Digital twin and user interface for autonomous inspection robots

Adam Daniel Sendzul[1*], and *Callen* Fisher[1]

[1]Department of Electrical and Electronic Engineering, Faculty of Engineering, Stellenbosch University, Stellenbosch, South Africa

Abstract. Autonomous robots are capable of greatly assisting human-led processes and operations. These robots help by performing tasks, such as mapping, exploration, assembly assistance, and transporting goods, in complex environments in order to benefit the safety and effectiveness of the people involved [1]. While these robots have the ability to improve the conditions for people in many different applications, working and interacting with these robots is often complex and not easily adoptable. Therefore, accurate remote monitoring and user-friendly interaction of autonomous robots is achieved through the creation of a digital twin and user interface using Unity3D [2] and the robot operating system (ROS) [3].

1 Introduction

Autonomous robots are seeing increasing usage in environments where mapping and inspection can benefit and improve the work of human workers/operators. Environments such as warehouses [4], factories [5], and even subterranean caves [1] are making use of co-bots and inspection robots to aid productivity and safety. The tools used by robotics engineers to interact with and work with these robots are seldom user friendly, and the users working with the robots often do not possess the programming and engineering skills required. Thus, there is room for more user-friendly robot monitoring and operation solutions.

Robots in these scenarios are equipped with a large array of sensors, such as LiDAR's and cameras, commonly used for simultaneous localisation and mapping (SLAM) as well as for autonomous navigation. Therefore, a large amount of data is available to be visualised. This can easily get overwhelming and be the opposite of informational. As such, an insightful display to provide context and use the data effectively can be developed from all the available data. Due to the plethora of available sensor data, such a solution can come in the shape of a digital twin, where a digital robot model is connected to the real-life robot and the sensor data used to provide real-time monitoring and context for the real robot. But, one needs to distinguish between just visualising the data, and thus creating a digital shadow, and creating a real-time, bi-directional connection between the robot and the software solution [6].

Software such as Unity3D [2] has been shown to improve user adoption of robot control and make the experience easier for users not familiar with robotics engineering tools and

* Corresponding author: 23557702@sun.ac.za
This work is funded by Enaex Africa.

robotic control [7]. Bi-directional communication is achieved by implementing a feature such as in [7] where the robot can both send mapping data to the virtual reality system, and the user can teleoperate the robot using the same connection.

Technology such as virtual reality (VR) also helps to bridge the gap between the tools needed to use these robotic technologies and the users, facilitating the creation of a human-machine interface (HMI) [8] [9]. Through making the experience seem more natural and immersive, the learning curve and ease of use of developed technologies can be greatly improved. Through the transmission of SLAM data (as opposed to camera feeds), a telepresence can instead be created and an immersive VR experience created for robot monitoring and control. While focusing mostly on head-mounted VR, [10] explains some of the benefits of telepresence technology. Simplifying the adoption of these teleoperation and monitoring technologies stands to benefit users of mobile robotics during rescue operations. The telepresence technologies also improve accuracy of desired movements when controlling the robots in these contexts.

This paper details how a TCP/IP connection is made between an autonomous robot and a software solution developed using Unity3D, providing real-time data to a digital model of the robot and a bi-directional communication interface. SLAM algorithms were used and the outputs transmitted to the digital twin so that a 2D or 3D map can be visualised by the user. The user is also able to use a gamepad remote controller to teleoperate the robot (while not in autonomous mode), demonstrating bi-directional communication. Using software such as Unity3D [2], an intelligible user interface and standalone application can be created for the digital twin as well.

2 Methodology and results

The digital twin was created using Unity3D as the underlying engine for graphical display and physics simulation. The ROS TCP Unity package from Unity Robotics [11] was used to bridge the communication between the Unity3D application (digital twin), and the robot operating system (ROS) software running aboard the robot [3]. The ROS TCP Unity package allowed TCP/IP communication from the computer running the digital twin application. This communication occurs through the network running on the robot, to the robot computers themselves. Thus, the sensor data can be directly retrieved over Wi-Fi for real time operation of the twin, as seen in Fig. 1. The robot then publishes its sensor data retrieved from its onboard sensors onto ROS topics on the wired network from which the twin then subscribes to and reads wirelessly over the network and outputs to the user interface.

