARCH

An Open-Source Multi-Robot Framework System for Collaborative Environments Based on ROS2

FRANCISCO YUMBLA[1], (Senior Member, IEEE), MARCELO FAJARDO-PRUNA[1], (Member, IEEE), ANTHONNY PIGUAVE[1], DIEGO RONQUILLO[1], RICARDO ORTIZ[2], JONGSEONG BRAD CHOI[2,3], (Member, IEEE), GABRIEL DÍAZ[4], (Senior Member, IEEE), XABIEL G. PAÑEDA[5], AND HYUNGPIL MOON[6], (Member, IEEE)

[1]Facultad de Ingeniería en Mecánica y Ciencias de la Producción, Escuela Superior Politécnica del Litoral (ESPOL), Campus Gustavo Galindo, Guayaquil 090902, Ecuador
[2]Department of Mechanical Engineering, The State University of New York (SUNY Korea), Incheon 21985, South Korea
[3]Department of Mechanical Engineering, Stony Brook University, The State University of New York, Stony Brook, NY 11794, USA
[4]Electrical, Electronics and Control Engineering Department, Spanish University for Distance Education (UNED), 28040 Madrid, Spain
[5]Departamento de Informática, Ingeniería Telemática, Universidad de Oviedo, 33007 Oviedo, Spain
[6]Department of Mechanical Engineering, Sungkyunkwan University, Jangan-gu, Suwon-si, Gyeonggi-do 16419, South Korea

Corresponding author: Francisco Yumbla (fryumbla@espol.edu.ec)

This work was supported by the Robotics, Automation and Mechatronics Engineering Laboratory (RAMEL), Ecuador.

ABSTRACT Despite the rise of robotics and automation in industrial applications, the widespread adoption of collaborative robotics still needs to be improved due to the lack of interoperability between robots and the low adaptability of existing systems. Solving this problem would mean a significant advance in robotics and industrial automation. Under this context, an open-source MultiRobot Framework based on ROS2 was developed in the present research to effectively communicate and coordinate robotics agents and sensors in closed collaborative environments. A simulation-based control and software design was performed using the GAZEBO tool. A centralized architecture was obtained with an autonomous navigation module for the planning and robot routes monitoring, a computer vision module for the location and management of uncertainties, and a task controller module to assign mobilization mission objects. In conclusion, using ROS2 to communicate and coordinate various mechatronic systems effectively results in a robust, flexible, and scalable solution critical to industrial processes.

INDEX TERMS Collaborative robotics, interoperability, artificial vision, autonomous navigation, scalability.

I. INTRODUCTION

In recent years, significant advances have been witnessed in robotics and automation. Robots have become flexible collaborators in various environments, with great adaptability. Robots actively collaborate with humans in industrial applications, medical services, precision agriculture, and many other areas to develop required missions. Today, we can find robots performing multiple tasks beyond industrial robotics. Lately, with the rise of autonomous vehicles, it can be seen that the current development of robotics is mobility with the development of vehicles and robots such as the Boston Dynamics Spot [1], a quadruped that can mobilize in various types of space, the bear robotics robot servers [2], an educational and research platform such as the TurtleBot 4 [3] that Roomba robot uses it as a base; and finally, there is the company Starship Technologies that developed a delivery robot that can move over long distances within a city to make home deliveries [4].

One of the main bases of collaborative robotics, being one of the most significant advances, is the Robot Operating System (ROS). ROS supports an open-source framework for developing packages to interact with real sensors, actuators, and robots, among others, using a publisher/subscriber

The associate editor coordinating the review of this manuscript and approving it for publication was Martin Reisslein.

system for communication between them. ROS provides the best option to develop important algorithms in robotics and is the one that is used worldwide today. The main advantage of this pseudo-operating system is that it is easy to use, develops new packages, communicates with existing packages, and provides the tools required to access information from sensors easily, process it, and be able to generate an adequate "response" to the robot. The significant number of users and developers in the ROS community makes this software more interesting. This infrastructure is applied primarily to robots, but an autonomous vehicle can be interpreted as another type of robot; therefore, the same programs can also control them [5]. This software framework will be a milestone in robotics research in the coming years [6].

It has been proven that implementing a Multi-Robot System (MRS) is more profitable than constructing a robot with multiple necessary functionalities [7]. Integrating several robots gives the system greater robustness, a broader coverage capacity, and efficiency. From participation in industrial processes, search teams, or delivery services in hospitals, restaurants, and warehouses, the possibilities of applications of systems made up of multiple robots are abundant. Even missions or tasks that currently can be executed by a single robot with several functionalities can benefit from an MRS since the robustness and reliability can be increased by combining several robots with lower robustness and reliability [8].

MRSs are an emerging area that is under ongoing research. However, the deployment, monitoring, control, and collection of a group of robots represent a greater challenge than an individual robot. In addition, the operator faces more difficulty controlling the system [9], and it is also important to ensure safe interaction between these robots and people.

Implementing an MRS in a collaborative environment improves efficiency in fulfilling required missions. Also, assigning missions with greater complexity makes it possible since there are more resources. Another advantage is its robustness. If one of the robots or sensors has a failure, the task can be completed thanks to the rest of the robots or sensors. This flexibility is an essential and required factor in industries. Finally, the constant exchange of information between the systems and equipment of an MRS extends coverage. This functionality is relevant in applications for exploring and surveilling large spaces.

In summary, implementing an MRS allows one to take advantage of the potential of the robotics systems belonging to a user. The efficiency, the robustness, and the coverage of the assigned mission will be improved. Designing a framework for MRS using ROS2 is a relevant challenge in robotics since it involves localization, communication, and autonomous navigation of several robots, as shown in Figure 1. The paper is organized as follows. Section II describes the Motivation. Section III details our Multi-Robot Framework System. At the same time, the experiments carried out in simulation and real cases using the proposed Framework System are described in Section IV. Finally, we present the conclusions and future scopes in Section V.

II. MOTIVATION

In recent years, a recurring R&D topic has been the study of robotic resource management. In [10], it is indicated that many companies worldwide, especially those dedicated to manufacturing, have proceeded to reconfigure their production systems to implement machines, products, and collaborative infrastructure, sharing data and information to provide the system with flexibility and robustness, increase productivity by automating the process, and obtaining a system with interconnected equipment and processes and in constant communication not only with other machines but also with the operators.

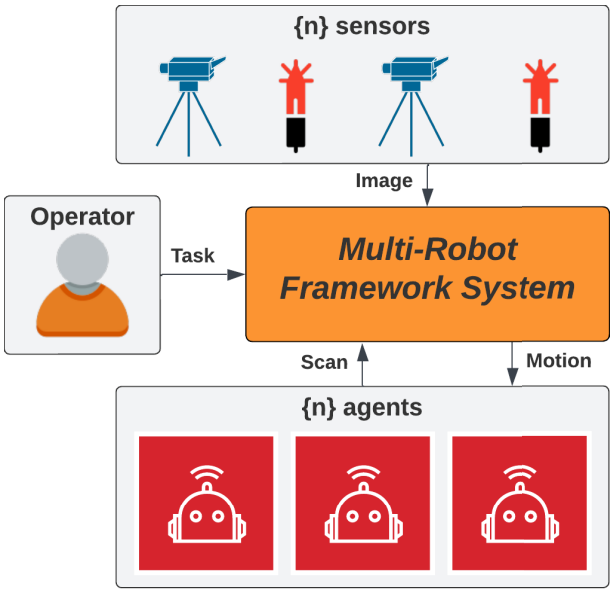


FIGURE 1. Open-Source multi-robot framework system. General diagram of the operation, interaction and communication between the MultiRobot framework with the robotic agents that execute tasks, the operator who indicates the tasks, and the sensors that allow understanding and collecting information from the environment.

According to Sachon [10], there are five fundamental pillars to consider: Data generation and capture, data analysis, man-machine interaction, flexible production, and intellectual property. This new paradigm of robotic resource management has caused the so-called "fourth industrial revolution," where companies seek to incorporate new technologies to improve their productivity. At the same time, research and development centers work in robotics, big data, and artificial intelligence to offer solutions to the industrial sector, seeking to facilitate the implementation of this industry model based on the pillars described.

