Hindawi
Journal of Robotics
Volume 2023, Article ID 6630038, 15 pages
https://doi.org/10.1155/2023/6630038

Research Article

Three-Dimensional Location and Mapping Analysis in Mobile Robotics Based on Visual SLAM Methods

Gustavo A. Acosta-Amaya,[1] **Juan M. Cadavid-Jimenez**,[2] **and Jovani A. Jimenez-Builes**[2]

[1]*Politécnico Colombiano Jaime Isaza Cadavid, Medellín 050005, Colombia*
[2]*Universidad Nacional de Colombia, Medellín 050005, Colombia*

Correspondence should be addressed to Jovani A. Jimenez-Builes; jajimen1@unal.edu.co

Received 22 February 2023; Accepted 29 March 2023; Published 16 June 2023

Academic Editor: L. Fortuna

One of the essential tasks required from a mobile robot is the autonomous and safe navigation of its working environment. However, in many cases, a model of the environment or map is not available to execute this task. Indeed, navigation requires a permanent estimation of the location for a map, which is not available for unknown environments. In such a scenario, the robot must have extended capabilities to solve, concurrently, the problems of localization and mapping. The simultaneous solution of these two problems is known as SLAM (simultaneous localization and mapping) and is a complex problem, not yet fully solved by the scientific community. This is due to the fact that localization requires a map that is not yet available since it is still under construction. In turn, the elaboration of a map requires the estimation of the robot's location. This is the reason why SLAM has been categorized as similar to the chicken and egg problem. In the case of a robot facing an unknown environment, it would be something like what to solve first, localization or mapping? The answer to this question is that the robot will have to solve both problems at the same time. This article presents a study of some of the most representative open source visual SLAM (vSLAM) methods, beginning from an analysis of their characteristics and presenting criteria selection for an experimental design that allows contrasting their advantages and disadvantages. Two of the most representative algorithms for solving vSLAM were considered (RTAB-Map and ORB-SLAM2). The experiments were validated with a robotic system designed for this purpose, which is fully compatible with ROS (robot operating system).

1. Introduction

One of the biggest challenges faced by robotics research is the incorporation of advanced levels of autonomy in robotic systems, understanding autonomy as the capacity of a system to carry out its processes and operations with no human intervention. Navigation is perhaps the problem most closely linked to the concept of autonomy in mobile robots. Precisely, some of the tasks that define a mobile robot as an autonomous entity are a safe navigation of its environment and path planning [1].

In robotics, the terms location, pose, and localization refer to the coordinates and orientation of a robot with regards to a global spatial representation system [2]. Localization and mapping, which is the ability of some robots to model their environment, are two fundamental problems in robotics and are closely related [3]. When a representation of the environment is not available, it must be constructed autonomously by the robot. This is a nontrivial problem; since in order to build models of the environment, it is necessary to establish a precise location in relation to a model that is not yet available. Similarly, in order to obtain a good estimate of the location, a map is required. Consequently, when a robot faces an unknown environment, it will have to simultaneously solve two problems that cannot be solved independently: localization and mapping [4]. These two problems are so strongly correlated that it is not possible to establish which of them is the cause and which is the effect. The need to solve

both problems concurrently results in the simultaneous localization and mapping (SLAM) problem.

To successfully carry out these two tasks, the robot must have extended capabilities to reliably estimate its localization (x, y, and θ)^T and be able to elaborate relatively accurate representations of its environment (mapping). Traditional methods employed in solving SLAM are based on using probabilistic techniques. In such techniques, parameters and measures are treated as random variables, and the uncertainty in localization estimation is modelled by using probability density functions.

This article is addressed to the problem of localization and mapping simultaneous of indoor environments based on vSLAM methods, under controlled conditions that allow highlighting the differential aspects of two of the most representative approaches, namely, RTAB-Map (real-time appearance-based mapping) and ORB-SLAM2 (oriented FAST and rotated BRIEF simultaneous localization and mapping 2). Both RTAB-Map and ORB-SLAM2 are representative methods of the current state of the art in vSLAM (visual SLAM). However, while the first produces dense maps, the latter generates a sparse one. Equally, the robot trajectory and the three-dimensional structure of the environment are estimated. In the authors' criteria, the contributions of this article are the following: (1) a holistic view of vSLAM methods that incorporates conceptual and procedural elements that support the selection of the most convenient method for the development of robotic applications. (2) A detailed discussion of simple experimental settings under controlled conditions that tend toward repeatability is provided, facilitating comparison of methods. (3) A complete interpretation of the models obtained from each approximation is provided in real environments to which a wide variety of objects that provide complex characteristics are incorporated.

This paper is distributed as follows: in the second section, materials and methods established for this research are discussed. The third section presents the qualitative selection method proposed in this study. The fourth section describes the experiments performed and discusses the results obtained. Finally, in the last two sections, conclusions are drawn and references are presented.

2. Materials and Methods

In recent years, robotics research and the development of robotic systems have experienced significant growth, widely due to the emergence of new modelling, estimation, optimization, and control methods, and advances in computing, sensing, mechatronics, and communications. This scenario is conducive to the arising of new applications of robotics in various areas of daily life [5, 6] and the need for versatile and powerful tools for the design, simulation, and development of functional robotic systems.

Robots are now considered complex distributed systems that include a wide variety of hardware and software components, which is why this research incorporates the highest possible degree of modularity in the design and implementation of the experimental robotic system, both at the hardware and software levels. Although modular design poses additional integration, communication, and interoperability problems, these can be solved by using an intermediate software layer or middleware for distributed robotic systems [7]. Generally, the latest generation of open source middleware facilitates integration with libraries and packages widely used in computer vision, point cloud processing, 3D geometry, and simulators. They greatly ease the development, testing, and debugging of algorithms with rigorous security and robustness requirements, mainly in applications where a close interaction with people is needed, such as assistance and educational and logistic robots for goods delivery [8–10].

As for hardware, several companies around the world (Clear-path Robotics, Fetch Robotics, PAL robotics, and Hokuyo Automatic) develop and supply a wide range of robotic platforms and sensing, perception, and actuation systems for research, learning, and development of industrial and commercial applications compatible with different middlewares.

The following four tools were used in the experiments conducted in this work (robot operating system-ROS, gazebo, turtlebot2, and workstation) [11].

2.1. Robot Operating System (ROS). Middleware, which has been well accepted in academic and industrial environments, has positioned itself, de facto, as a standard for robotic application development [12–14]. ROS is an open-source middleware for robotic application development, which provides a distributed programming environment for both real and simulated platforms. It also provides hardware abstraction, low-level control, message transfer between processes, and software package management [15]. ROS was conceived considering the following criteria (Table 1):

2.2. Gazebo. Simulators are compatible with ROS middleware. Preliminary tests of the implemented teleoperation algorithms were carried out with this tool, which offers a 3D simulation environment that requires a computer with good graphics processing capabilities, as the ones supported by GPUs (graphics processing unit) [17, 18].

2.3. Turtlebot2. It is an open hardware platform for research, testing, and development of methods, algorithms, and applications in robotics [12, 14, 15]. It holds the necessary perception, control, and actuation systems to start with the development of ROS-supported applications. Some low-cost modifications were carried out to improve the processing and autonomy of the platform [19].

2.4. Processing and Communications. Computers were available, one of the high performances that we call as workstation, which was used for system supervision through GUI (Graphical User Interfaces) and for the execution of high computational cost processes. The other equipment, of lower performance and that we call Robot-PC, was incorporated into the mobile robot to execute processes of lower computational cost, mainly those associated with

Table 1: Selected criteria for ROS.

Criteria	Features
Peer-to-peer	Processes connected at runtime without fixed clients and servers. Processes behave as both clients and servers
Distributed processing	Processes (nodes) can run on different hosts and communicate across the network
Lightweight system	ROS provides a lightweight packaging mechanism for third-party libraries
Multilanguage	Although ROS supports different languages such as Octave, LISP, Lua, and Java, the APIs (application programming interface) with the best support are those of C++ (roscpp) and Python (rospy)
Free and open source	The entire ROS source code is in the public domain
Tool-based	ROS design is based on a microkernel in which a large number of small tools are used to build and execute various ROS components, as opposed to a monolithic development and execution environment

Source: [16].

proprioception, exteroception, and effector control of the turtlebot2 robot. The connectivity of both the devices was executed through a router.

Features of the hardware used are as follows (Table 2):

The experimental mobile robot prototype was named RobSLAM. It was developed from the TurtleBot2 Personal Robot, an open-source platform supported by ROS (Figure 1). The robot incorporates an RGB-D exteroceptive sensor, trays, and brackets that allow incorporating new equipment, and as the main component, the iClebo Kobuki mobile base. It integrates proprioceptive sensing, low-level processing, and locomotion systems.

