Acta Scientific COMPUTER SCIENCES

Volume 5 Issue 4 April 2023

Research Article

Operative Guide for 4-wheel Summit XL Mobile Robot Set-up

Manex Ormazabal Arregi and Emanuele Lindo Secco*

Robotics Lab, School of Mathematics, Computer Science and Engineering, Liverpool Hope University, UK

***Corresponding Author:** Emanuele Lindo Secco, Robotics Lab, School of Mathematics, Computer Science and Engineering, Liverpool Hope University, UK.

Received: February 20, 2023

Published: March 10, 2023

Abstract

The aim of this work is to present the main feature of a commercial 4-weeled robot, namely the Summit XL Mobile Robot. The main Hardware and Software components of the robot are presented together with a set of ROS (Robot Operating System) instructions on how this device can be easily set-up in an operative condition where the end-user will be able to wirelessly control the robot movements as well as the mobile camera which is embedded in the front of the device. Real-time streaming of visual information from the camera will be also set-up.

The proposed solution is of interest for a variety of applications, including the exploration and inspection of environments which are characterized by harsh conditions where humans are involved on search and rescue operations. Moreover, the proposed solution can be implemented in other scenarios where it is required a 27/7 monitoring of the environment (i.e. factory inspection, human-robot interaction).

Keywords: Mobile Robots; Real-Time Wireless Control; Robot Operating System (ROS); Video Streaming

Introduction

The robotics field had its greatest impact on the manufacturing industry area, with the introduction of the robotic arms, which are capable of performing repetitive tasks in the assembly line such as pick and place, welding, or assembling [1]. However, these robots had the disadvantage of being positioned in a fixed place and hence, having a limited range of motion [1]. After a few years, with the growth of the technology, the field of mobile robots began, which brought the solution for the mobility of the industrial robot arms and for more applications out of the industrial sector.

The field of mobile robotics has grown incredibly in recent years. Nowadays, these robots are used in almost every kind of sector, and some of the most common applications are for industry, hospitals, exploration, hospitality, or rescuing. A mobile robot, as its name suggests, is a mechanical system capable of moving in its environment. These robots can be autonomous, which means that they are capable of doing tasks autonomously without any human intervention and navigating autonomously around their environment using actuators which allow them to move, sensors to detect and avoid obstacles, and thus, determine its location and an intelligent software. Nevertheless, although most mobile robots use autonomous navigation, they can also be teleoperated from a remote controller in an autonomous manner [2].

There are many types of mobile robots either depending on the kind of environment they are designed for and the type of locomotion they use [3]. Therefore, depending on the type of locomotion, today there are robots that can walk, jump, run, slide, skate, swim, fly, or roll [1,4]. The most common and used mobile robots are the Unmanned Aerial Vehicles better known as UAV-s or drones, and especially wheeled robots, which are used in most of the sectors.

The wheel locomotion mechanism is the most used in mobile robots, which with only a simple mechanical system, these robots can achieve very good efficiencies [1]. Despite the different design of wheeled robots [4], thanks to their robust layout and mechanism, they can carry heavy elements from one place to another, collect objects with an integrated robotic arm, clean the floor, or cut the grass by themselves, which makes them very useful for most human daily tasks.

The research of wheeled robots focuses mainly on the problems of traction and stability, maneuverability, and control. Depending on the locomotion of wheeled robots, they are classified according to the different types of wheel systems, which can be a standard wheel, Castor wheel, Swedish wheel, and the ball or spherical wheel [1,4].

In this paper, it was studied the wheeled mobile robot Summit XL, which uses four standard rubber wheels, developed by the Robotnik company. The purpose of this study is to demonstrate how this robot works, what are its features, and how it is configured and used from scratch. For this, the Linux and ROS basics were learned to work with the software of the robot, and the acquired learning was put into practice to test the real robot and the simulator. The scope of this project will first focus on the characteristics of the hardware and software of the robot, which will be discussed in the equipment and materials section. After, the procedure of the software that needs to be carried out for the configuration, operation, and simulation of the robot, will be described in the procedure section. Finally, the results obtained when testing the real robot and the simulator will be discussed, as well as the conclusion and the future work that was required.