From this idea, a key requirement for efficient use and management of robotic resources is cooperation between them, a concept from which the Multi-Robot System concept is born, where multiple robots perform activities in a

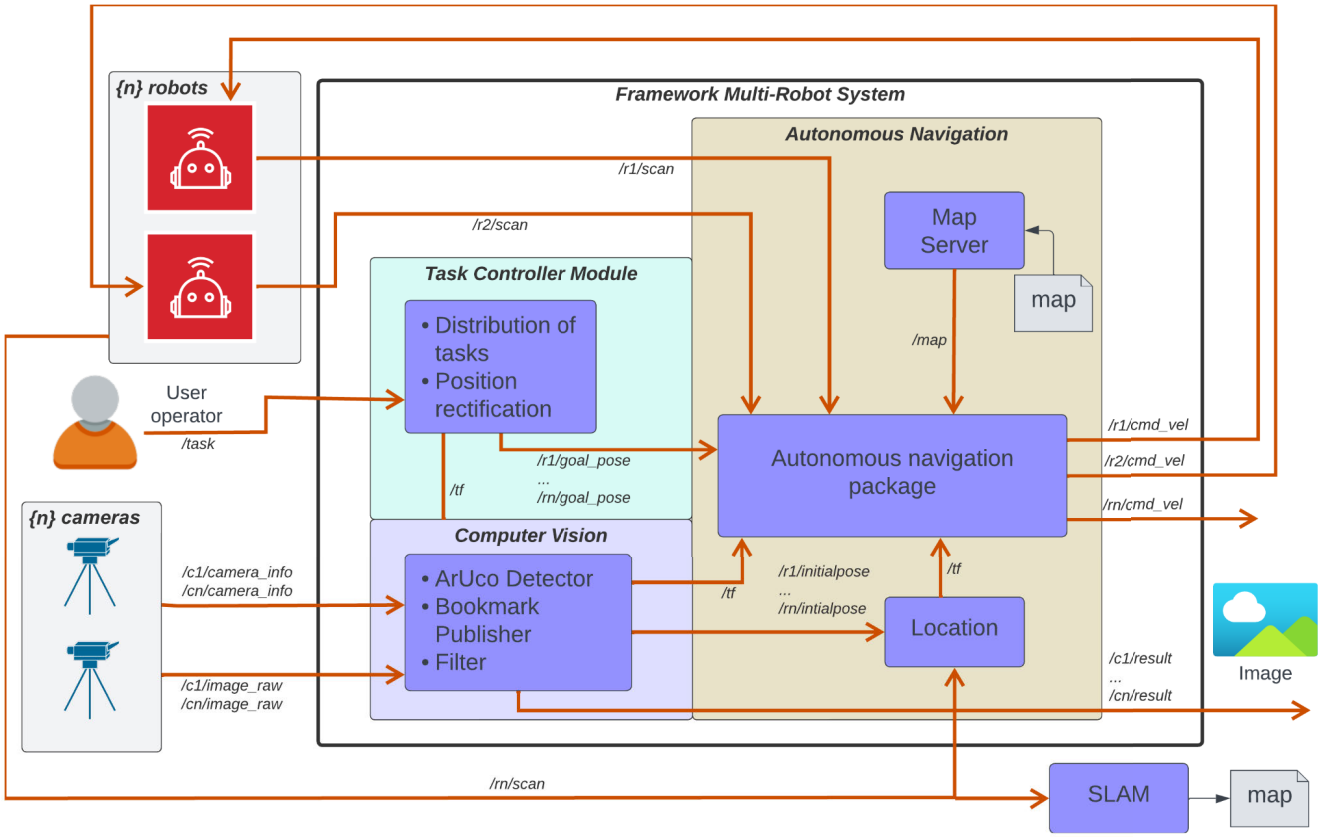


FIGURE 2. Framework arquitecture including task controller module, computer vision, and autonomous navigation. Detail of the modules that make up the proposed MultiRobot framework: Task controller module, autonomous navigation and computer vision, indicating the processes that each one performs and the interaction and communication between them and with the environment.

coordinated manner, such as locomotion and manipulation for a task or mission that a single robot could not achieve [11], providing flexibility in the operations and tasks to be performed, as well as the generation and exchange of information.

The use of MRSs today has spread to different areas beyond research. Consequently, there are applications in agricultural solutions, logistics, maintenance, search, and rescue, among many other scenarios in which it is sought to take advantage of the capabilities of robots to improve the effectiveness and efficiency of procedures by automating them. However, there are various challenges [9] that these systems still have to overcome and for which they have a boom in research topics, being the object of study in different investigations related to task planning, control architectures, and integration of new technologies such as Virtual Reality.

Next, different works related to MRS will be discussed, focusing on the organizational model of the agents that influences their communication, and on the challenges that must be overcome during the implementation of the system.

According to Lima and Custodio [12], the most relevant challenges that are the subject of research in MRSs are summarized in the uncertainty in the values obtained from the sensors and in the result as such of the actions carried out by the agents, the added complexity (in matters of reasoning, planning, task assignment, programming, control and learning) due to the need of robots to cooperate and coordinate actions, the quality of communication that tends to worsen as the number of members of the work fleet increases, and finally, the integration of different technologies that manage the subsystems of each robot.

Related to the first challenge described by Lima and Custodio, there are different investigations related to designing an optimal control system for an MRS and managing uncertainties. In this way, Zhang et al., in their article Decentralized Control of Multi-Robot Systems in Cooperative Object Transportation Using Deep Reinforcement Learning [13], show their results by implementing deep Q-network (DQN) decentralized controllers in an MRS to improve the cooperation of robots in transport tasks and the robustness of the system. On the other hand, Shule et al. [14], managed location uncertainties by implementing a system based on Ultra-wideband (UWB). This recent technology has become a robust solution for location problems in GNSS-denied environments (environments where communication between satellites and robots is inconsistent).

To handle the added complexity, some authors like Mutawe et al. [15], focus on decentralized architectures, proposing,

in this case, a control algorithm for trajectory tracking and path coordination, seeking to avoid collisions between them by implementing PID controllers in each robot, while others focus on centralized architectures, for example Matoui et al. [16] introduce a ''supervisor'' into the system who performs the necessary calculations and controls the robots, thus developing a centralized architecture for path planning in an MRS. Finally, in their article, Madridano et al. [17], review different methods and algorithms for trajectory planning, focusing on their implementation around MRS and solutions already studied.

On the subject of communication, Klavins [18] in his report Communication Complexity of Multi-robot Systems, studies the scalability of the algorithms used in MRS, considering the agents' coordination level and introducing the concept of complexity in communicating these systems and concluding that it is not easy to determine the minimum complexity that the system presents when performing a complex task, and Soni et al. [19], in his report a A Decentralized Relay-Based Approach for Multi-Robot Unknown Area Exploration, studies a decentralized network with distributed copies of exploration information. Other studies are aimed at testing and evaluating other communication solutions [8], such as Chen et al. [20], who study the feasibility of implementing a wireless network for the communication of an MRS in a so-called ''smart factory,'' seeking to incorporate flexibility, improve productivity and efficiently take advantage of the factory's energy.

An interesting work that aims to bring the study of multirobot systems closer to society is the so-called Robotic Park [21], a decentralized, indoor multirobot system that combines real aerial and terrestrial robots with simple simulated environments, using ROS2. This project involves topics such as digital twins, sensors and planning of simple tasks aimed at education, achieving the exchange of information between agents, especially spatial information, as well as the performance of simple tasks.

Other research projects such as the one presented by Kashid and Kumat [22] who evaluate their task assignment system, or the one presented by Mazdin et al. [23] also evaluating their task assignment system based on utility information, also make use of a decentralized MRS, which is interesting because it allows agents to share information among themselves, however it can present problems when the number of robots increases considerably, and it is also susceptible to information uncertainty, especially odometric information, due to external factors such as slippage on the surface or the effect of unexpected external forces.

In the following sections, we will be able to apply the study of prior art to make our multi-robot system configurations, based on a centralized system in conjunction with a supervisor module in order to manage uncertainties in the position of the agents and the assignment of tasks, being a proposal oriented to large-scale applications such as industrial processes, using the advantages of simulation in ROS2 to study their behavior.

III. MULTI-ROBOT FRAMEWORK SYSTEM

In the conceptual design of the MRS framework, the system comprises a central node responsible for managing and coordinating the different agents and sensors. The central node functions as the system's brain and is responsible for making high-level decisions, task distribution, and resource management. The central node collects the information the agents and sensors obtained and processes it to make decisions based on control algorithms. Using a centralized system allows for an efficient approach to coordinating multiple robots.

On the other hand, agents and sensors are physical entities involved with the environment. The agents are responsible for completing the required tasks and collecting environmental information. They can be robots equipped with sensors for location and perception. At the same time, there are external sensors such as cameras to perform computer vision or other IoT devices relevant to the chosen application. The communication between the agents and the central node is bidirectional, while the cameras and sensors have one-way communication, as shown in Figure 2.

The MRS framework consists of 3 modules: autonomous navigation, task distribution, and computer vision. Figure 2 shows a conceptual diagram of the MRS framework. The system inputs are the operator's commands to perform tasks and topics for an indefinite number of robots and cameras. The topics of the robots that enter the framework refer to the information from the LiDAR sensor of each robot. Previously, the work environment must be mapped and saved in a file. Subsequently, a map server is run to host the map file information in a ROS topic. The navigation module receives the information from the map, each robot's LiDAR, the location of each robot, and the target or final pose required. The output of the navigation module is the speed commands that will be sent to the robot controllers to follow the calculated trajectories.

On the other hand, camera topics contain data about the camera and the raw image obtained. With this information, the nodes are executed to detect ArUco markers and publish their transforms. Additionally, a filter was added to publish only the transform obtained from the camera closest to the detected ArUco. ArUco markers detect the poses of robots and workstations in the environment. The outputs of this module are a resulting image containing ArUco marker detection and initial positions for the probabilistic localization module. Finally, a task distribution module was developed that receives the operator's commands and assigns the task to the best-prepared robot, considering several criteria, among which proximity to the destination is prioritized.

A. LOCALIZATION BY COMPUTER VISION

One of the challenges present when implementing an MRS is the uncertainty of the data, which, as described in the previous chapter, refers to the lack of precision and unreliability of the information that a robot can provide, such as its location. There are multiple solutions, from the use of GPS systems to new techniques such as Visual-Inertial Odometry (VIO) and Simultaneous Localization and Mapping (SLAM) [24]. However, for closed environments, there is the possibility of using a computer vision system with cameras to acquire images. In this way, the design of the multi-robot framework of this project shows a computer vision component, seeking to provide the system with an extra source of information about the environment.

