Virtual Reality Teleoperation Robot

Alexis Koopmann, Kressa Fox, Phillip Raich, Jenna Webster

***Abstract*—The ability to access and interact virtually in remote, and sometimes dangerous, environments have become highly applicable in many industries. Our senior project team developed a virtual reality device to increase the accessibility of these environments. The system our team created utilizes a wireless controller to operate a robotic car and send visual feedback to a virtual reality headset through a camera stream. This system provided a safe and entertaining way for an operator to access environments remotely.**

***Index Terms*—Computer Engineering, Robotics, Teleoperation, Virtual Reality, etc.**

I. Introduction

THERE are many places where people need to reach but cannot access due to physical limitations or safety concerns. These environments range from the depths of the sea to the vast expanse of space. There are countless areas that cannot support human life or are physically inaccessible to humans that have yet to be explored. Having a way to access these spaces, with enough information to successfully navigate them without being physically present is a valuable resource.

Many modern navigation systems haven't caught up with the rapid development of new technology. Some still rely on low-level machine code commands, difficult user interfaces, and poorly designed displays as a means of control. One example of where updated control systems would be useful is in nuclear reactors. "Today's nuclear I&C maintenance departments are faced with outdated electronics, increased failure rates, and tight plant operating budget" [1]. As Virtual Reality becomes increasingly more cost effective, integrating Virtual Reality into these systems could reduce human error, increase operator engagement, and update back-end code frameworks to a more modern language (C#). Sites like Chernobyl already use 3D mapping with teleoperated robots in order to navigate inhospitable environments. When accessing potentially hazardous environments, human error or lack of engagement can result in dangerous consequences. The system we are creating will provide a high quality visual and control interface that benefits users. Instead of interfacing with an environment through low-level machine code, our systems' users can see the space visually, and interact with it via a game controller. Virtual reality (VR) interfaces provide a massive amount of information to users, minimize human error, increase efficiency, and reduce stress when operating machinery.

Our project will consist of a small vehicle that will be linked to a Virtual Reality Display headset. The robot will operate like a remote-control car. For our project, we mixed the word car and the acronym VR to create the projects' name "CarVr". This robot will be able to move, provide camera input, and potentially interact with an inaccessible environment.

The main objective of the CarVr system is to demonstrate the operation of a mobile vehicle that is located in a physically different area from that of the user while using augmented reality to maintain a sense of presence with the vehicle for that user. The augmented reality (AR) is a result of combining the VR display with the real world camera input. When the user puts on the headset and starts the application then they will see a camera feed from the vehicle. The user can then use the remote control to instruct the robot where to go, moving based on the provided camera input. The user will be able to use the device easily and enjoy the experience. If time permits, then the VR space will have a display resembling a car cockpit surrounding the camera feed so that the application feels more realistic. Extra functionality may include a NERF dart turret to allow the user to interact with objects in the environment. The NERF turret however is not vital to the success of the project and will be implemented secondarily.

The complete CarVr system can be split into two significant connections: camera to VR, and controller to the car. These connections make up the core functionality of the project and are where most of the effort will be centered. They differ from similar VR integration projects by the use of a mobile car and camera system. Since the purpose of the CarVr system is to allow the user to observe the world as if they were in the car, a camera will be mounted upon the car that feeds video into the VR headset. Utilizing the camera feed, the user of the CarVr can control the direction of the car with an external controller to explore the environment around them.

When the project is completed and demoed successfully, all materials, code, designs, and implementation will be posted open-source on GitHub. By creating and sharing our project for anyone to replicate, we are hoping to inspire more creators to start making new and different types of VR systems, integrated with unique hardware. If one can hook a VR environment to a mobile car, the knowledge can be extrapolated to work with other hardware. Adding to the knowledge pool for this growing industry will benefit more than just the research community. Our team aimed to demonstrate a proof of concept, highlighting that these types of systems can be created and that they provide unique functionality. Specifically, our team wanted to solve this accessibility problem in a safe and engaging way by combining VR and robotics. On a larger scale, we are setting up a framework for other groups to develop similar cross-discipline devices for society. To accomplish these goals, our team is designing a system that reduces human error, is visually engaging, and provides safe access to dangerous environments with VR.