The intermediate tray was removed from the original platform to reduce the height of the robot and keep its center of gravity low. Four 13 mm diameter by 51 mm high aluminium spacers were available, the length of which was extended by 20 mm high acrylic cylinders. These spacers were used to anchor the NUC minicomputer (Robot-PC) to the bottom of the upper tray, for which a 130×150 mm rectangular support base with its fixing screws was provided. The hardware architecture of the RobSLAM system is shown in Figure 2.

Since the 19.1 V DC @ 2 A voltage available on one of the Kobuki base connectors was insufficient to power the NUC minicomputer, a regulated power supply system was designed that includes a 12 V battery @ 5.5Ah, a DC/DC converter to boost the voltage to 19.0 V DC, and a fixing structure to prevent the battery from shifting. The system was anchored to the robot's bottom tray so as not to displace its center of gravity. The design allowed a convenient power supply for the Robot PC and energy autonomy of up to three and a half hours under full battery charge conditions, with sufficient time to advance the proposed experiments and collect data for analysis. Figure 3 shows the final appearance of the mobile robot.

2.5. Criteria for the Selection of vSLAM Methods. This section presents the qualitative criteria that support this study. The types of sensors and detectors/descriptors used by each of the methods and the environment model kind were considered. Finally, the associated specific aspects were presented, such as 2D and 3D modelling support, model dimensions, map reuse, path detection, and global optimization.

Table 3 presents some of the most popular vSLAM algorithms, which are classified according to the type of visual sensors they support and the type of detector/descriptor they use. MonoSLAM was the first real-time vSLAM algorithm to use a camera as the only sensor. In this method, feature-based sparse probabilistic maps are produced by estimating camera states, features, and associated uncertainties. PTAM (parallel tracking and mapping) is a monocular algorithm also used in augmented reality. It often does not deliver full maps but instead produces small maps that include a single physical object in the environment. Both MonoSLAM and PTAM were developed in the last decade and are now considered somewhat long-outdated, which is why they were discarded.

Figure 4 summarizes a classification based on the method of extracting information from the image sequence and the type of map produced. Sparse maps consider small subsets of pixels, precisely those associated with the detected features and their neighborhood. In contrast, dense maps use most or all of the pixels in the image. Two of the most representative vSLAM algorithms for dense mapping are DTAM (dense tracking and mapping) and RTAB-Map (real-time appearance-based mapping), while RTAB-Map is classified as an indirect method, DTAM is considered a direct one. In indirect methods, features of interest are first extracted from the image, and from them, the camera is located and the map is constructed. On the other hand, in direct methods, the intensity of the pixels is considered without taking into account a preliminary stage of feature extraction.

An important matter to consider is that DTAM is designed for small and localized scenes or spaces [20], while RTAB-Map supports 2D and 3D mapping of large dimensions and long term. Another direct method worth highlighting is LSD-SLAM (large-scale direct monocular SLAM), which considers areas with higher intensity gradient than those considered by DTAM. In this sense, LSD-SLAM ignores regions with low or no texture for which it is difficult to estimate depth. For this reason, it is considered a semi-dense method [21]. A recently proposed algorithm is DSO. In this approach, it is sought to minimize the photometric error of the set of pixels between images, instead of minimizing the geometric back-projection error. DSO performs

Table 2: Features of the hardware.

Elements	Technical specifications
Workstation	ASUS N56J series computer with intel® core™ i7-4700HQ CPU @ 2.4 GHz × 8, 8 GB RAM, 1 TB hard drive and NVIDIA® GEFORCE GTX® 760 M GPU
Robot-PC	Intel® NUC6i3SYH mini-computer (19 V DC @ 3.43 A) with intel® Core™ i3-6100U CPU @ 2.3 GHz × 4, 4 GB RAM, 500 GB hard disk
Router	TP-link 3G/4G TL-MR3420
USB camera	Logitech® C270 with video capture up to 1280 × 720 pixels @ 30 fps

Source: [11].

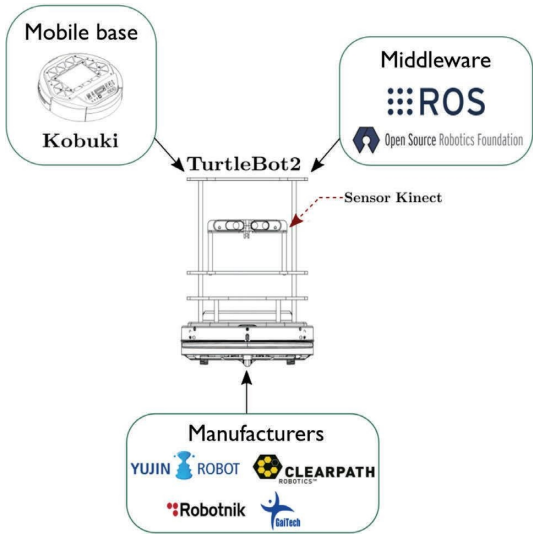


Figure 1: TurtleBot2: base platform of the RobSLAM experimental system.

a joint probability optimization of all model parameters [22, 23]. ORB-SLAM2 is an extension of its predecessor ORB-SLAM [24] that supports not only monocular but also stereo and RGB-D cameras (Table 3). ORB-SLAM2 includes map reuse, closed path detection, and relocation capabilities. It operates in real time on standard processing systems and in a wide variety of environments, from small indoor enclosures with short sequences of images captured with a handheld RGB camera, through industrial environments with sequences captured from drones to large sequences obtained from automobiles driving in urban environments [25]. Based on previous data, the methods selected to conduct the experiments are ORB-SLAM2 in its monocular component [26] and RTAB-Map in its RGB-D component, which are highlighted with green circles in Figure 4. For this purpose, it was considered that [11] both methods are part of the current state of the art in visual SLAM. DSO (direct sparse odometry) is also part of the most recent algorithms; however, it does not perform global optimization and closed-loop detection, both of which are supported by ORB-SLAM2 and RTAB-Map. DTAM is applicable only for localized and small-scale environments, while ORB-SLAM2 and RTAB-Map allow SLAM to be solved for both small and large-scale environments. MonoSLAM does not perform global optimization, and as with PTAM (parallel tracking and mapping), it does not perform closed-loop detection either. Moreover, these methods date back to 2003 and 2007, respectively. The development of RTAB-Map dates back to 2013, and its extension dates back to 2017. Concerning ORB-SLAM, it dates back to 2014 and its extension ORB-SLAM2 dates back to 2016. Although the development of LSD-SLAM (large-scale direct monocular SLAM) dates back to 2013, it is considered a semidense system, located halfway between the sparse and dense paradigms.

2.6. Other Four Reasons Why These Two Methods Are Often Preferred Over Others

2.6.1. Accurate and Robust Performance. RTAB-Map and ORB-SLAM2 have demonstrated high accuracy and robustness in various challenging environments, including low-light conditions, fast movement, and dynamic environments.

2.6.2. Real-Time Performance. Both methods are designed to provide real-time performance, making them suitable for applications where real-time feedback is critical, such as in autonomous vehicles, drones, or industrial automation.

2.6.3. Feature-Based. Both RTAB-Map and ORB-SLAM2 are feature-based SLAM methods, meaning that they rely on extracting distinctive features from the environment to construct a map and estimate the robot's pose. This approach has been shown to be effective in many scenarios.

2.6.4. Availability and Support. Both methods are open-source and have a large community of users and developers, making them well documented and well supported. This ensures that users can access help and resources if they encounter any issues during their application development.

2.7. Disadvantages of RTAB-Map

2.7.1. High Computational Requirements. RTAB-Map can be computationally intensive, particularly when using large datasets. This can result in slower processing times and higher hardware requirements.

2.7.2. Difficulty with Large-Scale Environments. RTAB-Map may struggle to accurately map large-scale environments due to the accumulation of errors over time.