Computer vision, or artificial vision, is a branch of artificial intelligence that seeks to provide a computer with the ability to receive and process information from its environment in the form of images, just like a human being [25]. For this project, the use of artificial vision lies in the location of robots and jobs. For this reason, a rapid system is required to provide information online to the multi-robot framework capable of identifying the robots' position and orientation, providing data accurately and with a minimum margin of error. At the same time, it must also be scalable to multiple robots that are different from each other.

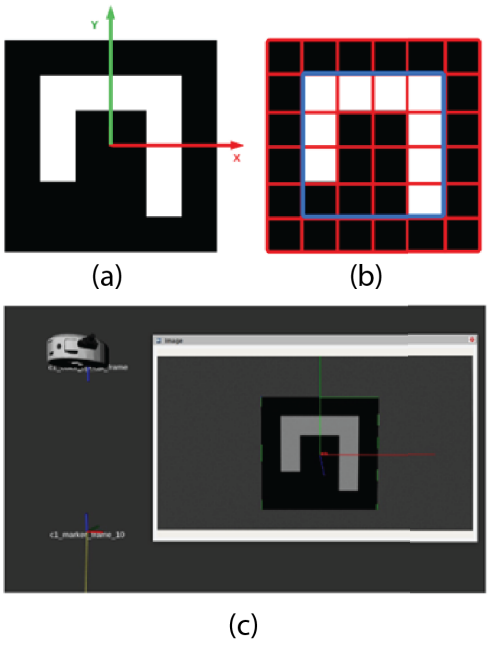


FIGURE 3. (a) Structure of an ArUco marker. (b) Cells of a 4 × 4 size marker with border. (c) Simulation of detection of ArUco marker ID 10 from the 4 × 4 dictionary by a camera.

Based on these requirements, a standard computer vision system that is trained to recognize robots in an image infers the position from the robot's size in the image and recognizes workspaces in a closed environment. However, it may require much investment in preliminary training data, and even then, it would not provide precise and accurate enough information for the system to be robust. For this reason, implementing an artificial vision system based on ArUco markers was evaluated.

ArUco is an open-source library developed by Rafael Muñoz and Sergio Garrido written in C++ and aimed at detecting fiducial markers or indicators in images, in addition to providing the ability to estimate their position within the camera's perspective, with the purpose of its application in augmented reality and robot localization [26]. This library allows you to generate indicators and implement a detection and positioning system. For its implementation within the multi-robot framework, the code provided by the OpenCV computer vision library [27] will be used, which uses the original ArUco model and other artificial vision components to detect markers.

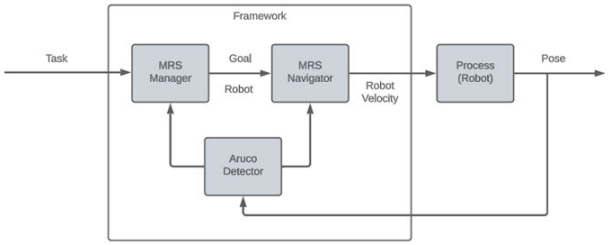


FIGURE 4. The full Framework Control System including the MRS manager, Navigator, and ArUco detector to control all robots and sensors. The manager assigns a task to an agent while the vision module allows the correction of the information on the agents' position to ensure that the task is completed.

ArUco indicators basically consist of 2D codes with a unique binary pattern for each indicator, allowing their identification, as seen in Figure 3. The ArUco markers are grouped into dictionaries according to the base used to form the binary patterns, according to the size and the number of indicators it contains. In this way, the dictionary can be customized according to the application, considering it is easier to detect identification codes. Smaller size and dictionaries that contain a small set of indicators. The project, requiring robustness and scalability, makes use of a default dictionary from the library: DICT_4 × 4_50, which consists of 2D 4 × 4 bit binary codes in order to use easily detected indicators, with a total of 50 ArUco codes that can be used, from indicator 0 to 49, being easy to access and manipulate by the end user, who has various tools both internal to the code and external on the web to generate indicators according to their needs, in addition to being able to change the dictionary when required, providing flexibility to the system. This entire artificial vision module will be tied to each camera used in the SMR.

B. AUTONOMOUS NAVIGATION MODULE

Autonomous navigation allows robots to move independently to reach defined destinations while avoiding obstacles. An autonomous navigation module comprises local, global,

and localization gliders. Global planners generate the optimal path based on the global map from the robot's current position to the specified goal using the A* algorithm. On the other hand, the local glider considers the robot's kinematics and the information about the environment captured by the sensor. It modifies the robot's trajectory in real time to avoid new obstacles [28]. The control algorithm used for autonomous navigation is a behavior tree. A behavior tree is a control architecture used to define the decision-making process of an agent in a modular way. A navigation-to-target behavior tree with replanning and recovery was used from the Nav2 package. In this way, the control system results as shown in Figure 4. The behavior tree is the control algorithm in the MRS Navigator block, which receives the target position from the task controller and operator commands. In addition, thanks to computer vision, the current transforms or positions of the robots are received through feedback. The variable controlled by the control algorithm is the speed command to the robots' motor controllers, which affects the process or system plant.

The Nav2 package was used to implement autonomous navigation. Each of the robot's package's nodes with the robot identifier must be executed. In addition, the configuration parameters were edited according to each robot, specifying the coordinate frames (tf) and corresponding topics of the robot. Since it is a framework, it was programmed so that there can be n robots. The user will only have to send the number of mobile robots in the collaborative environment as a parameter, and n navigation modules will be executed. The inputs to the autonomous navigation are a BehaviorTree XML file, the TF transforms, the map, and the LiDAR information. The "BT Navigation Server" node, the highest-level component, hosts the BehaviorTree to implement the desired navigation behaviors. The BT server node communicates with the rest of the modular nodes using ROS actions. Among the modular nodes is a server to create smoother routes, global and local planners, map servers, recovery servers, and the possibility of adding more of your servers. The output of the navigation module is speed commands for the motors of the mobile robots [29].

The Behavior Tree consists of two main subtrees shown in Figure 5 (a): Navigation and Retrieval. The NavigateRecovery node sets the number of attempts the robot will make to recover from an error in the navigation subtree before moving to the recovery subtree. In this case, after the navigation subtree returns an error six times, it will proceed to the navigation tree.

The navigation subtree, shown in Figure 5 (b), is responsible for the main navigation functions. It sequentially performs the actions of calculating a path to the goal and following the path. In the case of the planning action, a frequency of 1 Hz was established to prevent overloads. If any actions fail, contextual recoveries are attempted before proceeding to the recovery subtree. Contextual recoveries for the planning action check if the objective has been modified and clear the global cost map. On the other hand, the contextual recoveries of the path-following action are checking for changes in the target and clearing the local cost map.

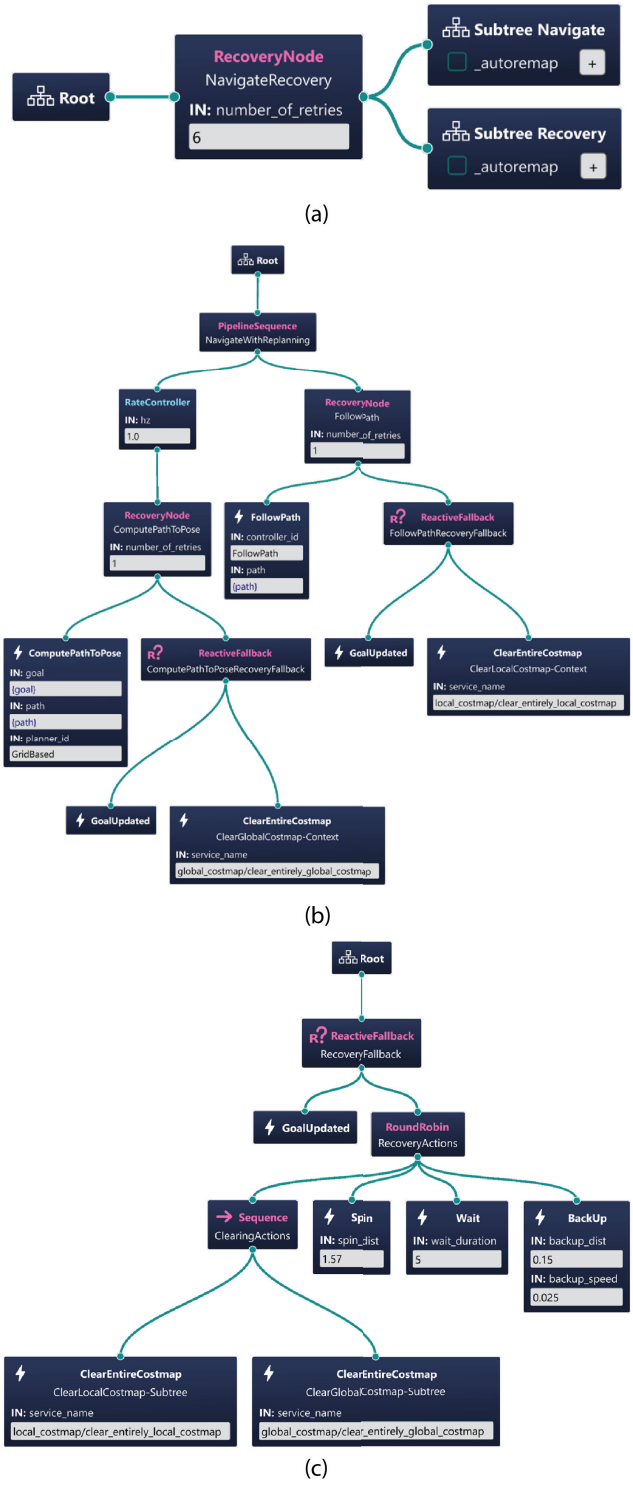


FIGURE 5. Autonomous navigation: (a) BehaviorTree of navigation and retrieval which controls the behavior of the agent in case of success or failure in the execution of tasks, (b) Navigation subtree responsible for carrying out navigation tasks, and (c) Recovery subtree responsible for executing the tasks of recovery and preparation for retrying a task.