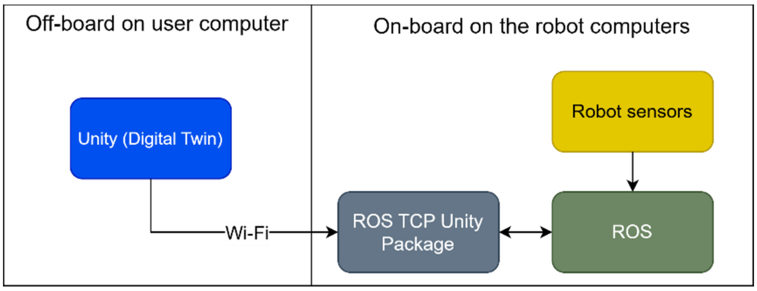


Fig. 1. The structure of the communication for the digital twin to the robot.

The ROS TCP Unity package runs its own ROS node to facilitate communication between the digital twin Unity3D side and the ROS robot side. If connection is lost to the server from the twin, the twin will indicate an error and resume work once the connection is restored. However, if the server node on the robot crashes, it will need to be manually restarted. This manual intervention can be achieved through SSH connection to the robot's terminal, still achievable on the network by an off-board computer, limiting exposure to the actual robot even for connection error resolution.

2.1 Mapping visualisation and teleoperation

The robot computer used for testing of this work has on-board SLAM algorithms that produce 2D and 3D maps of the surroundings of the robot. The 2D map is an occupancy grid represented numerically with the LiDAR as the sensory input. The numeric data describing this map is received on the appropriate ROS topic by the twin and a greyscale flat occupancy grid is displayed around the twin by using the message data to dynamically texture a Unity3D game object.

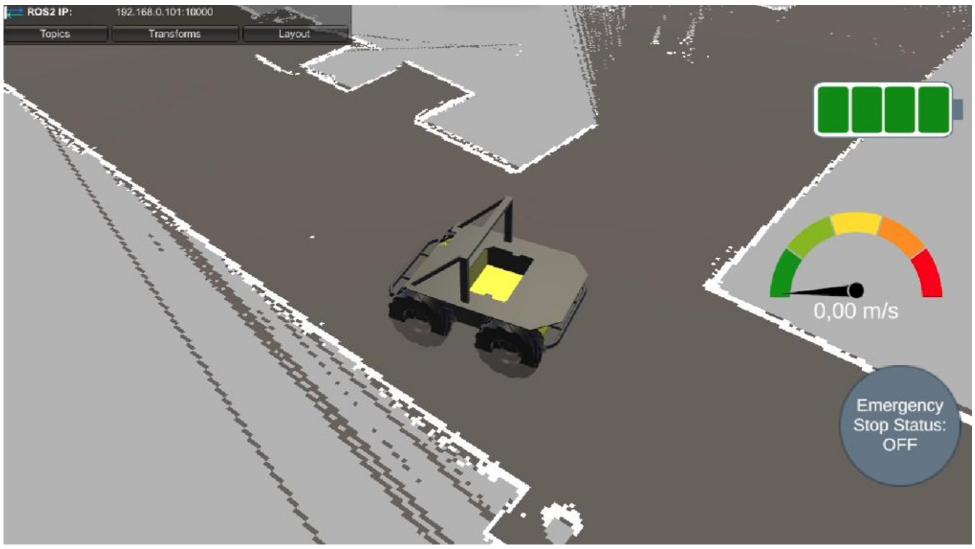


Fig. 2. Screen capture of the visualisation of a 2-Dimensional map through the digital twin.

Fig. 2 shows the results of plotting the occupancy grid results of the 2D SLAM algorithm running on the robot computers. The dark grey areas are navigable by the robot, and the light grey is unscanned territory. The white areas depict "walls" or boundaries detected by the LiDAR. The robot can then be navigated either autonomously or through teleoperation in this visualised map, moving the real robot in the same area. Fig. 3 shows a real-life comparison to the 2D map as well.