Equipment and Materials

According to the introduction, the model of Robot that is used in this work is the Summit XL robot from the Robotnik Company (Figure 1). In the next sections, the hardware and software parts of the robot will be described.

Hardware

The following section details the main hardware components of the Summit XL Robot.

Figure 1: The Summit XL Robot (Robotnik) in the Robotic Lab, MCSE School, Liverpool Hope University.

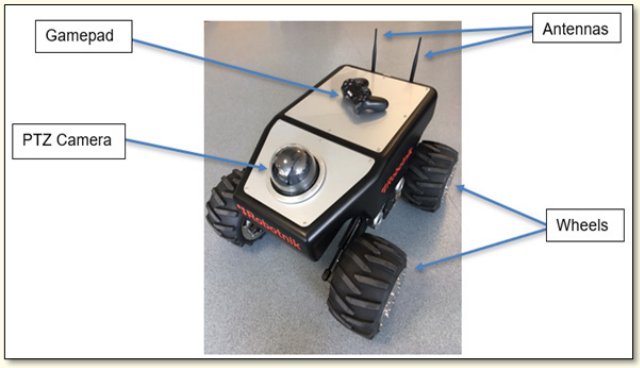

Figure 2: The main external elements of the Summit XL Robot.

The robot

The Summit XL robot is a wheeled land mobile robot, which uses skid-steering kinematics configuration to move in the environment. Although this machine is mainly used for research and development, it is also used in academic research, military, remote monitoring, surveillance, and hazardous areas. The robot, by default, is endowed with 4 rubber wheels, brushless motors, an Inertial Measurement Unit (IMU), a Wi-Fi router, a Linux-based embedded computer, and ROS software architecture [5,6]. Besides, it comes with a Bluetooth gamepad to control the robot. However,

if wanted, more optional equipment can be added or changed, depending on the type of task that is wanted to use it for. for example, a GPS can be included to locate the robot, or also, conventional rubber wheels can be replaced by mecanum wheels or better known as Swedish wheels, to have an omnidirectional configuration, and thus, have better maneuverability for indoor environments [7-9]. In this project, it is opted to use the robot with default settings and with the optional Pan-Tilt-Zoom (PTZ) camera. Therefore, the equipment of the robot used specifically in this project will be explained. In Figure 2, the external elements of the robot are indicated, and in Figure 3 the internal elements.

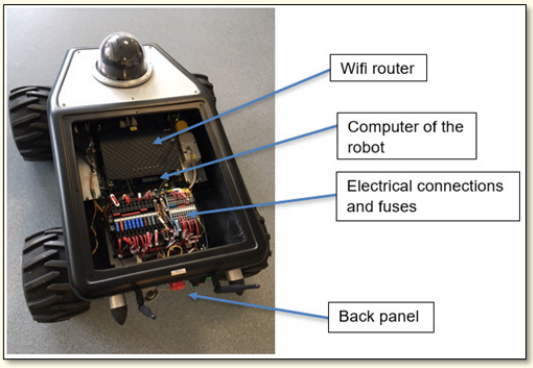

Figure 3: The main internal components of the Robot.

The robot uses brushless motors with encoders and a gearbox for each wheel, and each motor is controlled by an individual driver. These kinds of motors have a much longer life expectancy and a higher efficiency than brushed motors [6]. The drivers of the motors are connected between them and the computer by Controller Area Network (CAN) bus, which allows the computer to send CAN messages to move the robot. The device has also a skid-steering kinematics configuration to control its drive. This type of locomotion system is based on wheel slippage in order to turn right or left, consisting of controlling the velocities of the left and right sides motor drives [10].