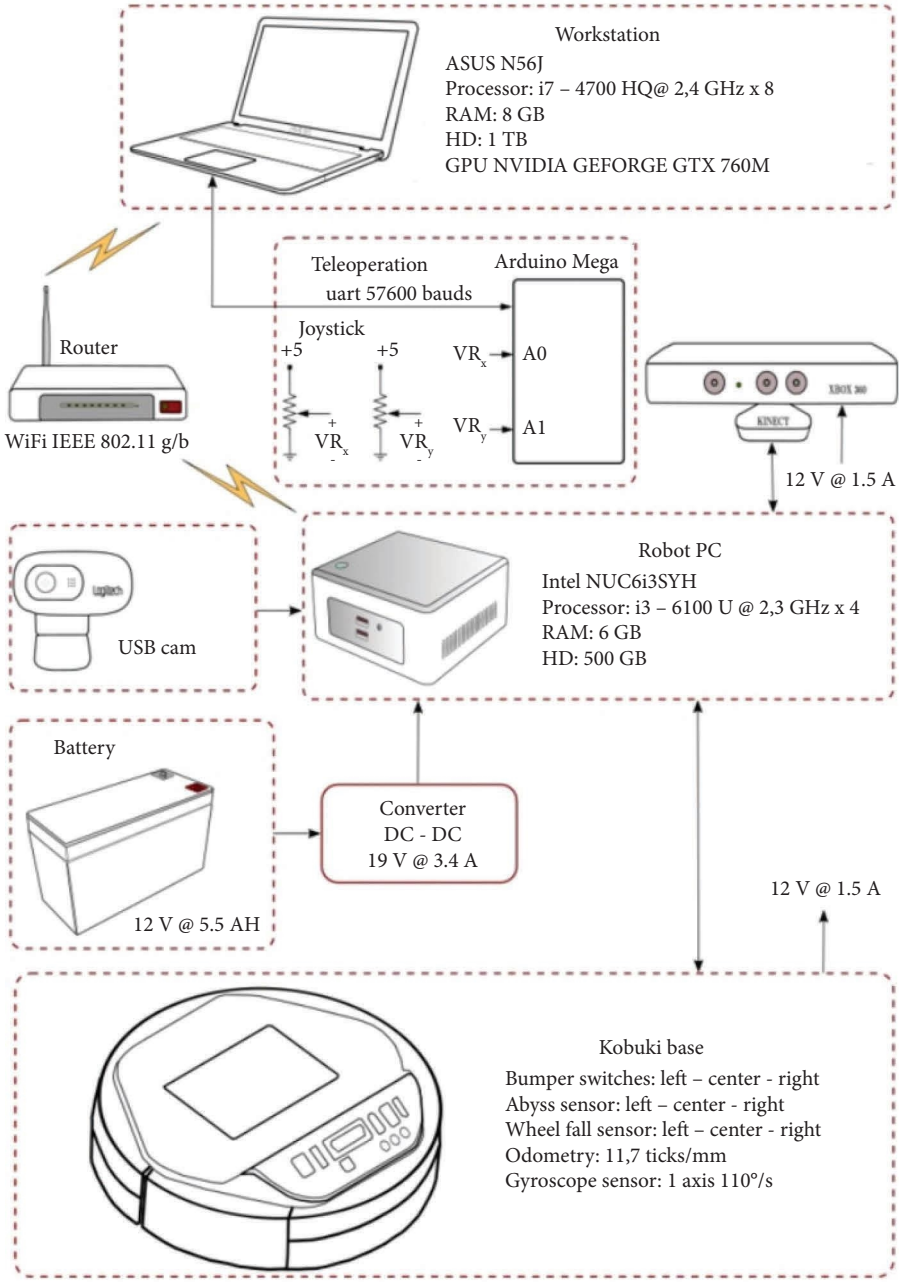


Figure 2: RobSLAM robotic system hardware architecture.

2.7.3. Limited Accuracy. RTAB-Map may not produce as accurate maps as some other SLAM methods, particularly in dynamic environments where the scene is constantly changing.

2.8. Disadvantages of ORB-SLAM

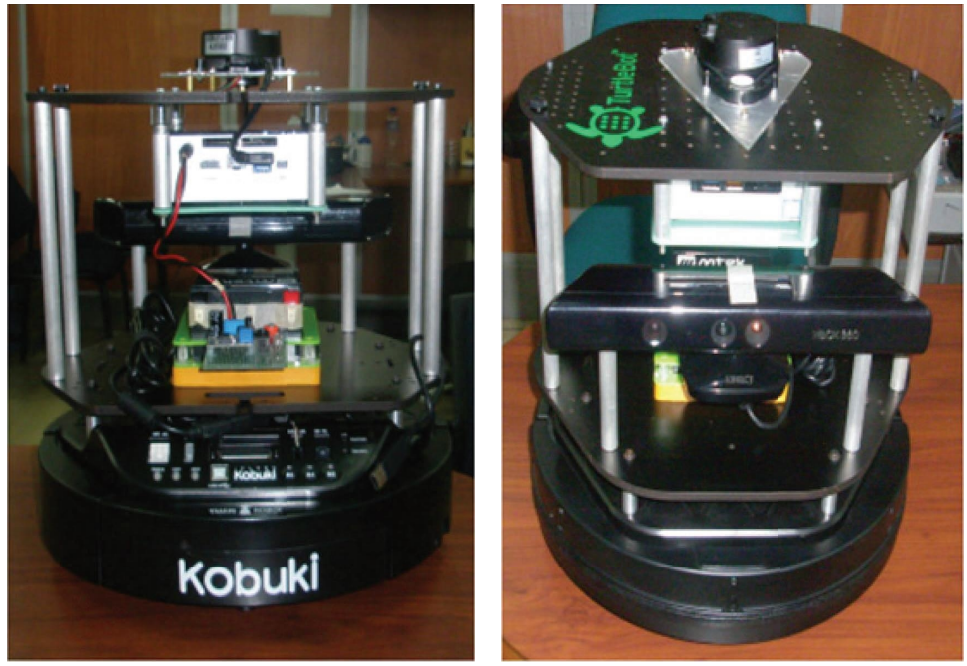
2.8.1. Limited Robustness. ORB-SLAM can struggle in low-texture environments or when there are few distinctive features, as it relies heavily on feature detection and matching.

2.8.2. Difficulty with Dynamic Scenes. ORB-SLAM may struggle to accurately track the location of the camera in dynamic scenes, where objects are moving and the environment is constantly changing.

2.8.3. Difficulty with Large-Scale Environments. ORB-SLAM may struggle to accurately map large-scale environments, particularly when there are many repeated patterns or textures.

3. Results and Discussions

Visual SLAM experiments are conducted using the selected methods (RTAB-Map and ORB-SLAM), and their advantages and disadvantages are weighed. The experiments were conducted in an indoor office environment, whose plan is



(a) (b)

Figure 3: Mobile robot component of the RobSLAM system. (a) Rear view where the regulated power system can be seen and (b) front view where the robot-PC and the Kinect sensor can be seen.

Table 3: Visual SLAM algorithms' classification.

Algorithm	Sensors	Detector/descriptor
MonoSlam	Mono	Shi-Tomasi
PTAM	Mono	FAST
DSO	Mono, stereo	N.A. degree of intensity
ORB-SLAM2	Mono, stereo, RGB-D	ORB
LSD-SLAM	Mono	N.A., semidense
RTAB-map	Stereo, RGB-D	SURF, bag-of-words
DTAM	Mono	N.A., pixel-wise

N.A.: not applicable. It does not use a detector/descriptor because it is a direct method.

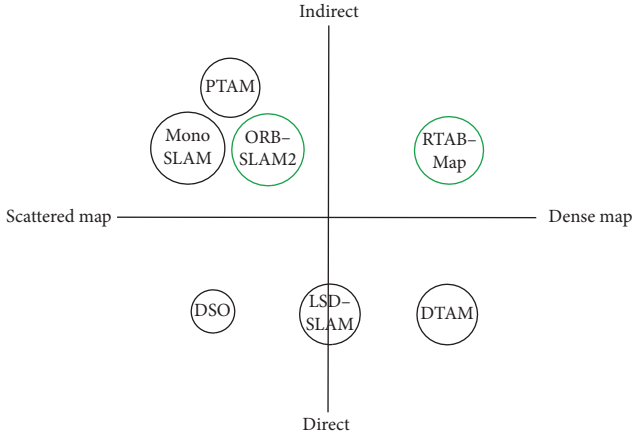


Figure 4: Some of the most recent open source vSLAM methods.

shown in Figure 5. The space consists of 14 modules of 2.44 m × 1.75 m located on either side of a central corridor in which two beams are located. Each cubicle is denoted by a literal (A to N), and there is only one access door located at one end of the enclosure. In the path of robot movement, cubicles *H* and *J* were considered in the section of the corridor that connects them.

3.1. ORB-SLAM2 Hand-Held Experiment. Figure 6 shows the test environment for the ORB-SLAM2 algorithm in its monocular component together with its dimensions. It corresponds to the desk located in cubicle *J* of the office space in Figure 5. Some of the possible positions adopted by the camera during the experiment are indicated from the starting point to the endpoint.

The experiment is conducted by holding the camera with one hand and following a trajectory close to the one indicated by the orange dashed line in Figure 6. This type of SLAM experiments is known in the literature as hand-held reconstruction [27, 28]. During the path, the camera was aimed at the desk at all times, at a height of approximately 130 cm with respect to the floor. Different objects were placed on the desk to provide texture to the algorithm and the possibility of detecting and tracking different features. A round-trip path was designed, starting at the point labelled start, on the far left of Figure 6, advancing to the point labelled end, and finally returning to the starting point.