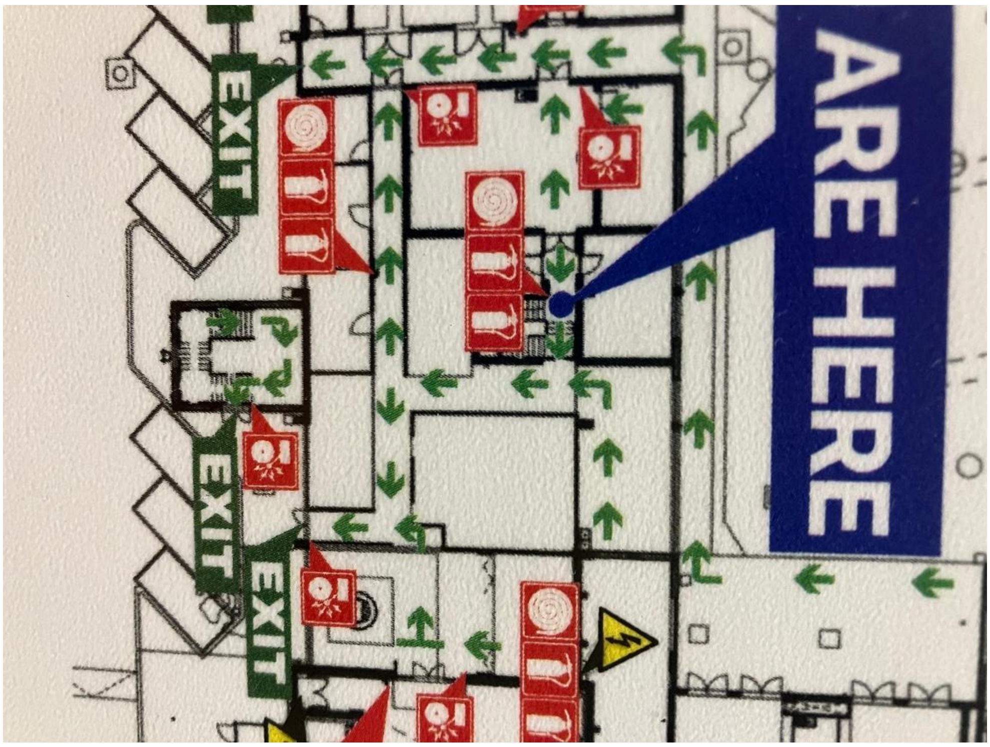
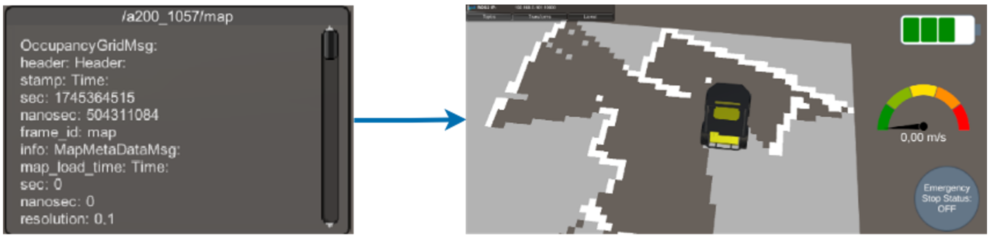


Fig. 3. Floor plan of the testing area, showing in the centre, the area that was mapped in 2D.


/a200_1057/map

```
OccupancyGridMsg:
header: Header:
stamp: Time:
sec: 1745364515
nanosec: 504311084
frame_id: map
info: MapMetaDataMsg:
map_load_time: Time:
sec: 0
nanosec: 0
resolution: 0.1
```

Fig. 4. Diagram showing a snippet of the map data text and how the map data is translated to a visualisation in Unity3D.

Fig. 4 shows how the map data for the occupancy grid is received in Unity3D and then translated to a visualisation by the dynamic texturing of a game object. The message data is a numeric array indicating occupancy of the area scanned, and thus using this numeric array, a colour code is formed.