The PTZ camera and Wi-Fi router

The PTZ camera of the robot is one of the essential elements for teleoperation, as it records and transmits video in real-time to see from a remote computer. This camera can move both in the X-axis and in the Y-axis, and it offers a 360° pan with Auto-flip, 180° tilt, and 12x zoom capability (AXIS P5514 PTZ Network Camera, [11]). Furthermore, it has a high video resolution of 720p. The device is powered from 12 VDC, and the data communication is transmitted by an ethernet cable, which is connected to the robot's internal computer.

The wireless router that is inside the robot is responsible for connecting the elements of the robot, such as the camera in our case, the integrated computer, and other components, with the remote computer. Thus, the remote computer receives all the information of the robot via Wi-Fi, and all the commands are also sent from the remote PC to the robot (Robotnik Automation).

The IMU and the GamePad interface

The robot used for this project also includes the Pixhawk FPU device, which is used as an Inertial Measurement Unit (IMU). This element provides better-estimated information of the robot's position, thanks to its integrated gyroscope and accelerometers [8].

The gamepad used to control the movements of the SUMMIT XL robot is the same as the PS4 DualShock remote controller. This device connects with the robot through Bluetooth, by using the Bluetooth receiver of the gamepad, which is located inside the robot and connected to the USB port of the robot's computer. This remote controller has two joysticks to control the direction, traction, and the elevation of the robot, and also, another button to control the speed of the robot, which is able to select between five different speeds: very slow, slow, medium, high, and very high.

The on board robot PC

The embedded computer is a Mitac PD10BI model, and it is based on the Linux operating system. The computer is the main "brain" of the robot, as it manages all the devices connected to it and controls every function of the robot [6]. Most of the devices are connected to the computer's USB port, however, the camera is connected to the computer through the wireless and the ethernet cable. The drivers are communicated by CAN bus between them and then connected to the Peak-CAN.

Software

This sections refers to the main feature of the Software.

The ubuntu operating system (Linux)

As previously mentioned, this robot is programmed using the software ROS, and this framework only works on Ubuntu operat-

ing systems. Ubuntu is an open-source operating system, which is distributed by Linux and based on Debian software [12,13]. There are many versions of Ubuntu nowadays, and they are continuously updating new versions every two years. However, for this project, Ubuntu 16.04 LTS (Xenial Xerus) version is used both in the robot computer and the remote computer, as the ROS version that is needed only runs on this Ubuntu version [13,14]. This operating system has a terminal, also called Shell, which is a common tool to execute commands for most of the actions in Ubuntu, like installing packages and programs, creating a new folder, launching a program, etc. In this project, it was needed to familiarize with the use of this terminal, since the installation of ROS in the remote computer is made from the terminal, and also because to program in ROS it is necessary to use this terminal [15].

Robot operating system (ROS)

The version of ROS required to work with the SUMMIT XL robot is ROS Kinetic Kame, and it is primarily targeted at the Ubuntu 16.04 operating system, which was the version that was installed for the remote computer for this project. ROS open and collaborative software framework characterizes its compatibility with many different types of robots and the community that create developers and users of this software [16]. In the project, some specific packages already created for the SUMMIT XL robot have been used, which were found in the GitHub repositories of the ROS community [15]. Finally, packages contain executable files or also known as launch files, which are created to do a specific function, like for example to move the robot. These launch files are code scripts, normally written with Python, Java, or C++. Usually, these launch files for any type of robot can be found on the GitHub website, which are created by other developers.

ROS packages

The packages for the SUMMIT XL robot are found in the Github repository, and they are clustered into three stacks or Metapackages [15].