The recovery subtree, shown in Figure 5 (c), is activated when the navigation tree returns failure. This subtree is responsible for performing system-level recoveries. The ReactiveFallback node is used to finish the recovery actions and return success if the target is updated. Recovery actions are performed until one of them returns successful or all of them fail. First, the global and local cost maps are cleaned. Subsequently, the robot performs a rotation and then pauses for five seconds. Finally, the robot moves back a specified distance at the set speed. If one of the recovery actions works, the robot tries to navigate the navigation subtree again. The navigation tree structure is robust and flexible, allowing the robot to navigate efficiently and handle various problems.

C. TASK ASSIGNMENT MODULE

A multi-criteria task assignment Algorithm 1 was implemented based on the state of each robot. The algorithm considers the proximity to the task and the availability of the robot. Furthermore, the algorithm was developed to be adaptive and dynamic. That is, if the state of a robot changes or variations occur in the environment, the algorithm will reevaluate the assignment of tasks. Figure 6 shows an example of a task assignment. A pick-up and delivery item is sent, requesting a robot perform this task. Since robot three is the closest to the picking position (green vector) and is not occupied, the task is performed by i.

Algorithm 1 Robot Near Task

Require: Robot_list
Require: Position_list
  $robot\_task \leftarrow 0$
  $distance\_min \leftarrow +inf$
  for $robot\_i \leftarrow 1$ in $Robot\_list$ **do**
    $distance\_i \leftarrow ||position\_goal - position\_robot\_i||$
    if $distance\_i < distance\_min$ and not $robot\_i\_bussy$ **then**
      $distance\_min \leftarrow distance\_i$
      $robot\_task \leftarrow robot\_i$
    else ▷ Nothing
    end if
  end for
  if $robot\_task = 0$ **then**
    Alert "Not robot available"
  else
    $robot \leftarrow robot\_task$
    $assignment \leftarrow task$
    Alert "Task assignment to robot"
  end if

D. STRUCTURE AND COMMUNICATION IN ROS2

All our framework is developed in ROS2, and the packages necessary for the construction of the framework, a repository was organized on GitHub[1] to give the community an

[1]https://github.com/RAMEL-ESPOL/MultiRobot

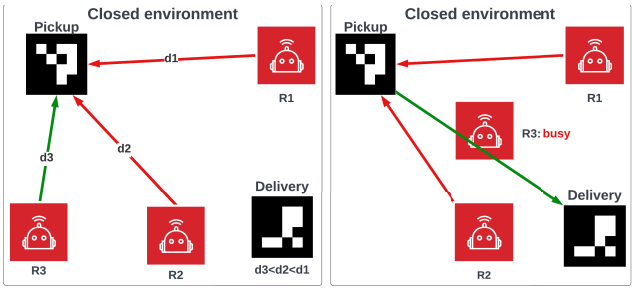


FIGURE 6. Task assignment to the free robot. (a) Search for the nearest robot for task assignment: PickUp an delivery. (b) Task execution by the robot: Robot status "Busy".

Open-Source Multi-Robot Framework System for collaborative environments. The repository structure is divided by modules, and components can be differentiated, as seen in Figure 7. Thus, the mrs_sensors and mrs_agents folders contain the sensor and robot packages, in this case, the Realsense camera model package and the TurtleBot3 and Create3 robot packages. On the other hand, the mrs_computer_vision folder contains the packages for ArUco marker recognition. In contrast, the mrs_main folder contains the task manager, navigation modules, application launchers, and the environment map file. Other folders like mrs_images hold project images, and mrs_ros2_dependencies contain modified dependencies for MRS.

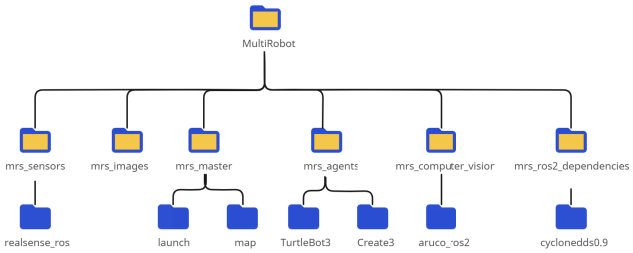


FIGURE 7. Project repository structure. ROS2 project structure by packages differentiated into modules: Main and computer vision; Agents with their respective navigation modules; Sensors used to collect information from the environment, dependencies and utility images.

The ROS2 visualization tool 'rqt' was used to obtain a concise, visual representation of the node topology and communication between components in ROS2. The result is summarized in Figure 8 (a). About cameras, they provide the topics "cnimage_raw" and "cnimage_info." These topics are received by the "marker publisher" node, which processes the information contained in them and detects ArUco markers. Subsequently, the node transmits the detections in the topic "cn markers." Once this process is completed, the topics generated by each camera converge on the "ArUco filter" node; in this node, the closest camera is selected to determine the precise position and orientation of the ArUco marker in question. Once this selection has been made, the corresponding position and orientation information is published in our system. On the other hand, robots receive

information from the map server through the ''map'' topic. Within the encapsulation of each robot, the processes of location, route planning, navigator, behavior, and control servers, among others, occur. Said topology has been broken down from the figure to allow a more understandable visualization, which will be addressed in a subsequent figure.

Figure 8 (b) shows the node graph for each MRS robot. The global cost map node receives the map and LiDAR topics. On the other hand, the local cost map node only receives the LiDAR information. The amcl location node uses the topics ''scan'' and ''map'' but also receives the robot's initial position from the task manager node. This node provides the transform between the map frame and the robot's odometry frame. The task manager node commands actions of type NavigateToPose to the ''bt_navigator '' node as Said node is the main of autonomous navigation. The ''ComputePathToPose'' action is commanded to ''planner_server'' to plan routes, the ''FollowPath'' action to ''controller_ server'' to follow the planned route avoiding obstacles and recovery actions such as ''Spin,'' ''Wait,'' and ''Backup.'' The ''lifecycle_manager_navigation '' node allows communication between all nodes through the topic ''bond.'' All the nodes in the figure's gray area are connected by Said Topic.

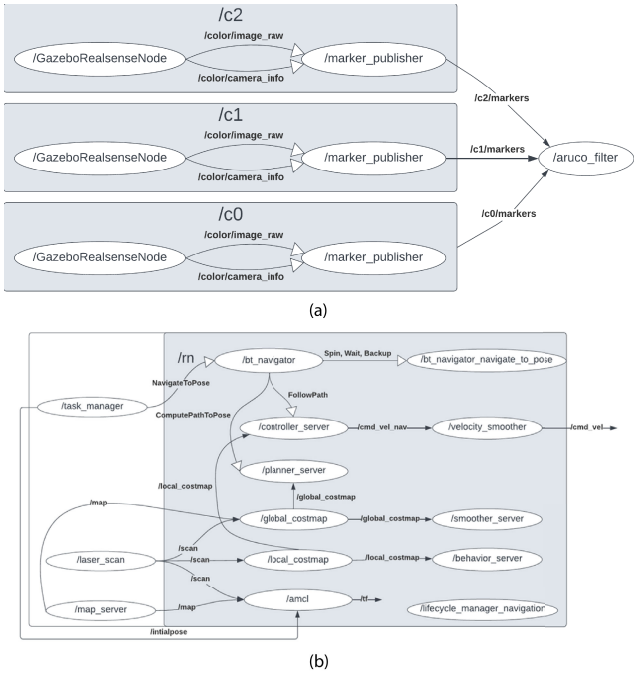


FIGURE 8. Topics and node ROS communication. (a) Simplified node graph for each camera, which are responsible for collecting information from the ArUco markers. (b) Simplified node graph for each robot, focused on its mobilization and navigation capacity.

Because our system needs several agents to be managed, namespaces in ROS2 were used for efficient coordination and communication. Each robot was assigned a namespace, which encapsulates the topics, nodes, and services of said robot, avoiding name and information conflicts. The namespaces that were used follow the r[n] structure. That is, the first robot added to the MRS has the namespace ''r1'' and so on progressively. Thanks to namespaces, topics are uniquely identified by their prefix, allowing communication between robots without interference. For example, the topic ''/r1/scan'' corresponds to the information captured by the LiDAR of robot 1, while the topic ''/r2/scan'' refers to the LiDAR data of robot 2.

Furthermore, using namespaces, greater modularity and reusability were obtained because the nodes developed for one robot can be easily adapted to the others by modifying the namespace. In the same way, the namespace was managed for the cameras added in the simulation with the reference c[n]. Figure 8 (a) shows the nodes corresponding to 3 cameras, encapsulated within the corresponding namespaces c0, c1, and c2. In addition, the topics (represented by arrows) also carry the identifier. However, the robot namespace does not affect the agents' TF coordinate frames or transforms. For this reason, prefixes with the same structure as the namespaces were decided to be used. In this way, there will be no conflicts with the references of each robot.