The 3D map is a point cloud reconstruction of the surrounding environment represented numerically with RGB-D cameras as the sensory input. Similar to the 2D map, the data for the 3D map is received on the appropriate ROS topic by the twin and a correctly coloured cloud of points is displayed around the digital twin. This is done by using the numeric data for the coordinates of each point and the colour data and dynamically texturing a Unity3D game object.

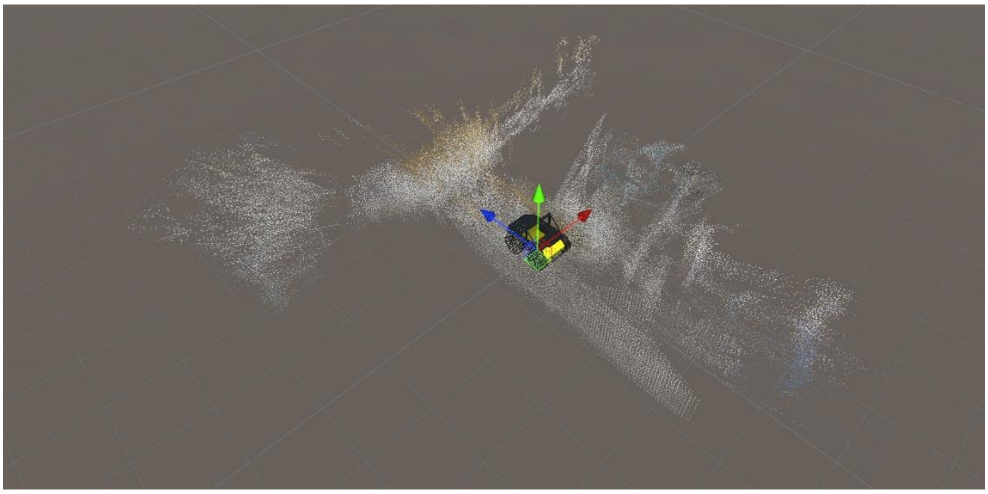


Fig. 5. Zoomed-out bird's eye view of the 3D point cloud mapping.

Fig. 5 show the results of visualising 3D point cloud mapping data from the SLAM algorithm running on the robot computers. The mapping was done in a hallway, hence the map resembling corridors (as can be compared with Fig. 6). The mapping data was more comprehensible when viewed using tools on the robot, indicating that the Unity3D visualisation of the 3D point cloud data was limited in quality. Visibly it seemed that the cloud of points was not being drawn densely enough, since as one zooms out the map becomes more comprehensible.

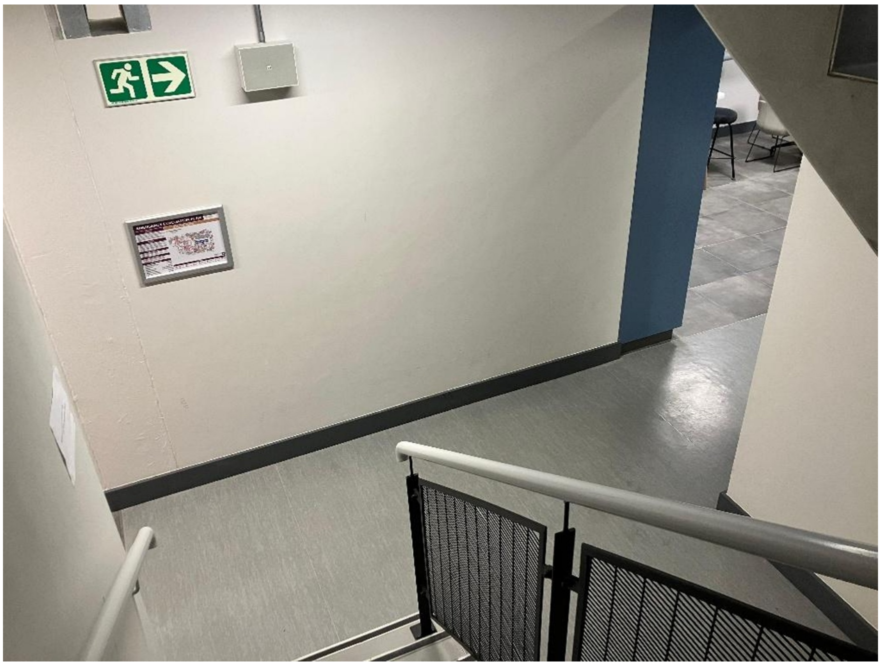


Fig. 6. A comparison of the real-life room that was mapped in Figure 4.