- The first stack is the summit_xl_robot, which contains all the necessary packages to control the robot, such as bringup, which contains launch files to run the robot, controller that contains the ROS Control controller, and the "web" package for the web interface.
- The second stack is the summit_xl_sim, and as its name suggests, includes the packages to allow the simulation of the robot and its sensors in Gazebo. Inside this stack, there are two packages that are, sim_bringup which contains the launch files to run the robot simulation, and the gazebo package for the robot simulation using Gazebo.
- The third stack is the summit_xl_common and contains packages shared between the robot and simulation stacks. These packages include the description package, which contains robot models available in the SUMMIT XL robot, control package that contains the ROS Control controllers used by the SUMMIT robot, "localization" package for the localization of the robot, navigation which contains the configuration to use the robot with the ROS Navigation stack, the padpackage to read the gamepad controller, aubo_moveit_config for using the AUBO i5 arm with the mobile robot, and the ur5_moveit_config package to use the UR5 robotic arm also with the mobile robot.

Graphical interfaces (RViz and Gazebo)

RViz is a 3D visualization tool developed for ROS framework [17,18], and it allows the visualization in real-time of the information of robots published in the ROS topics. Thanks to this graphical interface, the user can visualize what the robot is seeing, thinking, or doing at any time, with the information obtained from the sensors, camera, etc. [16,17]. These can be for example, the position of the robot in an environment, the map that is generating, or the images of the camera.

Besides, the user can interact with the interface and select different options depending on what the user wants to see, by adding or deleting the topics that are interesting for him. In this project, the RViz tool is used to visualize the real-time image of the PTZ camera of the SUMMIT XL robot [19].

Gazebo is a powerful 3D simulation tool for robots, which offers high-quality graphics and a graphical interface [20,21]. This tool offers the ability to test algorithms and robot performances, design robots, train AI systems or simulate populations of robots, using complex indoor and outdoor virtual environments [20,21]. Although it is normally worked on a physical robot, it is always recommended to use this simulator to test the robot before starting to use the real robot, and thus, to see how the robot behaves doing a specific task and also avoid any damages that can cause when testing directly with the real robot. Furthermore, the simulation gives the opportunity to use any kind of function of the robot and

to explore the different tasks that are possible to do with the robot, since usually the real robot does not have all the elements available and is not possible to perform some tasks.

Experimental Set-up

In this section, the basics of ROS and Linux, and the procedure to be carried out to set up the ROS and to be able to use the functions of the real robot and the simulator are explained step by step. In addition, the different tasks that can be done with the robot in the simulator are described, as well as how to connect the gamepad with the robot and a brief mention about the internal web server of the robot.

Linux and ROS commands

Before starting to work with ROS on the robot, some basic concepts of the Linux operating system and ROS were necessary to know. All the commands are executed from the Linux Shell, which is the terminal that is included in any computer that runs a Linux system. In the following list, the most important Linux commands that are necessary to use to work with the Summit XL robot are described [22].

- catkin_make - This command is used to compile the ROS workspace, and hence, creates any packages and files that were installed in the workspace. For this, the command needs to be launched in the root directory, i.e. in the catkin_ws. For example, once all the packages are downloaded and installed, to execute this command is essential, so that the system creates the libraries in the Devel folder.
- Touch - Similar function to the previous command, but this command creates files instead of folder.
- Export - Provides the ability to update the current shell session about the change that is made to the exported variable.
- Ssh - The Secure Shell (ssh) is a network protocol that allows users to connect to a remote machine in a secure way. For example, in this project this command is used to send commands from a remote computer to the robot. The structure of this command is as follows: ssh<username>@<hostname or IP address>. An example of this would be, ssh summit@192.168.0.200.

The Linux Shell is used as an interface of ROS to execute the commands. The main commands of ROS that are needed are:

- Roscore - This command executes the ROS master node, which is the most important node and on which the rest of the nodes depend. If the master node falls, everything will not work.
- Roscd - Allows to change rapidly of directory.
- Rostopic echo - Allows to see what a node is publishing.
- Rostopic list - Shows on the screen the active topics that exist.
- Rosrun - Allows to launch an isolated node. One of the examples is to use the command rqt_graph, which draws a graph of the current system with all the topics and nodes that are being used.
- Roslaunch - Allows to launch the files from the Launch folder.
- Rosbuild - Creates a package, ready to be executed.
- Export - Defines what node is executed in another machine.