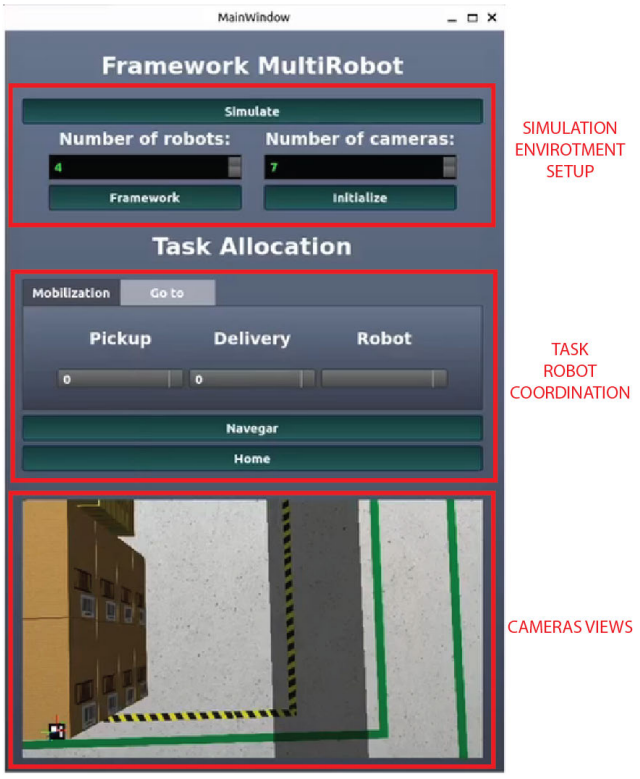


FIGURE 9. Graphical user interface: Simulation environment setup (number of robots and cameras connected to the system), Task robot coordination where the positions marked with ArUco markers are specified by number, and cameras views.

E. GRAPHICAL USER INTERFACE

A graphical user interface was implemented to facilitate the use of our framework for an operator, providing an intuitive and interactive experience even if they need in-depth

knowledge of ROS2. Figure 9 shows the final result obtained. The figure's square (SIMULATION ENVIRONMENT SETUP) includes the simulate button, opening the configured simulation. In the next scroll box, the user can enter the number of robots and the number of cameras either for simulation or for a real system and set up the framework with the Framework button. Finally, the button to initialize the system, setting the initial positions of the robots using our localization module by computer vision.

The square TASK ROBOT COORDINATION is about two tabs for task designation. The user can choose between object mobilization or objective position missions, as explained in the previous section, TASK ASSIGNMENT MODULE. Subsequently, set the parameters of each mission and send the task using the buttons below. The last square shows a real-time visual representation of the detection of ArUco markers, and you can select which camera you want to monitor.

IV. TESTS AND RESULTS

During the design and validation phases of the framework, simulations were carried out in GAZEBO. GAZEBO is a simulation environment that allows you to emulate real scenarios, so each framework module could be worked on and improved before fully implementing it in natural environments. A simulated world with obstacles and representative spatial layouts was created to represent the laboratory realistically. Our system tests include two Turtlebot3 and two Create3, working like a robot assistant inside the laboratory for different tasks, as shown in Figure 10. The system test included three Intel Realsense L515 cameras distributed in the laboratory, lying in the room to see the environment and the robots. In this way, the performance of our framework system can be evaluated in a context close to the real one. Figure 11 left shows the laboratory representation in the GAZEBO simulation environment.

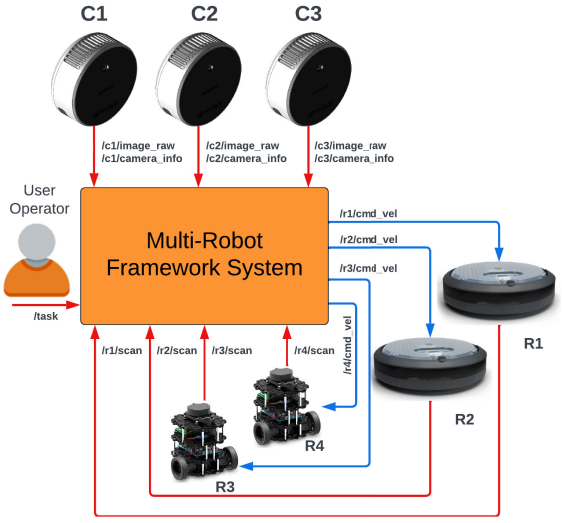


FIGURE 10. Simulation system test including the robots and cameras. Setup of the MultiRobot system test using 3 cameras and 4 robots in a simulated environment of the RAMEL laboratory.

Four different tests will be carried out on the modules of our framework to ensure that our system is effective for any number of robots within collaborative environments. The first test will be to get the position of the robots within the collaborative environment for which the location module was used, and the tests were carried out to obtain the position error of each robot within the environment based on what our cameras detect compared with the robot position at GAZEBO. The second test is about autonomous navigation, which was implemented in each of the robots, and it was checked several times to see if they complied with the calculation of the best routes of the target positions. At the same time, they do not collide with the environment or with each other. The third test is the task controller, which ensures that the free and nearby robot can always carry out the tasks. The test is that the robot picks up an object at one station and takes it to another, with the robot being closer and on a shorter route efficiently. Finally, the entire multi-robot system was tested with its modules within different collaborative environments, such as a warehouse and a restaurant; these tests are detailed below.

In order for robot tests to navigate inside our simulation environment at GAZEBO, a map of the environment in which they are located was required with LiDAR sensors were used to collect information from the environment and build a map with SLAM techniques, as shown in Figure 11 center. SLAM is the process in which a mobile robot can construct a map of the environment and simultaneously use it to obtain its location [30]. The map-obtaining process can be carried out with any ROS2 package or even with a method independent of ROS2. The final result to use the framework must be a map parameter file in ".yaml" format and an image of the map that can be in ".png", ".jpg", ". pgm", among others, as shown in Figure 11 right.

A. LOCALIZATION

ArUco markers were added to the simulated environment to test the localization module and recognize the real-time position of the robots and workspaces. During the startup of our Multi-Robot Framework, an ArUco recognition node is associated with each camera in the environment, which processes the image obtained from the camera to obtain the position and orientation of the ArUco markers it can recognize. These markers must belong to the dictionary. 4×4 and the actual size of the markers and the global reference on which the distance and orientation will be calculated can be configured as parameters. In Figure 12, you can see the result of detecting the ArUco markers carried out by camera c1, where two robots and two workspaces in black and white are included. One of the advantages of using ArUco markers is the generation of local coordinate systems for each marker so that by specifying a global reference coordinate frame in the parameters, the transformation of these local reference frames concerning an origin was obtained. The default, in this case,

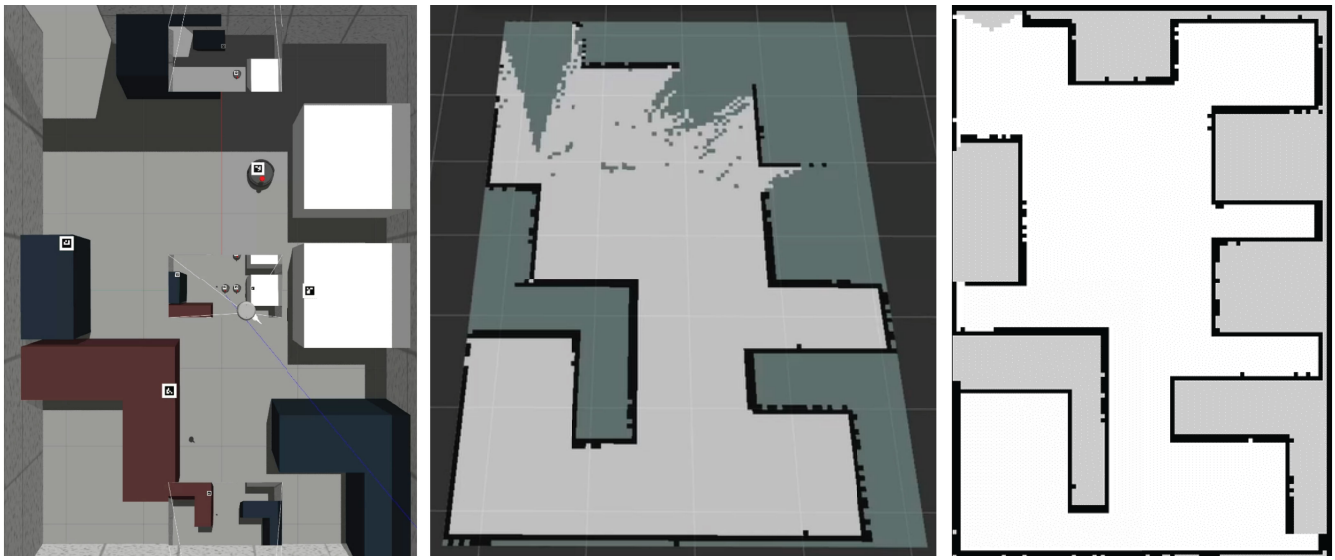


FIGURE 11. Laboratory real environment simulation in GAZEBO, SLAM process and map. From left to right: (1) Simulation of the environment recognizing the location of work stations and robots using ArUco and 3 cameras. (2) Mapping process of the work environment prior to starting the execution of tasks. (3) Resulting map.

is the center of the map, which can be displayed in the RVIZ tool with all ArUco references.

Our framework can identify how the ArUco of one of the robots is identified by two cameras simultaneously, showing two coordinate frames approximately in the same position. An arc filter node was programmed to handle these cases, which allows obtaining the information from each ArUco's recognition node belonging to each camera, gathering information into a single structure or message, and processing the repeated statement in order to simplify the necessary information and be able to use it for some functionalities of the framework.