II. Background

Groups that have started combining VR with teleoperation systems have seen positive results including increasing operators' spatial awareness to decreasing operator stress. A project

done by graduate engineers at Gifu University combined VR and a construction robot. They then monitored the operators' task efficiency and engagement with various displays. It was confirmed that the VR display far surpassed the conventional display in utility. Using VR to operate the robot was easier to operate, safer, and reduced stress on the operator [2].

MIT's Computer Science and Intelligence Lab (CSAIL) recently linked VR and a manufacturing robot. MIT found that the unusual design-led to participants having a higher success rate with the VR system than with more high-end expensive systems currently on the market. VR hardware control systems could provide a cost-effective alternative for companies. Alongside this, video game players were asked to test the MIT device. They found the gamers were quickly able to learn the basics of operating the manufacturing robot. VR displays could lower the barrier of entry into manufacturing jobs for individuals who play games [3]. While skeptical of the original design, CSAIL ended up declaring the project a success. Our team will be replicating a similar system by integrating a VR display into our "task system". Our tasks will differ from MIT's research, involving moving a mobile robot instead of stacking Legos.

Another example of combining VR and hardware systems comes from Brown University. They developed a software project that allowed users to control robots with everyday VR hardware. The system links a research Baxter robot with an HTC Vive headset. Tasked with stacking 16 different sized cups, operators found that the VR interface was more fun and easier than the keyboard interface. Not only did they stack the cups faster using the VR interface, but most participants were smiling and enjoying their experience [4]. Making sure users are enjoying themselves is important to efficient and easy task completion. Our design will again, incorporate similar display ideas to harness these positive effects.

To utilize VR, we needed a development program to make a VR environment. A baseline requirement for CarVr is for the video feed from the vehicle to be live-streamed to the VR headset. Other projects, such as the mobile robot produced by students at Dong-A University, have found novel solutions to providing live camera images to a VR headset [5]. However, these solutions did not allow for a virtual environment to be built up around the camera feed, or any extensible integration like Augmented Reality. Our project aims to extensible and adaptable, as a proof of concept project should be something that can be built upon. Brown's free software will not work with our project, as it can only operate with Baxter research robots. Two other major software programs exist for VR development: Unreal Engine and Unity. Unreal Engine is a VR software development tool better suited for game development than hardware interfaces. Unity has more options for real-world hardware interfacing, specifically integrating the camera views needed for the CarVr system. Based on this research, we decided to use Unity for CarVr.

A unique aspect of CarVr comes from the specific hardware design of integrating a mobile vehicle and camera system. The mobile vehicle will resemble a mars rover, with a camera fixture, tank tracks, and a box-type design. This is created to mimic the types of devices often found in dangerous and inhospitable environments, like the rovers used by NASA to explore space. Our rover type of design will provide more stability and functionality to the vehicle. Five key pieces of hardware equipment of the CarVr are the controller, camera, VR headset, the motorized vehicle to be controlled, and a single-board computer to control the vehicle. The code for communicating instructions from controller to car will be directly implemented on a computer system added to the car.

The car control system will be developed using a Raspberry Pi 4 and Python scripts. This setup was selected due to its large documentation support and its open availability. Many versions of Raspberry Pi operating systems are available; however, this project will move forward using the latest officially supported version Rasbian [6]. Raspberry Pi has many integration capabilities, and offers interfaces with many embedded systems, such as cameras, Bluetooth modules, haptic feedback devices, and more. We hope by building this new system, it will provide the operator with the benefits of the VR display, with the added functionality of a mobility robot. This will provide insight into the benefits VR integration systems provide in a more accessibility focused application.

III. Results

A. Implementation Overview

This implementation of the project was split into the following sections: the control system, the virtual reality system, and the locomotion system. These sections were implemented and debugged independently from each other, then integration testing and implementation were performed on the system as a whole. A simple block diagram can be seen in Fig. 1. How each of these sections was implemented will be explained in more detail in the following sections.

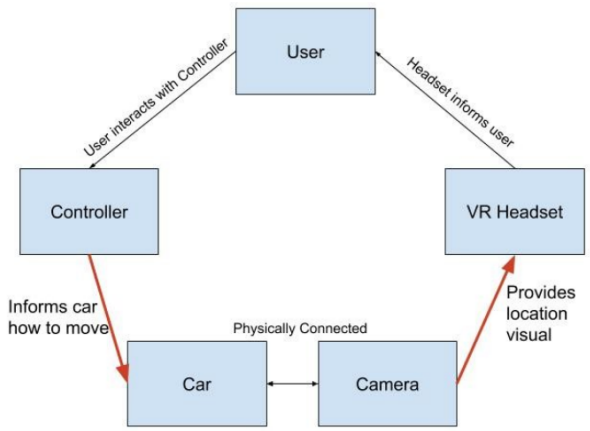

Fig. 1. A general overview of the CarVr system and connections.