Workspace set-up and packages installation (ROS)

The installation of ROS Kinetic and the packages are made using the Linux terminal, also called the Linux Shell. Once the workspace is set up properly, it can be started to install all the necessary packages inside the previously set-up src directory. The package is installed directly from the terminal, by copying the URL of the package file or "git" file from the website and pasting it in the terminal. In this project, the packages for the Summit XL robot are downloaded from the GitHub website. The metapackages for the Summit XL robot can be found in the "GitHub" website, and these are the web addresses of each of them

- https://github.com/RobotnikAutomation/summit_xl_sim
- https://github.com/RobotnikAutomation/summit_xl_robot
- https://github.com/RobotnikAutomation/summit_xl_common

After the folder to install the packages is accessed, the installation of the package is made. For this, in the terminal, the command git clone is added followed by the copied URL of the package, which can be simply pasted from the clipboard (Figure 4). So, an example of the full command to install a package will be as follows: <git clone https://github.com/RobotnikAutomation/summit_xl_sim.git>

After the package dependencies are installed, it is necessary to compile the program in order that the system generates the libraries in the workspace. Finally, once all the packages and system dependencies are installed correctly, the launch files will be ready to be used.

```
user@user-OptiPlex-7010:~$ cd catkin_ws_manex/summit_xl/src/packages/
user@user-OptiPlex-7010:~/catkin_ws_manex/summit_xl/src/packages$ git clone https://github.com/RobotnikAutomation/summit_xl_sim.git
fatal: destination path 'summit_xl_sim' already exists and is not an empty directory.
user@user-OptiPlex-7010:~/catkin_ws_manex/summit_xl/src/packages$
```

Figure 4: Example of how to write the command to install a package on a Linux system.

PC-robot wireless connection

As previously mentioned in the robot´s features section, the robot has its own internal computer and a Wi-Fi router to which it is connected from another remote computer. In the end, what is intended to do is a connection between two computers. The connection between the robot and the remote computer can be wired, using an ethernet cable, or wireless, using the Wi-Fi connection. In this project, both options were used. To establish the connection between the robot and the remote computer using an ethernet cable, it is necessary to configure the IP address on the remote computer, by adding the robot's IP address, the subnet mask, and the gateway address, as shown in figure 5. The ethernet cable is connected directly from the remote computer to the back panel of the robot.

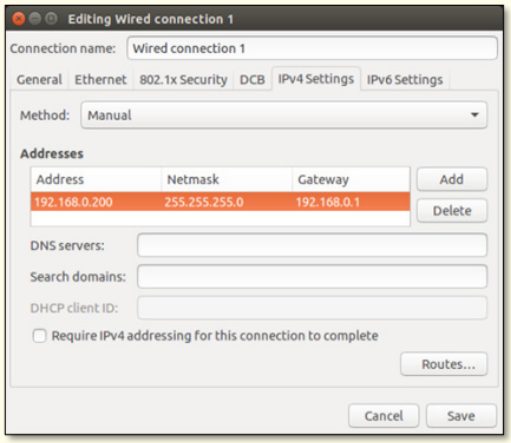

Figure 5: Setup of the IP address of the Summit XL robot.

For the wireless connection, it is simply connected to the robot's Wi-Fi SSID, which is indicated with the serial number of the robot SXL00-190911AA and entering the password that is specified in the manuals, which in this case is R0b0tn1K. After the connection is established with the robot, it is important to check if the remote computer and the computer of the robot are communicating with each other, which can be seen using the command ping 192.168.0.200, in the terminal. Finally, To be able to operate the robot it is essential to also connect the configured ROS program of the remote computer, with the robot's ROS Master node.