One of the most important functionalities of the framework is the feedback on the positions and orientations of the robots, which allows updating information and correcting location problems due to external factors that interfere with the robot based on the information. From the ArUco filter, the ArUco robots are selected, and the initial_pose is published for each robot in the GAZEBO environment. The artificial vision localization module using ArUcos has an error that depends on the camera, its correct calibration, the distance the marker is located, and external factors such as lighting. Next, the absolute error of the positions of the ArUcos in different cases, visualized by camera c0, was calculated as shown in Figure 12.

For each of the cases, as can be seen in Figure 12, the 3D models of each marker in the GAZEBO simulation space have their reference system displaced to one of the sides in the x-axis in the negative direction, while the Markers identified through our location module have their reference systems located in the center of the marker, for this reason, to compare the results, for the positions shown by GAZEBO the corresponding value of 0.080 m will be added to the location on the x-axis to the displacement above (0.05 m due to half the size of the marker and 0.03 m from the white border). The error formulas to use for each of the coordinate axes correspond to equations 1, 2 and 3. In contrast, the location error will be obtained using equation 4, considering the position of the markers in the XY plane. On the other hand, the orientation error will be obtained by transforming the resulting quaternion to Euler angles, with the orientation error formula being equation 5 corresponding to the absolute value of the difference between the actual value (90° rotation in the z-axis) and the measured rotation value. Finally, Table 1 summarizes the results obtained in the detection of three ArUco markers carried out by camera c0 as shown in Figure 12.

$$X_{error} = ||(pose\ x_{GAZEBO} + 0.08) - pose\ x_{Rviz}|| \quad (1)$$

$$Y_{error} = ||(pose\ y_{GAZEBO}) - pose\ y_{Rviz}|| \quad (2)$$

$$Z_{error} = ||(pose\ z_{GAZEBO}) - pose\ z_{Rviz}|| \quad (3)$$

$$Error_{pose} = \sqrt{(X_{error})^2 + (Y_{error})^2} \quad (4)$$

$$Error_{rot} = |90^o - Z_{angle}| \quad (5)$$

Figure 12 left shows the real and estimated position for the ArUco with ID 0, located at a considerable distance from the camera. The errors calculated for the marker position were 0.0283m, 0.0146m, and 0.0723m in the x, y, and z axes, respectively. On the other hand, in the Figure center, the position errors of the ArUco marker of ID 1 (which is the closest to the camera) were 0.0104m, 0.0012m, and 0.1061m in the corresponding axes. Finally, the errors obtained for the ArUco of ID 2 in Figure right, which was the furthest from the camera, were 0.0193m, 0.0105m, and 0.0268m in the respective coordinate axes.

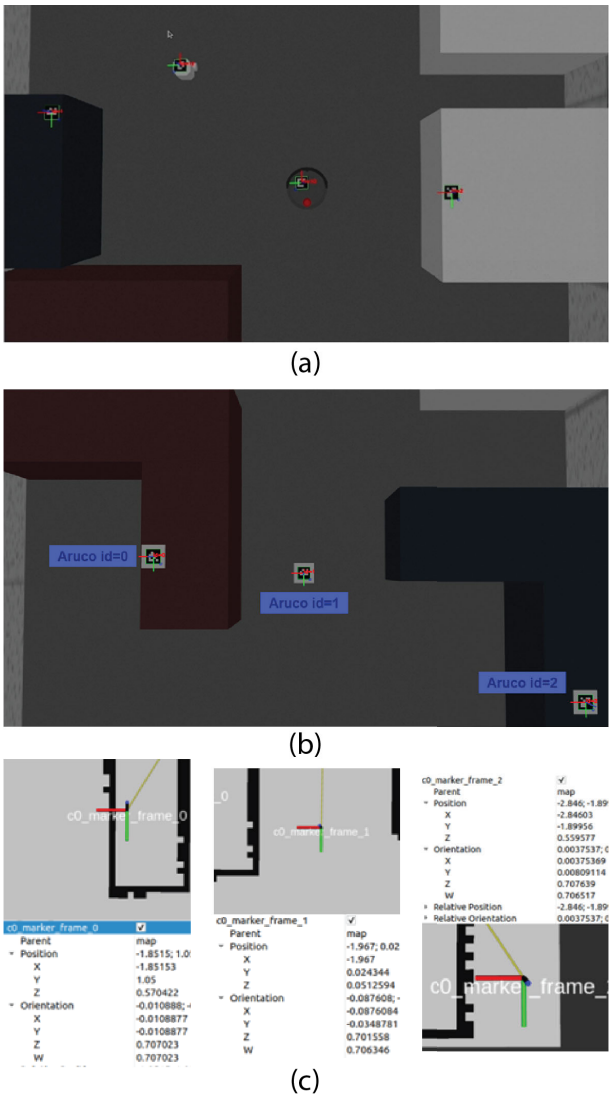


FIGURE 12. Computer vision module. (a) Detection and recognition of ArUco markers in GAZEBO. (b) Recognition of ArUco markers by the c0 camera in GAZEBO and (c) detected location and information of the ArUco markers.

From these results, only the position in the XY plane will be considered for the location error, so using Table 1, a general position error of 0.0244 m is obtained, which is an acceptable error considering that the camera was at an approximate height of 3.5m. Mobilization applications do not require high precision as long as they have sensors, obstacle detection, and avoidance modules. Regarding the orientation or rotation of the markers, the average error obtained was 0.29°, an acceptable error that allows correcting orientation problems in the robot in case of impacts or slips. In the application to be carried out, the position on the z-axis is not of utmost importance because the mobilization of the robots will be carried out in the XY plane. However, the error presented in this axis is low enough to consider using this application in future tasks to be implemented in the framework.

TABLE 1. Position errors occur during ArUco marker recognition.

ArUco	Position error [m]	Rotation error [°]
Id 0	0.0250	0.00
Id 1	0.0253	0.77
Id 2	0.0229	0.09
Average	0.0244	0.29

B. AUTONOMOUS NAVIGATION

With the autonomous navigation module implemented in our framework for n robots, each robot can calculate the optimal route to the target pose and avoid collisions with obstacles autonomously. Figure 13 shows a navigation sequence of three robots. Consecutively, they were assigned an objective pose with the RVIZ tools. The robots calculate the optimal route to reach the pose and follow it until they reach the target pose. The objective poses of each robot can be differentiated by the namespaces used. Every figure shows the RVIZ, including the three cameras and the local cost map of all robots, which allows for optimal route calculation based on occupancy grids and the real simulation in GAZEBO, which appreciates the laboratory environment. At the same time, If it encounters an obstacle, the robot will recalculate the optimal route to reach the target pose. We tested by adding a cube to the simulated GAZEBO environment to represent a chance obstacle. Previously, one robot had been assigned an objective pose on the other side of the map. The robot begins to follow the calculated route, but upon perceiving the new obstacle, it creates a new route and manages to avoid it to reach the assigned pose successfully.

C. TASK CONTROLLER

Our multi-robot framework developed has a task manager responsible for assigning tasks to the robots, executing position update commands, and choosing the robot that should accomplish the task based on its distance in case the operator does not specify the robot. Figure 14 shows the execution of the collection and delivery task where the framework has been specified, through the graphical interface, the requirement for a robot to approach workplace zero, pick up an object, and deliver it to workstation two, in this case, the task is executed by robot one successfully.

Another example is the execution of the task of going to a position where the framework has been specified through the graphical interface, the requirement for a robot to approach the point (x,y) = (−2.0), this task being executed by the robot two. The robot that must execute the task in question is not specified in both tasks. For this reason, the manager obtains the positions of each robot as well as their states (free or busy), choosing a robot to execute the action considering that said robot is the closest to the destination point (Pickup in the case of the ''Mobilization'' task for collection and delivery of products, and the point (x,y) specified in the case of the ''Go To'' task. For this reason, in the first case, the task was assigned to robot one, while in the second case, the task was

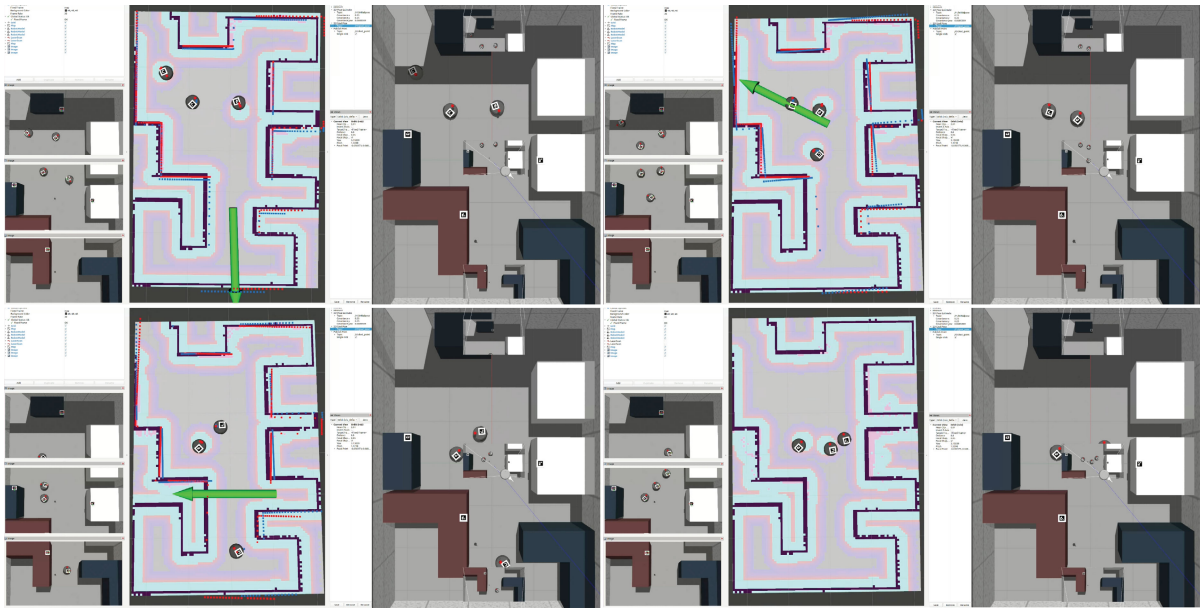


FIGURE 13. Autonomous navigation test. Using RVIZ tools to execute mobilization tasks with each of the agents, the use of namespaces is important to differentiate each of the agents and correctly assign the task. In each of the four captures you can identify the view of the 3 cameras, the navigation cost map and the view of the simulated environment.