A Raspberry Pi 4 is the primary controller of the system. Located inside the car, it handles a majority of the connection logic. The Raspberry Pi 4 compiles the camera feed, and then streams the feed to the internet for the virtual reality system to interpret. It also handles interpreting Bluetooth signals from the controller that direct the cars' motors. As a result of the Pi being a primary connection in all the system components, the board is where most of the debugging took place. A more

detailed diagram of the CarVr system is shown in Fig. 2. Critical data flows are shown by arrow colors: a green arrow flow indicates the data that provides video feedback to the user, while the orange arrows represent the data flow from the user to vehicle motors.

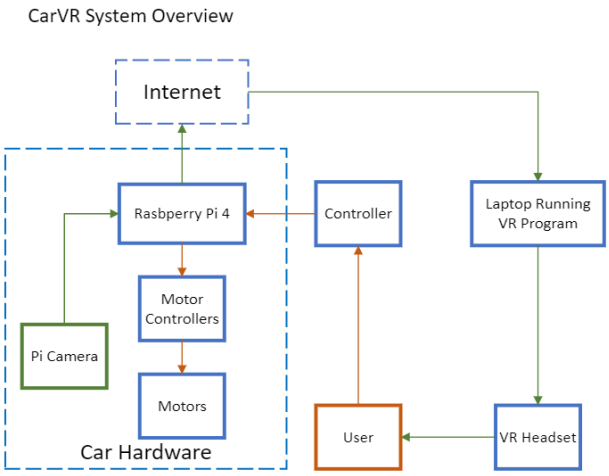

Fig. 2. A Specific Overview of the CarVr System and Connections

B. Required Resources

A complete list of all components has been noted in the references [7]. The resources used included an Oculus Quest 2 headset, a Raspberry Pi 4, a PS4 controller, a Raspberry Pi camera, VESC motor controllers, brushless motors, and various smaller electronic and 3D printed components. Our team had access to a 3D printer, soldering station, and a welder. These tools were crucial to the development, but substitutions can be made. The Oculus Quest 2 headset was chosen based on availability during the COVID-19 pandemic.

C. Camera to VR System

The camera to VR system is a crucial component of the project as it informs the user how to control the car. An image of the VR device we used for this system can be seen in Fig. 3.

Fig. 3. Oculus Quest 2 Headset [8]

This component required a Raspberry Pi compatible camera and an open-source Python camera script from documentation at *"rasbperrypi.org"*. The Raspberry Pi is run on Raspberrian OS and configured with an interface to facilitate the camera. This script sets up a server with the Raspberry PI IP address to send bytes from the Pi to a URL. From this setup, the Raspberry Pi produces a live, low lag, MJPEG stream that can be viewed in any browser with the corresponding IP address. This IP address is then inserted into the VR system using a C# Unity script. Linking real-time web streams remotely is not well documented. To address this, our team had to take an existing script from 2015, update it to be functional in Unity 2018, and trim unnecessary functionality. If reproducing this project, it is highly recommended to use the script provided in [9] and build upon it as needed. This MJPEG processing script handles sending and receiving HTTP requests and responses, as well as processes bytes coming in from the HTTP buffer. The MJPEG processing script interacts with a texture script, that works with Unity to display the bytes in a usable way in the VR environment. For the project to run, we setup the Oculus plugins on Unity. The Unity scene is comprised of a viewpoint, an empty room around a viewpoint, and limited virtual mobility inside the space. A screenshot of the scene can be seen in Fig. 4. In the scene, there is what appears to be a large screen to the viewer. There is also a camera icon which represents the users' field of view. Here the user is quite small in comparison to the screen size.