Gazebo and RViz simulation

To run the simulation of the Summit XL robot with Gazebo and RVIZ, before launching the simulator, there are some prerequisites that must be done. First, it is important to be inside the workspace directory while using the simulator, which can be accessed using the command that was previously mentioned. Once in the workspace, the ROS Master of the computer has to be set or run.

There are now three different options to move the robot using the terminal of the computer. One of the basic control functions is using the linear and angular XYZ parameters, which by changing the values of these parameters, the robot can move forward, backward, left, right and in circles (Figure 6).

```
user:~/catkin_ws$ rostopic pub /summit_xl_control/cmd_vel geometry_msgs/Twist "linear:
  x: 0.0
  y: 0.0
  z: 0.0
angular:
  x: 0.0
  y: 0.0
  z: 0.0" 
```

Figure 6: Twist commands example to move the Robot.

To use this function, simply the command is published into the topic /robot/cmd_vel. So, the command that would be executed, in this case, will be <rostopic pub /robot/cmd_velgeometry_msgs/Twist> and then hitting twice the TAB key.

Another type of function that offers the simulator to control the robot is the control through the keyboard of the computer. To use this function, it is necessary to create the Keyboard teleoperation launchfile (Figure 7). However, in this project to use this function with the simulator, it was opted to use a library that was downloaded from the Github website.

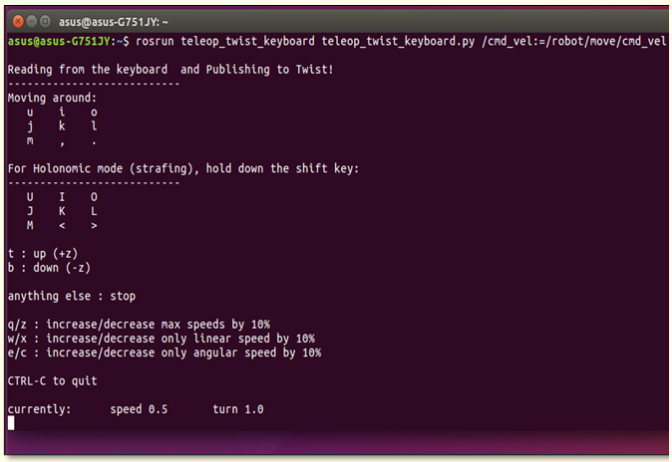

Figure 7: Keyboard teleoperation function.

The third option to move the robot is using the RViz Interactive Markers function, which allows moving the robot using the red arrows to control the linear velocity, and the blue circle for the angular turn.

Preliminary tests

In this project, all the procedure described in the previous section, to set up the ROS program and to use the robot, was carried out to put it in practice with the real robot and in the simulator. Although some functions could be run and used when practicing with the real robot and the simulator, it is important also to mention that there were some limitations when working on the physical robot and that some functions could not be possible to complete as expected, due to some issues with the configuration of the robot (Figure 8).

ROS configuration

First of all, whether ROS was installed on the remote computer was checked by typing the command ros version -d in the terminal, and ROS Kinetic was successfully installed. Also, using the command ros version roscpp it was checked the number of the version of ROS.

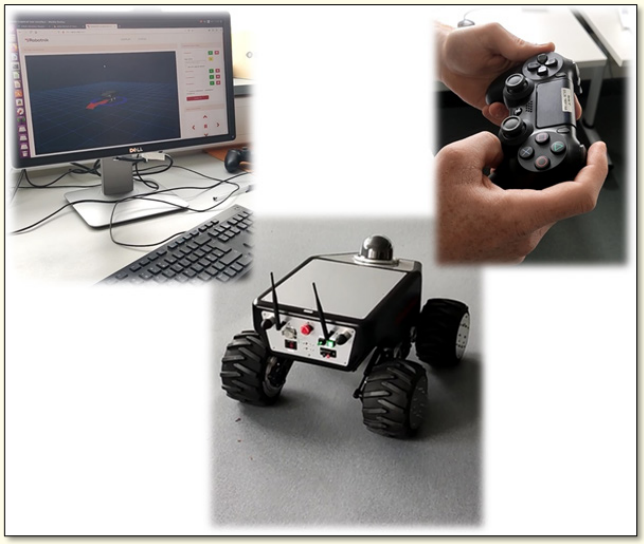

Figure 8: Testing of the Robot with the ROS interface (top left panel) and the GamePad (top right panel) in the Robotic Lab.