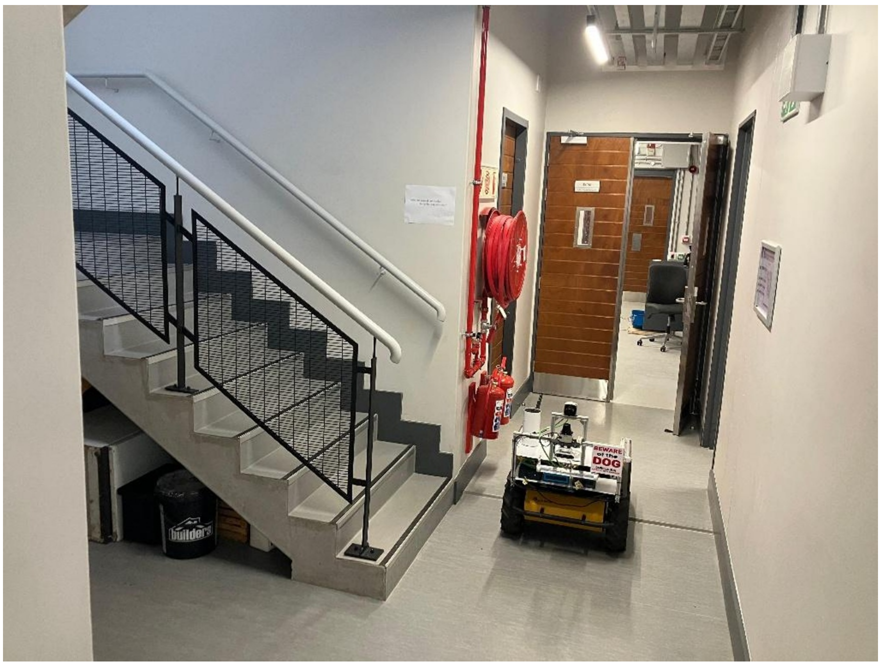
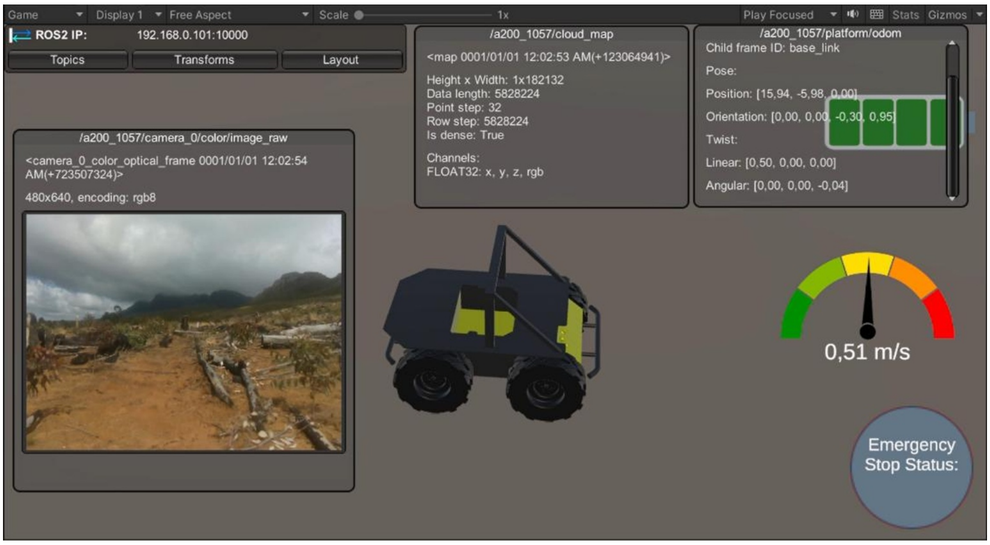


Fig. 7. Another comparison of the same hallway as before.