Figure 7 shows the working environment for this experiment, which includes a desktop and different objects placed on it, providing texture and a variety of features to the algorithm. The experimental setup for the SLAM experiment of the considered environment is shown in Figure 8. In this case, the moving platform is not required since the camera moves manually over the scene.

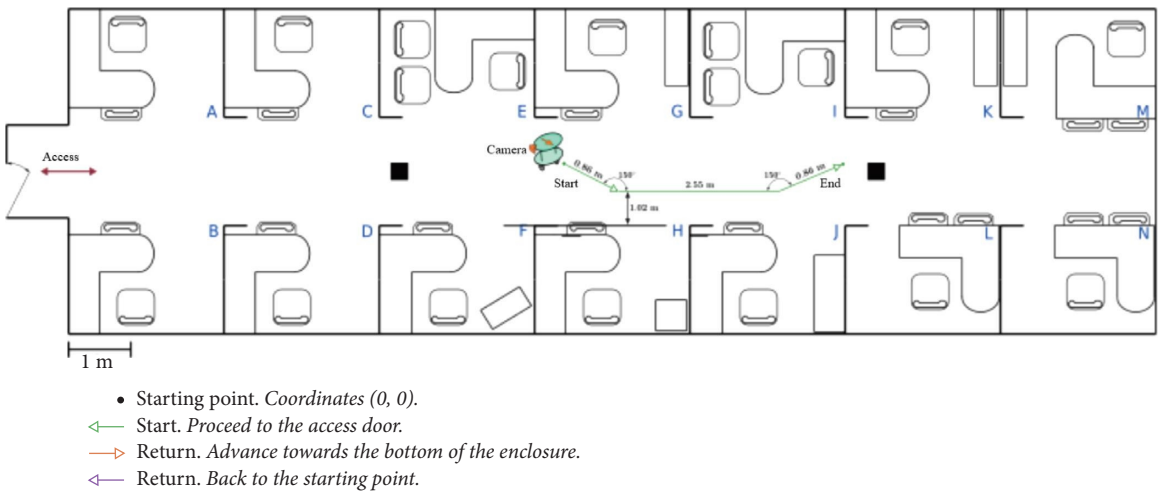


Figure 5: Architectural plan of the experimental environment and route designed for the teleoperation of the robot.

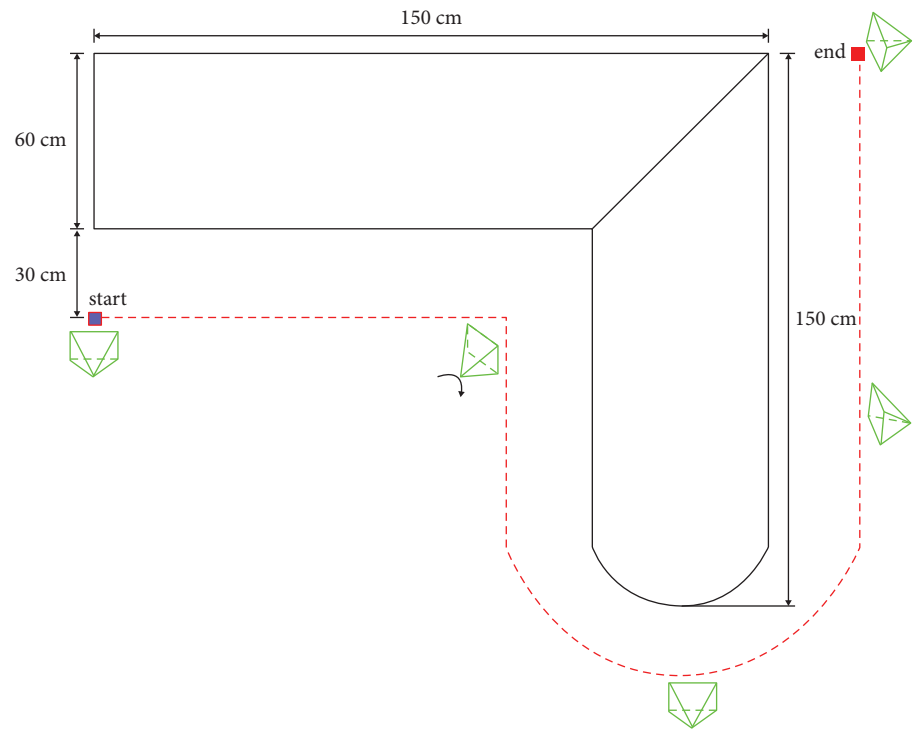


Figure 6: Desktop environment in "L" pattern for experiments with the ORB-SLAM2 algorithm in its monocular component. Some of the positions that the camera can take (in green) are indicated during tours that approach the trajectory considered in orange (dotted line).

Figure 9 shows the map displayed by the ORB-SLAM2 algorithm by using its monocular component, according to the criteria of [25]. The library can be compiled with or without ROS; in this experiment, it is used in combination with this middleware. To run ORB-SLAM2 with ROS, we proceed to open from the workstation PC three terminals with remote access from the NUC PC via SSH; thus, it is possible to monitor the process from the workstation PC (Figure 8). Subsequently, the commands in Table 4 are entered in the remote terminals:

The first two commands are straightforward. The first one executes the ROS Master, and the second one launches the usb_cam_node, a ROS driver for USB cameras, in this case the Logitech C270 device shown in Figure 8. This last command launches the Mono node of the ORB-SLAM2 package, which starts mapping the environment from the sequence of images provided by the camera. In this way, a real-time monocular SLAM process is implemented. The command requires two parameters. The first parameter indicates the location of the bag-of-words vocabulary contained in the OR-Bvoc.txt file, and the second one indicates the location of the ASUS.yaml file containing the camera parameters.

The mapping starts by extracting ORB features from the sequence of images to build an initial map (map initialization) on which to estimate the camera position. In the first

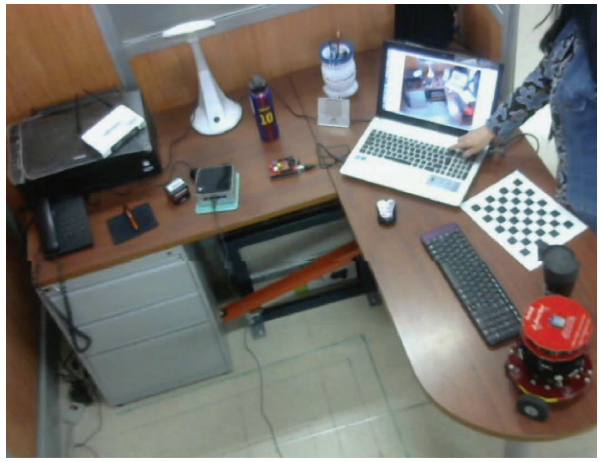


Figure 7: The objects arranged on the desktop, a work environment that offers different possibilities for detecting and tracking several representative features.