FIGURE 14. Task controller test. From top to bottom: (1) Preview of the environment with the location of robots, cameras and workstations marked with ArUcos. (2) Using the GUI to run a pickup and delivery task, assisting the robot closest to the pickup location of an object (in this case a non-relevant camera). (3) The task is completed with the robot arriving at the destination station.

assigned to robot two, which, during the assignment, was close to the destination points and accessible to any other task.

D. DIFFERENT SIMULATED ENVIRONMENTS

To test our MRS in different environments, we simulated an industrial warehouse where the robots deliver and collect packages between different warehouse sections. For this case, three turtlebot3 robots and two Create3 robots were used, resulting in the simulated environment shown in Figure 15 already with added cameras and ArUco markers. In package storage areas, the ArUco markers were added to three shed sections. The test consists of several package delivery and pickup tasks that are commanded consecutively using the graphical user interface. If the robot is not specified for each commanded task, the closest unoccupied robot approaches. The robots successfully complete their collection and delivery tasks between the warehouse stations and are again available to receive new tasks.

In the same way, a simulated environment was used to represent a restaurant or cafeteria to carry out the task of collecting and delivering tableware and plates of food. A Turtlebot3 was added, with five robots in this MRS. Figure 15 shows the result of the simulated environment with all the components necessary to use the MRS. The ArUco markers were placed on each of the five tables for customers. In addition, an ArUco marker was added to the counter next to the kitchen so that the robots could carry and collect the dishes from the kitchen. The test consists of various tableware collection and delivery tasks that are carried out, commanded from the graphical user interface. It should be noted that the collection station near the kitchen sometimes has a lot of robot congestion since it is a fixed station for each task. However, the robots avoid each other, and no collision occurs. The robots successfully execute their assigned task and become available again for newly commanded tasks.

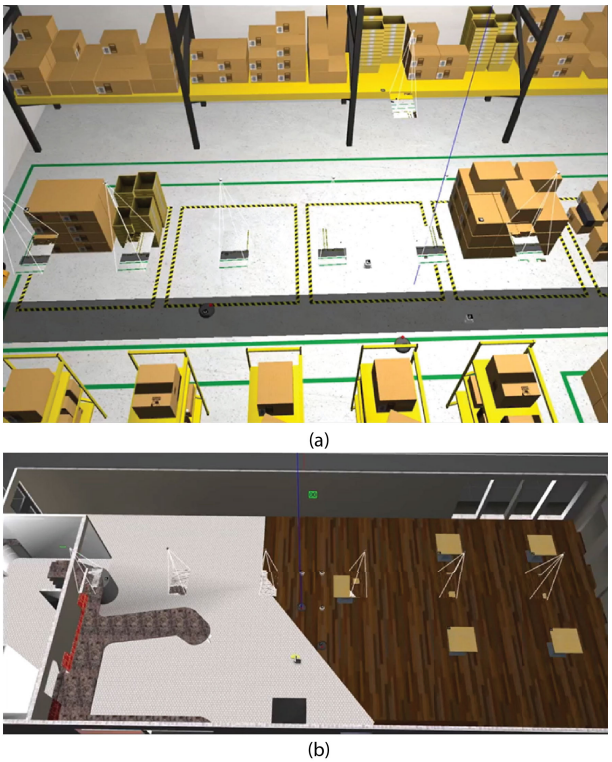


FIGURE 15. Industrial warehouse and restaurant or cafeteria simulation test with more than ten robots and cameras. The use of cameras to detect the position of robots and work stations (storage stations in the first case and tables in the second case) is visualized, being able to execute the tasks of delivery and transport of objects.

V. CONCLUSION

In contemporary industry, a rising need for automation and collaboration in closed environments has been identified, and the developed FrameWork MultiRobot turned out to be a significant advancement. It was possible to obtain a robust, flexible, and scalable solution - key parameters in industrial processes- using ROS2 to achieve effective communication and coordination of various robotics systems. Effective communication and coordination between system agents was obtained using the MultiRobot software architecture in ROS2. The software architecture allows adding more robots and sensors to the system. The robustness of the architecture highlights the robots' response capacity in highly complex situations.

A simulated environment was developed to represent RAMEL, allowing the FrameWork MultiRobot to be progressively designed. Likewise, simulated environments of an industrial warehouse and a cafeteria were used to validate the operation of FrameWork, obtaining correct operation for fleets of up to 6 to 18 robots and heterogeneous fleets in different environments. The precise position and orientation of the robots and workspaces in the environment were determined using computer vision techniques and ArUco markers. In addition, the localization module allows the robots' positions to be initialized on the navigation map and their positions rectified after completing or failing an assigned task. The location error of ArUco markers is ± 0.025 m in the XY plane, and a rotation error of 0.29°.

LiDAR sensors and SLAM techniques enabled autonomous navigation in simulated closed environments. Thanks to this, the agent can locate itself on the map while moving, comparing the LiDAR information with the generated map. Additionally, LiDAR allows the robot to identify obstacles not present on the map and compute a new route to its destination. A task controller assisted by artificial vision was developed to assign object mobilization missions to the robot closest to the collection point, improving operational efficiency by assigning several missions simultaneously. It is also applicable in industrial warehouses, hospitals, and restaurants, among other applications. A go-to XY coordinate mission was added to the task controller, allowing different missions to be assigned to the same controller.

In conclusion, the work developed has a significant impact on the field of collaborative robotics and system automation. Compared to existing works, the FrameWork MultiRobot provides a comprehensive solution that ranges from perception through artificial vision to reduce uncertainty in the positioning of agents to the assignment and execution of collaborative tasks. The MultiRobot systems implemented with the developed FrameWork will have strengths such as robustness, scalability, adaptability, and flexibility. However, it is possible to continue exploring improvements to the system, evaluating and improving the functioning of the modules which allow for the coupling of other possible variants that use other technologies.

REFERENCES

[1] B. Dynamics, "Spot®—The agile mobile robot," Boston Dyn., Waltham, MA, USA, Tech. Rep., 2023. [Online]. Available: https://bostondynamics.com/products/spot/
[2] *Servi Plus,* Bear Robot., Redwood City, CA, USA, 2023.
[3] *Turtlebot 4*, clearpath Robot., Kitchener, ON, Canada, 2023.
[4] *Autonomous Robot Delivery—The Future of Delivery—Today!*, Starship Technol., San Francisco, CA, USA, 2023.
[5] A. Gautam and S. Mohan, "A review of research in multi-robot systems," in *Proc. IEEE 7th Int. Conf. Ind. Inf. Syst. (ICIIS)*, Aug. 2012, pp. 1–5.
[6] M. Quigley, "ROS: An open-source robot operating system," in *Proc. ICRA Workshop Open Source Softw.*, Jan. 2009, pp. 1–6.
[7] A. Gautam, B. Jha, G. Kumar, J. K. Murthy, S. A. Ram, and S. Mohan, "FAST: Synchronous frontier allocation for scalable online multi-robot terrain coverage," *J. Intell. Robotic Syst.*, vol. 87, nos. 3–4, pp. 545–564, Sep. 2017.
[8] A. Gautam, A. Soni, V. Singh Shekhawat, and S. Mohan, "Multi-robot online terrain coverage under communication range restrictions—An empirical study," in *Proc. IEEE 17th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2021, pp. 1862–1869.
[9] J. J. Roldan-Gomez, J. D. L. Rivas, P. Garcia-Aunon, and A. Barrientos, "Una revision de los sistemas multi-robot: Desafios actuales para los operadores y nuevos desarrollos de interfaces," *Revista Iberoamericana de Autom. E Inform. Ind.*, vol. 17, pp. 294–305, Jul. 2020.
[10] M. Sachon, "Los pilares de la industria 4.0," in *Proc. Revista de Negocios del IEEM*, 2018, pp. 1–4.
[11] R. Cuautle, E. Berra, and M. Perez, "Robotica cooperativa para el transporte de objetos: Revision historica de estrategias," Universidad del Valle de Puebla, Puebla, México, Tech. Rep., 2018.
[12] P. U. Lima and L. M. Custodio, "Multi-robot systems," in *Innovations in Robot Mobility and Control*. Lisboa, Portugal: Instituto Superior Técnico, 2005, pp. 1–64.