The two-way monitoring and influencing nature of digital twins is achieved in this case not only through reading data from robot sensors, but also through communication of teleoperation commands from the twin to the physical robot itself [6]. Therefore, using a gamepad controller and the robot's network, simultaneous teleoperation of both the digital twin and the real twin is achieved in real time. Using the established TCP/IP connection to the real robot, the twin instead of subscribing to ROS topics, now publishes to the teleoperation topic of the robot's API. The data published to this topic is velocity commands calculated from scaling the user-selected speed mode (and thus maximum speed) with the gamepad analogue stick actuation amount. The user can select the speed mode using button presses between a fast and slow mode for the robot, where the speed modes limit the maximum speed commanded to the robot.

The digital robot model mimics the movement of the real robot. This mimicry is achieved by reading a coordinate transform from the ROS transform created by the SLAM algorithms between the map and the robot's pose and updating the digital twin's position in the displayed map. The digital twin's position relative to the map is given the same coordinate transformation translated into Unity3D's coordinate system.



```
ROS2 IP:    192.168.0.101:10000

/a200_1057/camera_0/color/image_raw
<camera_0_color_optical_frame 0001/01/01 12:02:54
AM(+723507324)>
480x640, encoding: rgb8

/a200_1057/cloud_map
<map 0001/01/01 12:02:53 AM(+123064941)>
Height x Width: 1x182132
Data length: 5828224
Point step: 32
Row step: 5828224
Is dense: True
Channels:
FLOAT32: x, y, z, rgb

/a200_1057/platform/odom
Child frame ID: base_link
Pose:
Position: [15,94, -5,98, 0,00]
Orientation: [0,00, 0,00, -0,30, 0,95]
Twist:
Linear: [0,50, 0,00, 0,00]
Angular: [0,00, 0,00, -0,04]
```

Fig. 8. Screen capture of outdoor test results showing a debug interface.

Fig. 8 shows a version of the user interface of the digital twin wherein debug data is shown to the user. Data such as camera feeds, as well as map data from the ROS topic (text data before visualisation) is shown to demonstrate functionality and help debug robot and system operation.

2.2 User interface

The user interface allows the functionality of the digital twin to be easily accessible and understandable. The elements of the user interface focus on ease of use and simplicity, with displays aiming to be understandable and clear. The application consists of a dashboard startup screen, as well as a 3rd person view of the robot for operation, be it observation or teleoperation. During operation, the user can make use of a "pause" menu to adjust visualisation and manipulation settings such as what map is being displayed, and whether the robot is in autonomous or teleoperation mode.

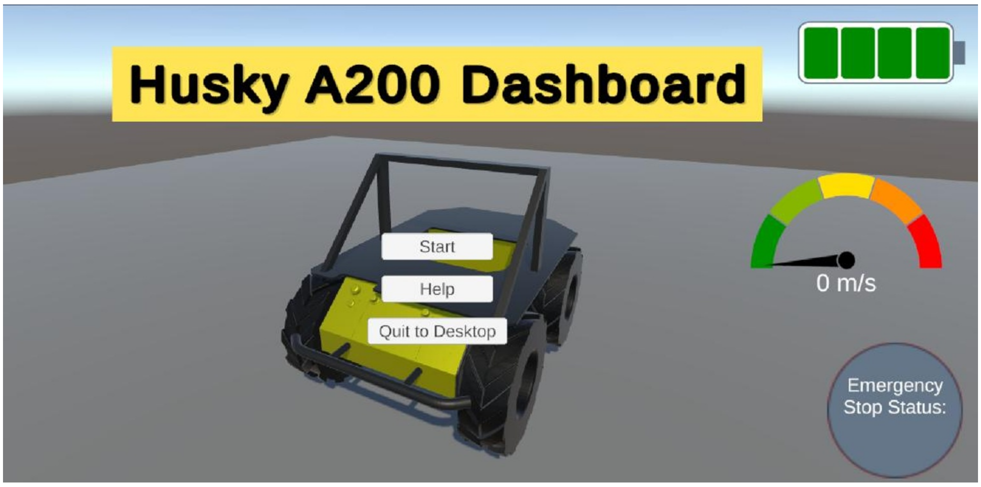


Fig. 9. Screen capture of the main menu screen of the digital twin application.