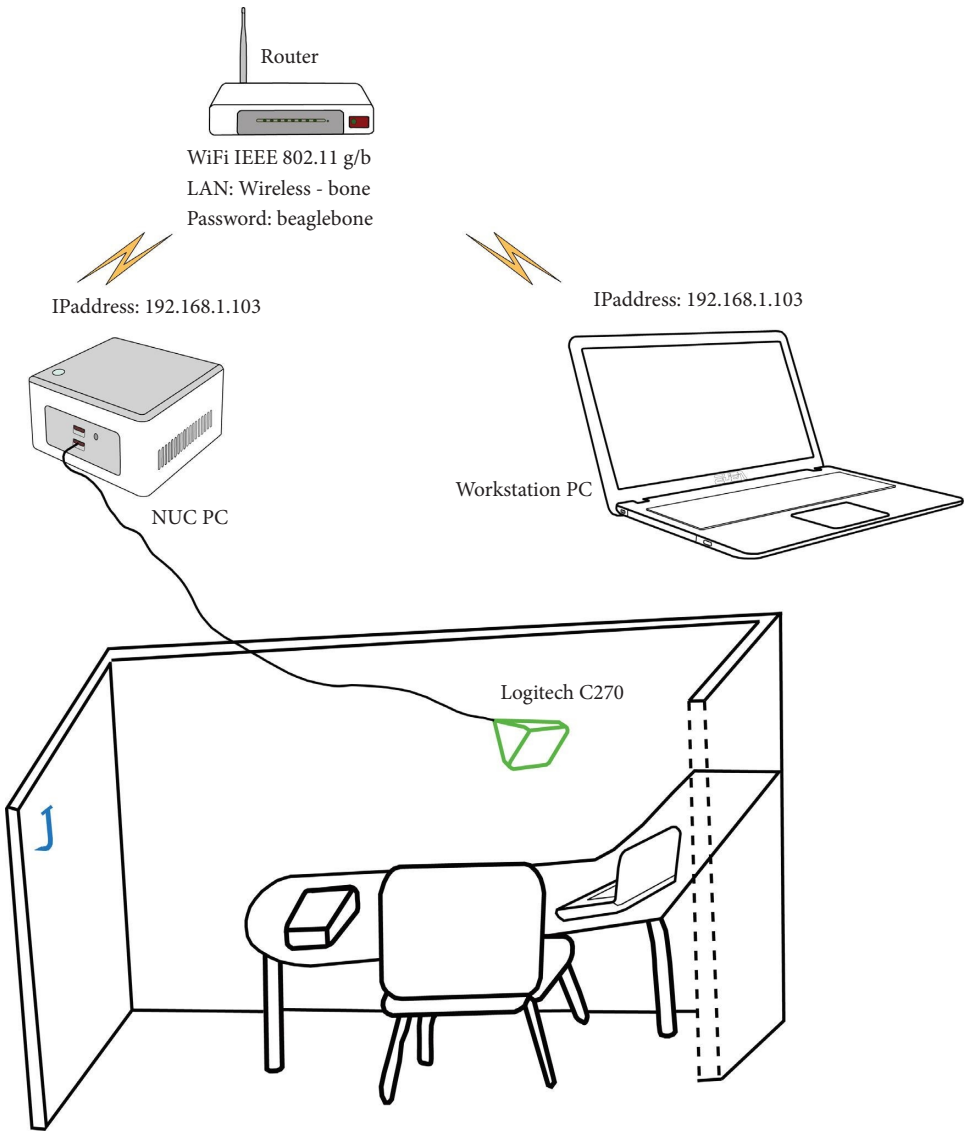
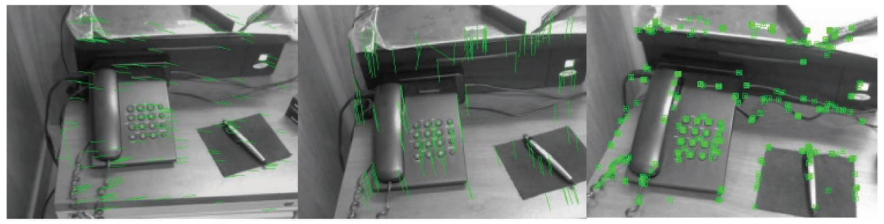


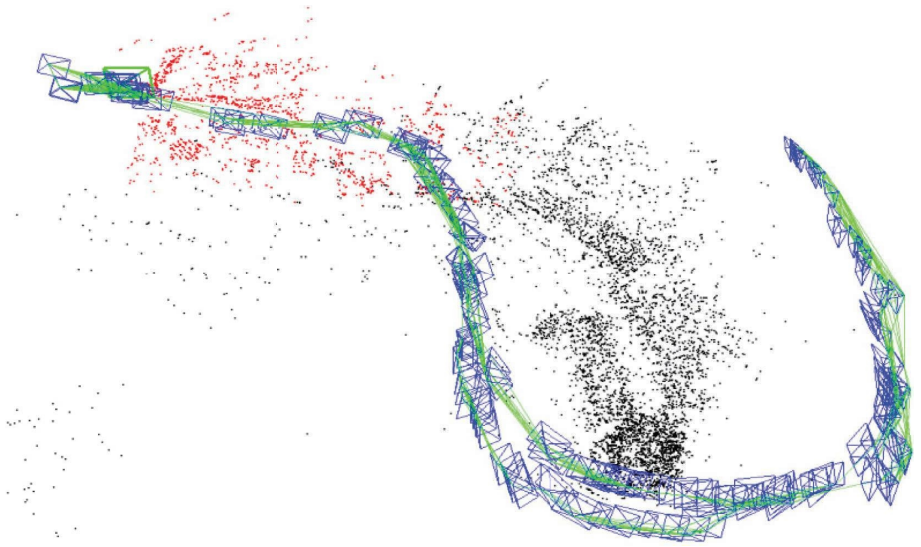
Figure 8: Experimental setup for a hand-held mapping with the ORB-SLAM2 method.

Table 4: Commands to run ORB-SLAM2 using ROS.

Task	Commands
T1	$roscore
T2	$rosrun usb_cam usb_cam_node usb_cam/image_raw:=/camera/image_raw
T3	Rosrun ORB_SLAM2 mono\ /Home/turtlebot/ORB_SLAM2/vocabulary/ORBvoc.txt\ /Home/turtlebot/ORB_SLAM2/examples/ROS/ORB_SLAM2/Asus.yaml



(a)



(b)

Figure 9: Sparse map made with ORB-SLAM2. In (a), the detection of characteristics prior to the initialization of the map (green line segments) and their follow-up after drawing up the first map (small rhom-buses enclosed by squares) is observed. In (b), the scattered point cloud corresponding to the map can be seen.

two images of Figure 9(a) (upper left corner), we can see how the system detects a set of relevant features of the environment, which is indicated by the green line segments on the observed objects. At this stage, the GUI (graphical user interface) of the system displays the message "trying to initialize" in the image flow window. Once the algorithm manages to build an initial map, it proceeds to estimate the location of the camera on it. The system is now in the SLAM mode, which is indicated by displaying the message "SLAM mode" in the image flow window. In addition, the feature signaling changes to small diamonds enclosed by squares, as seen in the third image of Figure 9(a).

Each point p_j of the ORB feature cloud stores the three-dimensional position x_{wj} in the world coordinate system [24]. The camera direction n_j corresponds to the mean unit vector extracted from the rays joining the point with the optical center of the observed keyframes. A representative D_i descriptor, corresponding to the ORB descriptor whose hamming distance is the minimum between all the keyframes where the point is observed. The maximum and minimum distances ($d$max and $d$min) from which the point can be observed, according to the scale invariance limits of the ORB features.

Each keyframe stores T_{iw} position, which corresponds to a rigid body transformation of the world points to the camera coordinate system [24]. The intrinsic parameters of the camera, which include the focal length and the coordinates of the main point. All ORB features extracted from the image, whether or not associated with a map point and whose coordinates are undistorted or, if distorted, a distortion model is available.

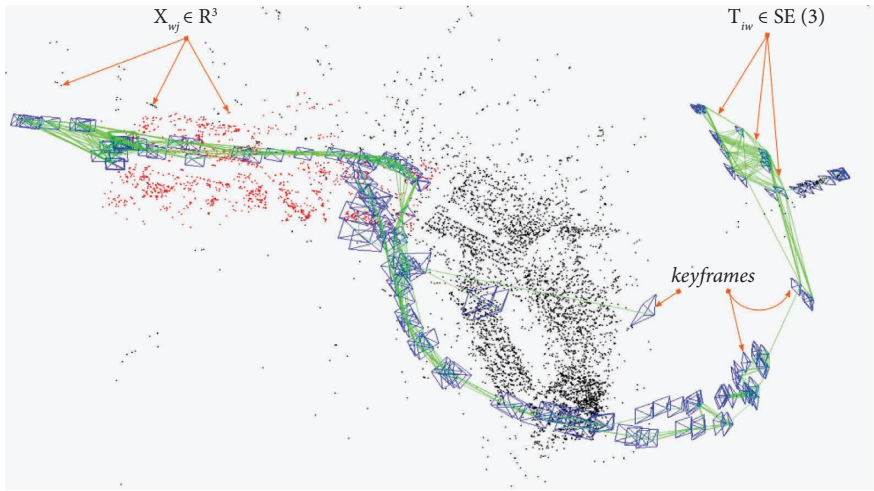


Figure 10: Interpretation of a sparse map made with ORB-SLAM2.

Figure 10 summarizes the previous information. The x_{wj} coordinates of the red and black points are part of the cloud. The red points correspond to the current local map and the black points to the global map. The keyframes are shown in blue, and the estimated T_{iw} trajectories of the camera are shown in green.

ORB-SLAM2 is a GraphSLAM method in which closed-loop detection is performed based on the optimization of the position graph (Figure 4). The optimization problem of the 3D structure, the camera position, and its intrinsic parameters are solved by employing iterative nonlinear optimization, which aims to minimize the distance between the back projection of the 3D model and the associated points in the image [29]. In the position graph, the keyframes correspond to the nodes and the camera trajectories correspond to the arcs (Figure 10). Figure 11 establishes the correspondence of some objects in the scene with the sparse point cloud or map.

3.2. RTAB-Map Hand-Held Experiment. The same "L" shaped desktop environment (Figure 5) was considered to perform hand-held mapping experiments with RTAB-Map. In this particular case, a Kinect sensor was used, which delivers both depth and RGB images. Figure 12(a) shows one of the 3D maps obtained, highlighting the dense point cloud that gives a realistic appearance to the model. The pose graph is shown in Figure 12(b) together with the initial and final coordinates of the frame linked to the sensor.