Connectivity

At first, when starting to work with the robot, there was a hardware issue. This issue was that sometimes the Wi-Fi router of the robot turned off unexpectedly due to a bad connection of one of the power cables of the router in the robot, which cut the connection between the robot and the remote computer. This issue was later fixed by tightening the connection of the cable. Both the connection through the ethernet cable and Wi-Fi were made to connect to the robot, and the connection could be established with both options. However, when checking the communication between the robot and the remote computer through the Wi-Fi connection, it was noticed that sometimes it gave some issues with the communication with the robot and on the contrary, using the ethernet cable they communicated without any problem. Using the ping command in the Linux Shell, it was checked the communication of the remote computer and the robot.

Using the robot and the simulator

Apart from testing with the physical robot, the simulated version of the robot was also tested using different functions in the simulator. To run the simulator, the already mentioned launch file

examples were used, and despite one example, which is the one to simulate with 3 Summit robots, the rest of the example worked fine. After running the simulator, Gazebo and RViz interfaces opened successfully and could be used without any problem. Once RViz and Gazebo opened, some basic functions were used to test with the robot. The first step was to move the robot using the teleoperation interface that is already set in RViz by default and the different elements of the robot were also tested, such as the laser sensor readings, camera, or maps. Furthermore, the 2D Nav Goal function was also used to move the robot to an indicated position, and it worked fine.

Conclusion

This project started out with the goal of learning about the Summit XL mobile robot and demonstrating how this robot is configured and operated from scratch. With the study carried out of the robot, the features and functionalities of the robot were learned, as well as the Linux and ROS basics to use the robot and its simulator. Then, as a second scope, the concepts learned were put into practice with the robot and the simulator in order to do the necessary setups and to operate the physical and virtual robot.

Although the testing of the simulation of the robot was a success, it was not possible to complete the second objective completely, despite having followed the entire procedure to configure the robot step by step, since there have been some unforeseen errors which have been found while working on the configuration of the physical robot. Therefore, the arrangements that need to be made to fix the current issues, finish completing properly the setup of the robot, and thus, to be able to move and perform different tasks both with the real robot and in the simulator, will require a future work.

Overall, this study has achieved the main purpose, which was to learn about the Summit XL mobile robot and to show how it is configured and used from scratch.

With this, it is intended to guide and ease the work to those who will continue working in the future on this robot, giving more detailed information that sometimes it does not specify in the manuals of the robot.

The system that was proposed in this work can be clearly implemented and adopted in all scenarios where surveillance and inspection are required, including industrial application, where having a 24/7 system performing a proper monitoring of the plant is of absolute interest [23]. Moreover, the availability of such a mobile platform, suggests that other applications can be implemented where, for example, a different set of sensors could be integrated on the platform in order to improve the monitoring capability of the system; to this aim different sensorized systems can be considered, such as, for example, thermal imaging camera to monitor electrical equipment [24], air pollution monitoring system [25], computer vision system to enable recognition capability [26]. Finally, the proposed human-robot interaction system (i.e. the GamePad controller) could be furtherly enriched with other intuitive human-robot interface in order to make the overall architecture more approachable for non-expert end-users [27,28].

Acknowledgements

This work was presented in Coursework form in fulfilment of the requirements for the BEng in Robotics for the student Manex Ormazabal Arregi from the Robotics Laboratory, School of Mathematics, Computer Science and Engineering, Liverpool Hope University.