Fig 9 shows the opening screen, which is a dashboard view of the robot and some sensor data. The screen features a user-friendly user interface wherein the user can transition to a screen for more information and instructions on the use of the app ("Help"), a button to exit the whole application ("Quit to Desktop"), and finally a button to transition to the main scene wherein the maps can be visualised ("Start"). The user can use the gamepad to select an option. The screen also features a static model of the robot.

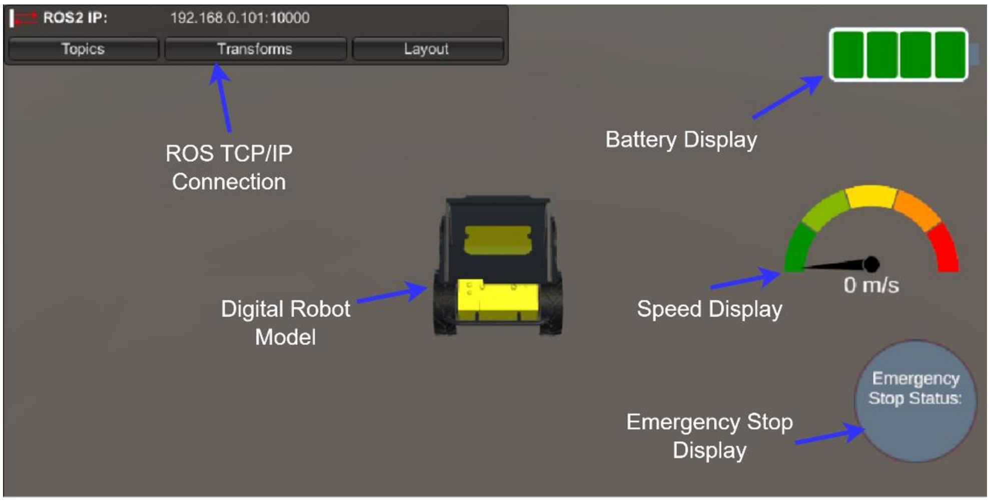


Fig. 10. Screen capture of the main operation screen, with labels indicating each UI element.

Fig. 10 shows the screen transitioned to once "Start" is selected. The screen features the same digital robot model, but now no longer static. The top left corner shows the status of the TCP/IP ROS connection between the digital twin and the robot's computers. The sensor data initially displayed before a mapping mode is selected, is the battery level, the speed of the robot, as well as the status of the emergency stop circuit. The speed display works in the same way as a speedometer, wherein the needle moves once a non-zero speed is received. The emergency stop indicator changes colour to a red when the circuit is engaged.

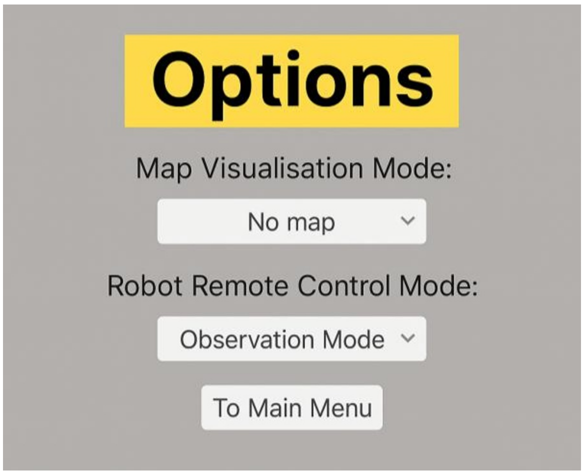


Fig. 11. Screen capture of the "pause" menu showing options to customise the visualisations and operation modes.

Fig. 11 demonstrates the customisation capabilities of the digital twin application. As long as both 2D and 3D SLAM algorithms are running on the robot computers, the mapping mode can be freely switched between the two visualisations. The menu also offers customisation options for robot control, wherein autonomous observation mode, keyboard teleoperation, or gamepad teleoperation can be selected. During autonomous observation mode, unlike the teleoperation modes, the digital twin is moved (located in the visualised map) by the autonomous navigation algorithms running on the robot computers.