The graph nodes store camera odometry, along with visual information from depth images, RGB images, and SURF features quantized into an incremental visual dictionary (visual words), used by the closed-loop detection algorithm [30, 31]. The algorithm allows establishing matches between the current camera location and previous locations of already visited sites employing the bag-of-words method [32].

Figure 13 shows the visual odometry and closed loop detection implemented by the RTAB-Map method.

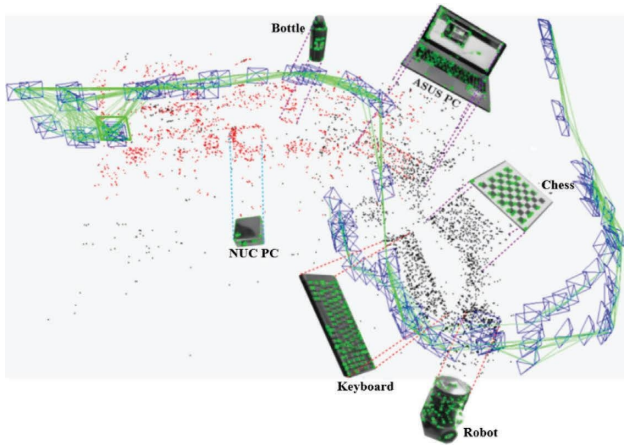
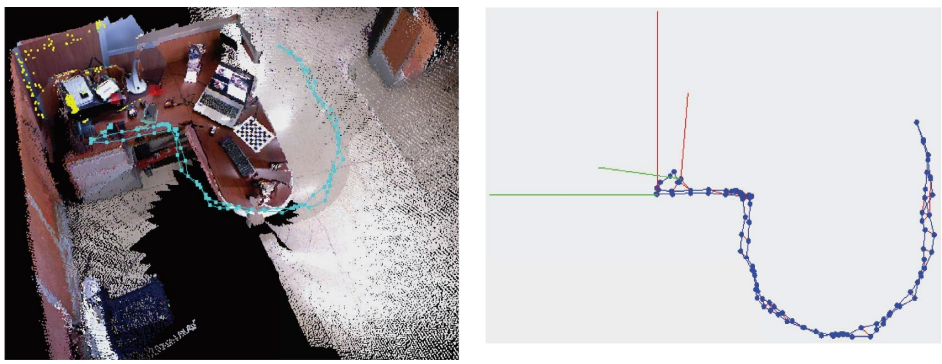


Figure 11: Object identification in the ORB point cloud.

3.3. Dispersed Mapping Experiment with the RobSLAM System. Figure 14 shows the working environment for the visual SLAM mapping experiments conducted with the RobSLAM system. Cubicles *H* and *J* of the enclosure in Figure 5 were considered. The designed path, which consists of three sections, is indicated in green. The first section of 0.86 m with an orientation of 150° with respect to a line parallel to the outer edge of the cubicles, the second section, on the same parallel, at a distance of 1.02 m with respect to the edge and 2.55 m long, and finally, a third section that is symmetrical with the first and of the same length.

Monocular SLAM systems estimate depth by triangulating corresponding key points in consecutive images, thus requiring lateral camera displacements that generate baselines of sufficient length for depth calculation (Figure 15). For such a reason, for robot displacements only forward or backward, they tend to fail [33]. In order to allow lateral displacements, the camera was located on the right side of the mobile robot, as shown in Figure 14.

The robot was teleoperated along the trajectory shown in Figure 14 using the hardware designed for this purpose, which is shown in Figure 2 of the Materials and Methods



(a) (b)

Figure 12: Map made with RTAB-Map. In (a), the dense point cloud gives the model a realistic appearance. The position graph in (b) shows the edges or estimated trajectory of the camera (line segments in blue), nodes (blue points), and the detection of closed loops (line segments in red).

(a) (b)

Figure 13: Visual odometry (a) and closed loop detection (b) carried out by the RTAB-Map system.

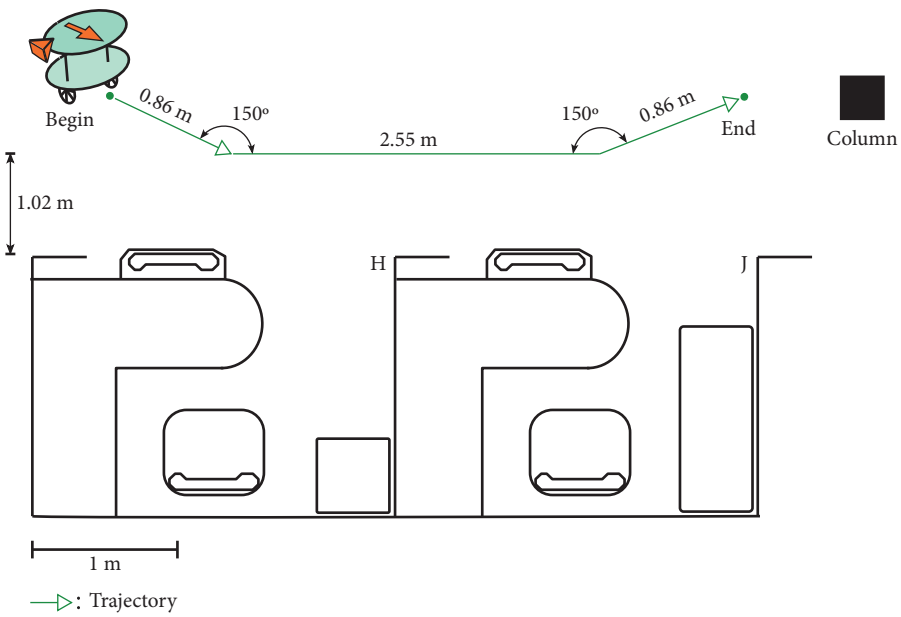


Figure 14: Path for mapping with ORB SLAM2. The total length of the route is 4.27 m.

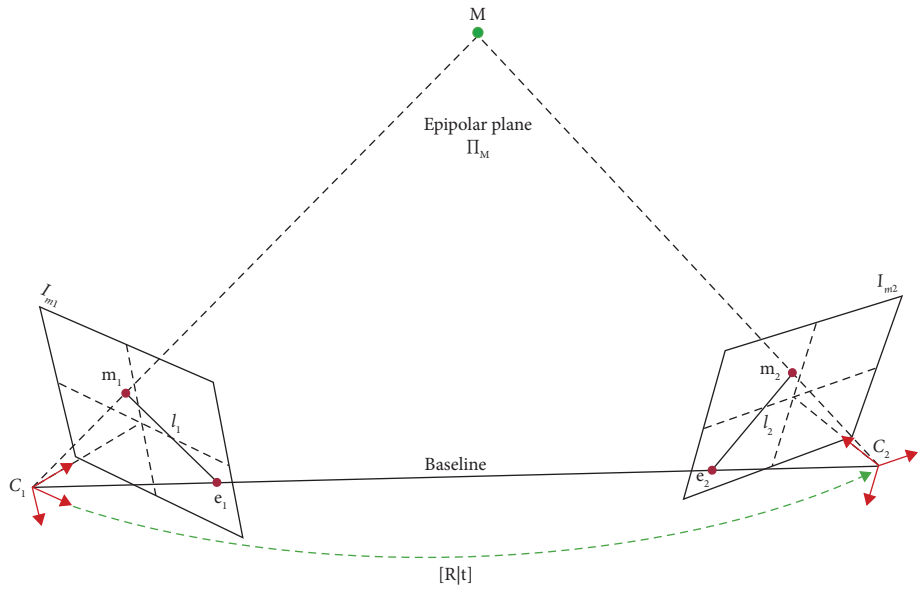


Figure 15: Main elements of epipolar geometry.

section. Figure 16 shows some stages of the map construction process with ORB-SLAM2. Figure 16(a) shows the beginning of the map construction (right), being necessary to incorporate a chess pattern impression to the first division observed by the camera (left). In fact, given the lack of texture of the divisions found in the environment, it was essential to locate this type of pattern in several of them; otherwise, the algorithm would not identify a sufficient number of features to perform the tracking.

As the robot teleoperation progresses, the trajectory of the camera and the 3D map are simultaneously estimated (Figures 16(b) and 16(c)). As some features are left behind, new ones are incorporated for tracking. Sites already visited and whose features are not part of the current scene appear in black in the point cloud and become part of the global map. The red points in the cloud are part of the features of the local map that is in the making.