Bibliography

1. Siegwart R and Nourbakhsh I. "Introduction to Autonomous Mobile Robots". Cambridge: MIT Press, The (2016): 13-15, 30-35.

2. Jaulin L. "Mobile Robotics". ISTE Press – Elsevier (2015): 1.

3. M Innes and EL Secco. Design of an interactive BB8-like Robot, 7th International Congress on Information and Communication Technology (ICICT 2022), Lecture Notes in Networks and Systems (2022): 137-144.

4. D Campbell and EL Secco. "Design of an Omni-Direction Robot with Spherical Wheels". *International Journal of Engineering (IJE)* 14.1 (2022).

5. Robotnik. "SUMMIT-XL - Robotnik Products". Robotnik ® (2021).

6. Robotnik Automation, Summit XL mobile robot-Manuals, Spain (2020).

7. Vectornav.com. "What is an inertial measurement unit?" (2021).

8. Arrow. "What is IMU? Inertial Measurement Unit Working and Applications" (2018).

9. Magiccvs.byu.edu. n.d. MDwiki (2021).

10. Wang T., *et al*. "Analysis and Experimental Kinematics of a Skid-Steering Wheeled Robot Based on a Laser Scanner Sensor". *Sensors* 15.5 (2015): 1-3.

11. Axis Communications. "AXIS P5514 PTZ Network Camera" (2021).

12. Helmke M. "Ubuntu Unleashed 2015 Edition: Covering 14". 10 And 15. 04. Sams Publishing (2014): 35-36, 39.

13. Tabassum M and Mathew K. "Software Evolution Analysis of Linux (Ubuntu) OS". *International Conference on Computational Science and Technology* 14 (2014): 1.

14. Tutorialspoint. n.d. Ubuntu - Command Line – Tutorialspoint (2021).

15. Mahtani A., *et al*. "ROS Programming". 3rd ed. [Place of publication not identified]: Packt Publishing (2018).

16. Garcia L. "Navegación sin mapa y mapeado en robótica móvil para entornos no estructurados". Licenciatura. Universidad de Sevilla (2017).

17. Bergmann P. "Sistema de supervisión y telemetría para un robot móvil con pila de combustible". Licenciatura. Universidad de Sevilla (2018).

18. Stereolabs.com. n.d. ROS - Data display with Rviz | Stereolabs (2021).

19. Packt. n.d. RViz- ROS Robotics By Example (2021).

20. Gazebosim.org. n.d. Gazebo (2021).

21. Dineshkumar E. Robotic Arm Simulation with ROS and Gazebo - from Skyfi Labs. Skyfilabs.com (2021).

22. The Construct. Robotics and ROS Online Courses - Mastering with ROS: Summit XL (2021).

23. Tharmalingam K and Secco EL. "A Surveillance Mobile Robot based on Low-Cost Embedded Computers". 3rd International Conference on Artificial Intelligence: Advances and Applications 25 (2022).

24. J Humphreys and EL Secco. "A Low-Cost Thermal Imaging Device for Monitoring Electronic Systems Remotely". Computing Conference (2023).

25. M Van Eker and EL Secco. "Development of a low-cost portable device for the monitoring of air pollution". *Acta Scientific Computer Sciences* 5.1 (2023).

26. K Myers and EL Secco. "A Low-Cost Embedded Computer Vision System for the Classification of Recyclable Objects". Congress on Intelligent Systems (CIS - 2020), Intelligent Learning for Computer Vision, Lecture Notes on Data Engineering and Communications Technologies 61 (2020).

27. Bilawal Latif., *et al*. "Hand Gesture and Human-Drone Interaction". Intelligent Systems Conference (IntelliSys) 3 (2022): 299-308.

28. J Hutton and EL Secco. "Development of an interface for real time control of a dexterous robotic hand, using MYO muscle sensor". *Acta Scientific Computer Sciences* 5.3 (2023): 25-36.