3 Conclusion

Through the use of a digital twin and user interface created in Unity3D, remote monitoring and optional control of an autonomous inspection robot was achieved. Data about the robot itself as well as its surroundings was displayed, giving a more natural and effective view when compared to camera feeds alone [7]. Despite this, options to view the camera feeds are still provided for a complete monitoring experience including debugging functionality. The digital twin system provides a more user-friendly experience of robotics interaction aimed at assisting the effectiveness of the adoption of autonomous robotics, in the interests of promoting safety.

A limitation identified was the quality of the 3D map and thus the impact on user experience and smoothness. The data being transmitted is displayed comprehensibly on the robot computers; therefore, future work involves improving the visualisation algorithm on the Unity3D side such that a more comprehensible map is visualised on the digital twin side. This will also improve navigation via teleoperation, since the user can better comprehend their surroundings. Further future work includes extending the functionality of the digital twin interface to integrate the sending of terminal commands to the on-board robot computers, potentially through facilitating a secure shell (SSH) connection. Future work also includes thorough latency testing to investigate to what degree the system can provide real-time feedback, and how it can be improved.

References

[1] T. Neumann, A. Ferrein, S. Kallweit, and I. Scholl, *Towards a Mobile Mapping Robot for Underground Mines*. 2014.
[2] ‘Unity Real-Time Development Platform | 3D, 2D, VR & AR Engine’, Unity. Accessed: Feb. 24, 2025. [Online]. Available: https://unity.com
[3] ‘ROS 2 Documentation — ROS 2 Documentation: Humble documentation’. Accessed: Feb. 24, 2025. [Online]. Available: https://docs.ros.org/en/humble/
[4] P. R. Wurman, R. D’Andrea, and M. Mountz, ‘Coordinating Hundreds of Cooperative, Autonomous Vehicles in Warehouses’, *AI Mag.*, vol. 29, no. 1, Art. no. 1, Mar. 2008, doi: 10.1609/aimag.v29i1.2082.
[5] S. Harapanahalli, N. O. Mahony, G. V. Hernandez, S. Campbell, D. Riordan, and J. Walsh, ‘Autonomous Navigation of mobile robots in factory environment’, *Procedia Manuf.*, vol. 38, pp. 1524–1531, Jan. 2019, doi: 10.1016/j.promfg.2020.01.134.
[6] A. Mazumder *et al.*, ‘Towards next generation digital twin in robotics: Trends, scopes, challenges, and future’, *Heliyon*, vol. 9, no. 2, Feb. 2023, doi: 10.1016/j.heliyon.2023.e13359.
[7] C.-Y. Kuo, C.-C. Huang, C.-H. Tsai, Y.-S. Shi, and S. Smith, ‘Development of an immersive SLAM-based VR system for teleoperation of a mobile manipulator in an unknown environment’, *Comput. Ind.*, vol. 132, p. 103502, Nov. 2021, doi: 10.1016/j.compind.2021.103502.

[8] J. E. Solanes, A. Muñoz, L. Gracia, and J. Tornero, ‘Virtual Reality-Based Interface for Advanced Assisted Mobile Robot Teleoperation’, *Appl. Sci.*, vol. 12, no. 12, Art. no. 12, Jan. 2022, doi: 10.3390/app12126071.
[9] C.-S. Chen and C.-W. Lui, ‘Applying virtual reality to remote control of mobile robot’, *MATEC Web Conf.*, vol. 123, p. 00010, 2017, doi: 10.1051/matecconf/201712300010.
[10] K. Kanazawa, N. Sato, and Y. Morita, ‘A Verification of a Teleoperation Interface for Rescue Robots using a Virtual Reality Controller with a Door-Opening Task’, in *2023 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, Nov. 2023, pp. 113–118. doi: 10.1109/SSRR59696.2023.10499932.
[11] ‘Unity-Robotics-Hub/tutorials/ros_unity_integration/README.md at main · Unity-Technologies/Unity-Robotics-Hub’. Accessed: Feb. 24, 2025. [Online]. Available: https://github.com/Unity-Technologies/Unity-Robotics-Hub/blob/main/tutorials/ros_unity_integration/README.md