Figures 16(d) and 16(e) show the mapping and localization process almost at the end of the tour. It can be seen how the estimation of the camera trajectory fits the designed tour and how the 3D reconstruction reflects the geometrical structure of the environment, in general terms. The final map is shown in Figure 17, which also shows a checkerboard pattern attached to a desk, adding texture to the scene.

Figure 18 shows that although the estimated trajectory of the camera is close to the robot's odometry (pink line), this estimate does not correspond to the robot's tour, which is indicative of the uncertainty in both estimates.

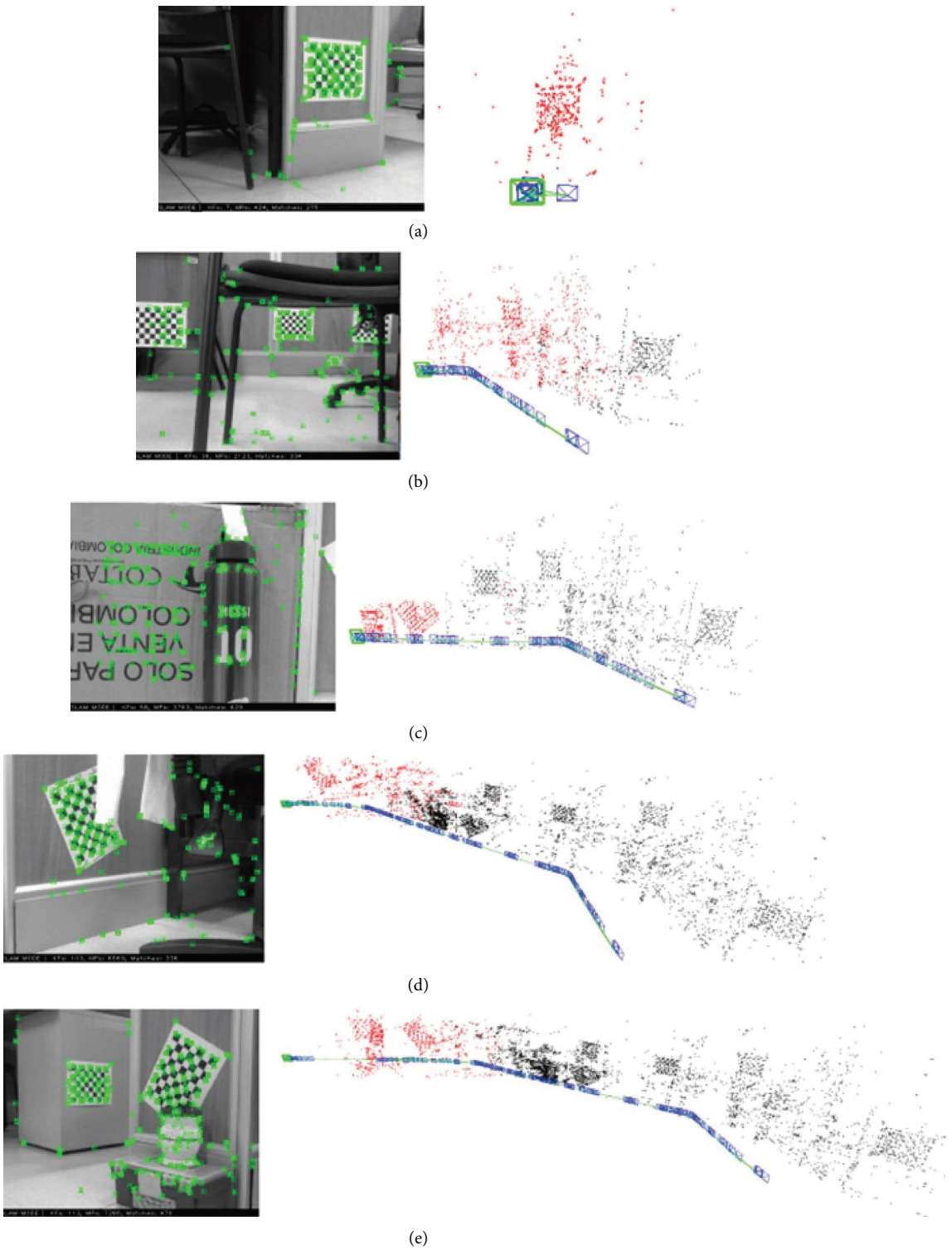
In this section, the algorithms ORB-SLAM2 and RTAB-Map were employed in hand-held mapping experiments, in which the sensor was manually moved through a working environment consisting of an "L" desk, obtaining three-dimensional maps with sparse and dense point clouds, respectively. The environment provided a sufficient and varied number of features to maintain visual odometry and to be able to elaborate its three-dimensional model [30].

An additional experiment using the RobSLAM system and the ORB-SLAM2 algorithm was conducted in an office building. A teleoperation of the robot was performed along two contiguous modules with objects such as chairs, tables, and partitions with wood veneer. In this case, a lack of texture was evident, mainly in the partitions, which caused a frequent loss of visual odometry and a failure in the experiment, even when the robot was slightly moved back to try to recover the odometry.

One solution to the problem of the lack of texture was to adhere to a checkerboard pattern to the divisions, such as those used in camera calibration. Although feature tracking and visual odometry were improved, a second problem was detected. It consisted of the fact that the boards observed in the middle of the tour were considered as previously visited sites. Indeed, the boards observed at the beginning of the survey and those observed later were considered as closed loops by the parallel loop detection process of ORB-SLAM2. This problem led to false camera location estimates and maps that did not reflect the geometrical properties of the environment. It was solved by rotating some of the boards and adding new objects to the scene. With this, it was finally possible to obtain consistent three-dimensional maps.

Both ORB-SLAM2 and RTAB-Map are GraphS-LAM algorithms, from which the following are obtained: (1) a point cloud and (2) a position graph. However, while ORB-SLAM2 produces sparse maps, RTAB-Map delivers a dense point cloud, which implies the need for hardware with higher processing capabilities.

The main advantages of ORB-SLAM2 are as follows: (1) the map can be initialized with only the first pair of images and (2) it is globally consistent, yet closing loops can result in large jumps in position estimation. Disadvantages include the following: (1) a trained bag-of-words vocabulary is required, (2) the localization mode requires an integrated map, and it is highly dependent on the relocation, and (3) the



(a)

(b)

(c)

(d)

(e)

Figure 16: Different stages in the process of building a 3D-map with the RobSLAM system and the monocular algorithm in real-time ORB-SLAM2. Note: the incorporation of checkerboard-like patterns to add texture to the modular partitions of the enclosure.

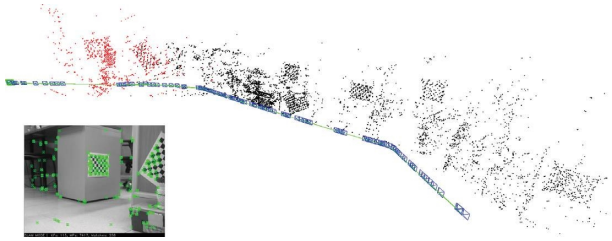


Figure 17: Final map produced by ORB-SLAM2 at the end of the tour.

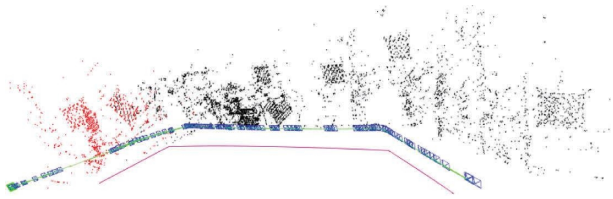


Figure 18: Final map obtained with ORB-SLAM2.

absolute scale is observable for the stereo and RGB-D components of the algorithm but unknown for the monocular.

4. Conclusions

The growing worldwide demand for service, logistics, personal assistance, and companion robots, as well as the need to increase their levels of autonomy has made SLAM a key player in robotics. In addition, its methods have been extended to the autonomous driving of all types of vehicles: land, air, and water-borne.

This article considered some of the algorithms that constitute the current state of the art in visual SLAM. Systematic review and comparison of two of the most representative open source algorithms for solving SLAM led to the choice of the ORB-SLAM2 and RTAB-Map methods.

Both algorithms were employed in hand-held mapping experiments, in which the sensor was manually moved around the environment to be mapped, which was the "L" desk of an office, providing a convenient number of features for visual odometry.

An additional experiment was conducted with the ORB-SLAM2 algorithm based on the lateral arrangement of a camera in the RobSLAM system and its teleoperation on a predefined trajectory.

The lack of texture of partitions and panels led to loss of visual odometry and detection of closed loops in previously unobserved parts of the environment. Both situations led to failures in the algorithm execution, which was finally solved by adhering to some checkerboard patterns with different orientations on the partitions and adding mark-rich and texture-rich objects in the scene.