[13] L. Zhang, Y. Sun, A. Barth, and O. Ma, "Decentralized control of multi-robot system in cooperative object transportation using deep reinforcement learning," *IEEE Access*, vol. 8, pp. 184109–184119, 2020.
[14] W. Shule, C. M. Almansa, J. P. Queralta, Z. Zou, and T. Westerlund, "UWB-based localization for multi-UAV systems and collaborative heterogeneous multi-robot systems," *Proc. Comput. Sci.*, vol. 175, pp. 357–364, Jan. 2020.
[15] S. Mutawe, M. Hayajneh, S. Banihani, and M. A. Qaderi, "Simulation of trajectory tracking and motion coordination for heterogeneous multi-robots system," *Jordan J. Mech. Ind. Eng.*, vol. 15, pp. 337–345, Oct. 2021.
[16] F. Matoui, B. Boussaid, B. Metoui, and M. N. Abdelkrim, "Contribution to the path planning of a multi-robot system: Centralized architecture," *Intell. Service Robot.*, vol. 13, no. 1, pp. 147–158, Jan. 2020.
[17] Á. Madridano, A. Al-Kaff, D. Martín, and A. de la Escalera, "Trajectory planning for multi-robot systems: Methods and applications," *Expert Syst. Appl.*, vol. 173, Jul. 2021, Art. no. 114660.
[18] E. Klavins, "Communication complexity of multi-robot systems," in *Algorithmic Foundations of Robotics V* (Springer Tracts in Advanced Robotics), vol. 7. Cham, Switzerland: Springer, 2004, pp. 275–291.
[19] A. Soni, C. Dasannacharya, A. Gautam, V. S. Shekhawat, and S. Mohan, "D-MRFTE: A decentralized relay-based approach for multi-robot unknown area exploration," in *Proc. IEEE Int. Conf. Robot. Biomimetics (ROBIO)*, Dec. 2023, pp. 1–7.
[20] K.-C. Chen, S.-C. Lin, J.-H. Hsiao, C.-H. Liu, A. F. Molisch, and G. P. Fettweis, "Wireless networked multirobot systems in smart factories," *Proc. IEEE*, vol. 109, no. 4, pp. 468–494, Apr. 2021.
[21] F.-J. Mañas-Álvarez, M. Guinaldo, R. Dormido, and S. Dormido, "Robotic park: Multi-agent platform for teaching control and robotics," *IEEE Access*, vol. 11, pp. 34899–34911, 2023.
[22] S. Kashid and A. D. Kumat, "Hierarchically decentralized heterogeneous multi-robot task allocation system," in *Proc. 7th Int. Conf. Intell. Robot. Control Eng. (IRCE)*, Aug. 2024, pp. 143–148.
[23] P. Mazdin, M. Barcis, H. Hellwagner, and B. Rinner, "Distributed task assignment in multi-robot systems based on information utility," in *Proc. IEEE 16th Int. Conf. Autom. Sci. Eng. (CASE)*, Aug. 2020, pp. 734–740.
[24] M. Kalaitzakis, B. Cain, S. Carroll, A. Ambrosi, C. Whitehead, and N. Vitzilaios, "Fiducial markers for pose estimation: Overview, applications and experimental comparison of the ARTag, AprilTag, ArUco and STag markers," *J. Intell. Robotic Syst.*, vol. 101, no. 4, pp. 1–26, Apr. 2021.
[25] C. Alonso and A. David, "Visión por computador: Identificación, clasificación y seguimiento de objetos," *Appl. Math.*, vol. 1, no. 1, pp. 1–17, May 2014.
[26] R. Muñoz-Salinas, M. J. Marín-Jimenez, E. Yeguas-Bolivar, and R. Medina-Carnicer, "Mapping and localization from planar markers," *Pattern Recognit.*, vol. 73, pp. 158–171, Jan. 2018.
[27] *Opencv: Detection of Aruco Markers*, OpenCV, USA, 2023. [Online]. Available: https://docs.opencv.org/4.x/d5/dae/tutorial_aruco_detection.html
[28] S. Macenski, F. Martín, R. White, and J. G. Clavero, "The Marathon 2: A navigation system," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 2718–2725.
[29] *Nav2—Navigation 2 1.0.0 Documentation*, Open Navigation, USA, 2023. [Online]. Available: https://docs-nav2-org.translate.goog/?_x_tr_sl=en&_x_tr_tl=es&_x_tr_hl=es&_x_tr_pto=tc
[30] H. Durrant-Whyte and T. Bailey, "Simultaneous localization and mapping: Part I," *IEEE Robot. Autom. Mag.*, vol. 13, no. 2, pp. 99–110, Jun. 2006.

FRANCISCO YUMBLA (Senior Member, IEEE) received the degree in electrical engineering with a specialty in electronics and industrial automation from the Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador, in 2014, and the Ph.D. degree in mechanical engineering in the area of robotics from Sungkyunkwan University (SKKU), South Korea, in 2021. He was a Postdoctoral Researcher with the Robotics Engineering Research Center, Sungkyunkwan University. In 2022, he joined the Faculty of Mechanical Engineering and Production Sciences, ESPOL, as a Professor. His research interests include autonomous vehicles, manipulation of robotic arms, design of robots and manipulators, and artificial intelligence.

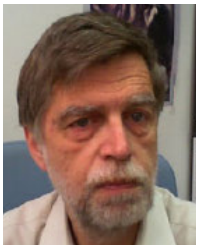
MARCELO FAJARDO-PRUNA (Member, IEEE) received the Graduate degree in mechanical engineering from the Escuela Politécnica Nacional (EPN), Ecuador, in 2011, and the Ph.D. degree in mechanical engineering, specializing in manufacturing processes from the Universidad Politécnica de Madrid (UPM), Spain, in 2018. Furthermore, he distinguished himself with UPM. In 2019, he started as a Research Professor with the Faculty of Mechanical Engineering and Production Sciences, ESPOL. His research interests include deeply invested in developing intelligent control strategies for autonomous systems and digital-twin technologies blending virtual and physical worlds for enhanced efficiency and precision. In micro-cutting technologies and machine tools design, he looks to optimize processes for the highest accuracy and productivity. His research into vortex systems also offers groundbreaking approaches to mitigate urban heat islands, demonstrating his commitment to addressing global environmental challenges.

ANTHONNY PIGUAVE received the degree in mechatronics engineering from the Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador, in 2023. During the last year of his career, he was a member of the Robotics, Automation & Mechatronics Engineering Laboratory (RAMEL), where focused on the study of multi-robot systems for collaborative indoor applications and artificial vision.

DIEGO RONQUILLO received the degree in mechatronics engineering from the Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador, in 2023. Furthermore, he enriched his academic journey through a research assistantship with the Universitat Politécnica de Catalunya (UPC), Barcelona, Spain, where he was introduced to ROS2 and immersed himself in the realm of robotics.

RICARDO ORTIZ received the degree in mechatronics engineering from the Escuela Superior Politécnica del Litoral (ESPOL), Guayaquil, Ecuador, in 2020, and the master's degree from The State University of New York, South Korea (SUNY Korea), in 2022. He is currently pursuing the Ph.D. degree. In 2021, fueled by his passion for advanced studies, he embarked on a journey with SUNY Korea. He is currently a Valuable Research Assistant with the Mechanical Systems with the Intelligence and Computer Vision Laboratory (MEICLab).

JONGSEONG BRAD CHOI (Member, IEEE) received the B.S. and M.S. degrees in ME from the University of Mississippi and the Ph.D. degree in ME from Purdue University. He is currently an Assistant Professor in mechanical engineering with The State University of New York, South Korea (SUNY Korea) and has been an affiliated Faculty Member of ME with Stony Brook University, NY, USA, since 2020. His principal research interests include exploit engineering-soft-power (e.g., computer vision, visual analytics, metaverse, and digital twin) as a tool to develop techniques. Since 2017, he has been made significant progress in the visual analytics field with multiple pioneering research projects supported by National Science Foundation (NSF), NASA, and European Union (EU), and National Science Foundation of Korea (NRF). To date, he has published over 40 journals and conference papers, with a major focus on visual analytics and remote sensing. Also, he holds a board committee position in Korean Society of Mechanical Engineers (KSME) and Korean Society of Prognostics and Health Management (KSPHM), where he won the Young Promising Scientist Award.



GABRIEL DÍAZ (Senior Member, IEEE) received the Ph.D. degree in physics from the Universidad Autonoma de Madrid, in 1988. He is currently an Associate Professor with the Electrical, Electronics and Control Engineering Department, Spanish University for Distance Education (UNED), Madrid, Spain. His research interests include various approaches for getting the best of ICT technologies applied to different kinds of security and electronics learning for higher education, security measurement and metrics, and security for process control systems.

XABIEL G. PAÑEDA received the Ph.D. degree from the University of Oviedo, in 2004. He is currently an Associate Professor with the University of Oviedo. His main research interests include the Internet of Things and intelligent transportation systems. He has been working on these topics and others to design technologies to enable the coexistence between autonomous vehicles and human tripulated vehicles.

HYUNGPIL MOON (Member, IEEE) received the B.S. and M.S. degrees in mechanical engineering from Pohang University of Science and Technology, Pohang, South Korea, in 1996 and 1998, respectively, and the Ph.D. degree in mechanical engineering from the University of Michigan, Ann Arbor, MI, USA, in 2005. From 2006 to 2007, he was a Postdoctoral Researcher with the Robotics Institute, Carnegie Mellon University. In 2008, he joined as a Faculty Member of the School of Mechanical Engineering, Sungkyunkwan University, Suwon, South Korea, where he is currently a Professor. His current research interests include robotic manipulation, SLAM, combined task and motion planning, and polymer-based sensor and actuators. He is the Co-Chair of IEEE RAS TC on Robotic Hand, Grasping, and Manipulation. He was a Technical Editor of IEEE/ASME TRANSACTIONS ON MECHATRONICS and the Editor-in-Chief of the *Journal of Korea Robotics Society*. He is also serving as an Editor for ICRA and an Associate Editor for *Intelligent Service Robotics*.