This work focused on the study of the problem of simultaneous mapping and localization in unknown environments or SLAM, specifically on the state-of-the-art open-source algorithms available for three-dimensional environment modelling. Also, a robotic system (RobSLAM) was designed and built for the experimental validation of the considered algorithms. The system is versatile enough to support the research and development of robotic applications, offering good energy autonomy and extended capabilities of processing and external sensing. RobSLAM takes advantage of the hardware and software abstraction features offered by ROS, which is one of the fastest growing and most pervasive middleware in the robotics community.

Data Availability

The data are available upon request from the corresponding author.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this paper.

Acknowledgments

The authors express their gratitude for the financial, academic, and research support to Universidad Nacional de Colombia (grant 3136791507) and Politécnico Colombiano Jaime Isaza Cadavid (grant 3052158705).

References

[1] O. Roesler and V. Ravindranath, "Evaluation of SLAM algorithms for highly dynamic environments," *Advances in Intelligent Systems and Computing*, vol. 1093, pp. 28–36, 2020.

[2] R. Negenborn, "Robot localization and kalman filters–on finding your position in a noisy world," M.Sc. thesis, Utrecht University, Netherlands, 2003.

[3] P. Guan, Z. Cao, E. Chen, S. Liang, M. Tan, and J. Yu, "A real-time semantic visual SLAM approach with points and objects," *International Journal of Advanced Robotic Systems*, vol. 17, no. 1, Article ID 172988142090544, 2020.

[4] Y. Fan, Q. Zhang, Y. Tang, S. Liu, and H. Han, "Blitz-SLAM: a semantic SLAM in dynamic environments," *Pattern Recognition*, vol. 121, Article ID 108225, 2022.

[5] V. Scilimati, A. Petitti, P. Boccadoro et al., "Industrial internet of things at work: a case study analysis in robotic-aided environmental monitoring," *IET Wireless Sensor Systems*, vol. 7, no. 5, pp. 155–162, 2017.

[6] M. Shneier and R. Bostelman, "Literature review of mobile robots for manufacturing," Technical report, National Institute of Standards and Technology, U.S. Department of Commerce, Gaithersburg, MD, USA, 2015.

[7] N. Mohamed, J. Al-Jaroodi, and I. Jawhar, "Middleware for robotics: a survey," in *Proceedings of the 2008 IEEE Conference on Robotics, Automation and Mechatronics*, pp. 736–742, Chengdu, China, September, 2008.

[8] A. Serrate, "Método para el Desarrollo de Aplicaciones en Robótica de Servicios basado en Industria 4.0 y Computación en la Nube," Master´s thesis, Universidad Nacional de Colombia, Colombia, 2021.

[9] A. Staranowicz and G. Mariottini, "A survey and comparison of commercial and open-source robotic simulator software," in *Proceedings of the 4th International Conference on Pervasive Technologies Related to Assistive Environments*, pp. 1–56, Crete Greece, May, 2011.

[10] K. Takaya, T. Asai, V. Kroumov, and F. Smarandache, "Simulation environment for mobile robots testing using ROS and Gazebo," in *Proceedings of the 20th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 96–101, Sinaia, Romania, October, 2016.

[11] G. Acosta, "SLAM Monocular en Tiempo Real," Ph. D. Thesis, Universidad Nacional de Colombia, Colombia, 2019.

[12] R. Halder, J. Proença, N. Macedo, and A. Santos, "Formal verification of ROS-based robotic applications using timed-automata," in *Proceedings of the 2017 IEEE/ACM 5th International FME Workshop on Formal Methods in Software Engineering (FormaliSE)*, pp. 44–50, Buenos Aires, Argentina, May, 2017.

[13] R. Walenta, T. Schellekens, A. Ferrein, and S. Schiffer, "A decentralised system approach for controlling AGVs with ROS," in *Proceedings of the 2017 IEEE AFRICON*, pp. 1436–1441, Cape Town, South Africa, September, 2017.

[14] A. Koubâa, M. Sriti, Y. Javed et al., "Turtlebot at office: a service-oriented software architecture for personal assistant robots using ROS," in *Proceedings of the 2016 International Conference on Autonomous Robot Systems and Competitions (ICARSC)*, pp. 270–276, Braganca, Portugal, May, 2016.

[15] Y. Quiñonez, I. Tostado, and C. Burgueño, "Application of evolutionary techniques and computer vision for the autonomous robots navigation using a Turtlebot 2," *Revista Ibérica de Sistemas y Tecnologías de la Información*, no. E3 (03), pp. 93–105, Article ID 1690371427, 2015, https://www.risti.xyz/issues/ristie3.pdf.

[16] M. Quigley, B. Gerkey, K. Conley et al., "ROS: an open-source robot operating system," in *Proceedings of the Open-source Software Workshop of the International Conference on Robotics and Automation*, pp. 1–6, Stuttgart, Germany, October, 2009.

[17] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an opensource multi-robot simulator," in *Proceedings of the 2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566) (IROS)*, vol. 3, pp. 2149–2154, Sendai, Japan, October, 2004.

[18] I. Afanasyev, A. Sagitov, and E. Magid, "ROS-based SLAM for a Gazebo simulated mobile robot in image-based 3D model of indoor environment," in *Proceedings of the International Conference on Advanced Concepts for Intelligent Vision Systems*, pp. 273–283, Auckland, New Zealand, October, 2015.

[19] E. Eaton, C. Mucchiani, M. Mohan, D. Isele, J. Luna, and C. Clingerman, "Design of a low-cost platform for autonomous mobile service robots," in *Proceedings of the Workshop on Autonomous Mobile Service Robots, IJCAI*, pp. 1–7, New York, NY, USA, July, 2016.

[20] S. Krig, *Computer Vision Metrics: Survey, Taxonomy, And Analysis*, Apress Open, New York, NY, USA, 2014.

[21] T. Taketomi, H. Uchiyama, and S. Ikeda, "Visual SLAM algorithms: a survey from 2010 to 2016," *IPSJ Transactions on Computer Vision and Applications*, vol. 9, no. 1, pp. 16–26, 2017.

[22] J. Engel, V. Koltun, and D. Cremers, "Direct sparse odometry," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 3, pp. 611–625, 2018.

[23] J. Engel, J. Stückler, and D. Cremers, "Large-scale direct SLAM with stereo cameras," in *Proceedings of the 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 1935–1942, Hamburg, Germany, October, 2015.

[24] R. Mur-Artal, J. Montiel, and J. Tardós, "ORB-SLAM: a versatile and accurate monocular SLAM system," *IEEE Transactions on Robotics*, vol. 31, no. 5, pp. 1147–1163, 2015.

[25] R. Mur-Artal and J. Tardós, "ORB-SLAM2: an open-source SLAM system for monocular, stereo, and RGB-D cameras," *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255–1262, 2017.

[26] S. Vithalani, S. Soni, and P. Rajpura, "Autonomous navigation using monocular ORB SLAM2," *Lecture Notes in Electrical Engineering*, vol. 618, pp. 59–68, 2020.

[27] B. Ummenhofer and T. Brox, "Dense 3D reconstruction with a hand-held camera," in *Lecture Notes in Computer Science*, , pp. 103–112, Springer, Berlin, Germany, 2012.

[28] M. Senthilvel, R. Soman, and K. Varghese, "Comparison of handheld devices for 3D reconstruction in construction," in *Proceedings of the 34th International Symposium on Automation and Robotics in Construction, (ISARC 2017)*, pp. 1–8, Taipei, Taiwan, July, 2017.

[29] T. Pire, "Localización y Mapeo Simultáneos mediante el Uso de un Sistema de Visión Estéreo," Ph. D. thesis, Universidad de Buenos Aires, Argentina, 2017.

[30] W. Aguilar, G. Rodríguez, L. Álvarez, S. Sandoval, F. Quisaguano, and A. Limaico, "Real-time 3D modelling with a RGB-D C and on-board processing," in *Proceedings of the International Conference on Augmented Reality, Virtual Reality and Computer Graphics*, pp. 410–419, Sendai, Japan, September, 2017.

[31] G. Popović, I. Cvišić, G. Écorchard, I. Marković, L. Přeučil, and I. Petrović, "Human localization in robotized warehouses based on stereo odometry and ground-marker fusion," *Robotics and Computer-Integrated Manufacturing*, vol. 73, 2022.

[32] K. Liu and R. Mulky, "Enabling autonomous navigation for affordable scooters," *Sensors*, vol. 18, no. 6, pp. 1–21, 2018, https://www.mdpi.com/1424-8220/18/6/1829.

[33] J. Frost, "Robust and scalable visual simultaneous localization and mapping in indoor environments using RGBD cameras," Ph. D. thesiss, University of Lübeck, Germany, 2017.