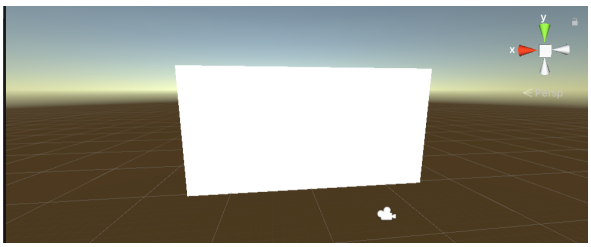

Fig. 4. Unity Scene Snapshot

The large screen size reduces the potential for nausea in the project. This setup also increases the novelty of driving a small car, as the user feels that they are also small. Both components were tested together until a live video stream could be seen in VR by running the camera python script at the same time as the Unity scene. A screenshot of the Unity scene running a camera feed can be seen in Fig. 5. Any camera stream can be used to test that the Unity environment is functional. The script used in testing pictured below is from Turkey [10].

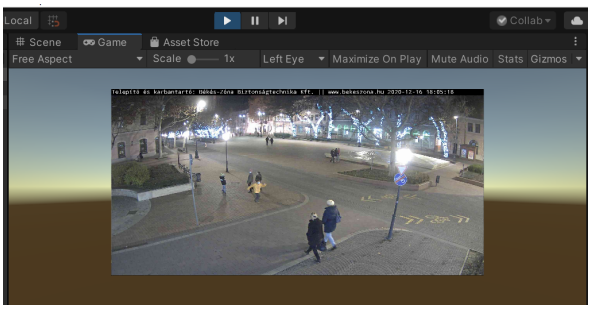

Fig. 5. Test Screenshot

Then the system was tested by running that Unity program on the Oculus Quest 2 headset. To set this test up, an Oculus Link cable is connected from the laptop to the Oculus Quest 2. Then the Unity project is run, and the resulting experience is displayed in VR. If the system is functional, a camera stream is visible in VR. These integration tests are highly recommended before continuing in development. Once complete, the baseline functionality of the VR environment is complete. The project can successfully provide a space for the user to view the camera feed, and be immersed. Other components can now be integrated into the camera and VR system.

D. Control Software

The control system is responsible for the speed and direction of the vehicle. This system is composed of the PS4 controller, Raspberry Pi, ESCs, motors, and a few signal converters. The control software of the car is a Python script, which runs on the Raspberry Pi. To control the speed of the system, the script creates a Bluetooth linking shell that interprets the PS4 controller input and determines the output signal for the motor controller.

For input, we use a PS4 controller. Rather than creating custom controllers, we pivoted to integrating existing wireless controllers instead. This choice not only made the project more accessible to the development community but saved a significant amount of development time as well. At first, we planned to connect the PS4 controller via a WiFi connection. This deliverable was pivoted to Bluetooth rather than WiFi as a result of system restrictions and lack of documentation. For the Bluetooth implementation, the script uses a Linux kernel Joystick API [11], which re-structures Bluetooth input devices as Joystick devices. Each time input on the controller is triggered (pressing a button, moving a joystick, etc.), an event structure is created and saved in an input stream. The project uses this API with pyPS4Controller [12], a lightweight module that reads the input stream and segments out events as they occur. This module also handles disconnection errors that report when a controller is offline. To make the connection process smoother, our team developed a single script file to link controllers and the Raspberry Pi, which facilitate a more streamlined startup process.

The output of the Python script associates the input variables from the PS4 controller with output signals that are used to control the speed of the motors. The output signal is a PWM (pulse width modulation) signal. The PWM signal outputs up to 35 different duty cycles, which defines 35 different speeds. The higher the joystick is pushed the higher the output duty cycle, which results in a higher speed.

To confirm the script controller script operated correctly, we analyzed the output logic of the Raspberry Pi 4 with a Saleae Logic analyzer. To perform this test we connected the logic analyzer to the output pin of the Raspberry Pi to see the PWM output. We shifted the joystick of the PS4 controller upward to make sure the output on the Saleae Logic analyzer shows that the duty-cycle becomes wider. This test is highly recommended when reproducing the project, as it lets a developer know if there is an issue with the software or the hardware components of the control system.

E. Hardware

The vehicle uses a three wheel design with two independent fixed motors for two wheel drive in the front, while a third caster wheel for stability in the rear. Using independent electronic speed controllers (ESCs) for each motor allowed for each wheel to operated at different speed, making steering possible without the need to pivot the wheels. The Bluetooth controller primarily uses the Y-Axis of the joysticks for throttle input. The left joystick position determines the left wheel's speed and the right joystick determines the right wheel's speed. This is a design commonly found in riding lawn mowers and works well once the operator has experience controlling such a system. An image of the finished vehicle can be observed in Fig. 6.

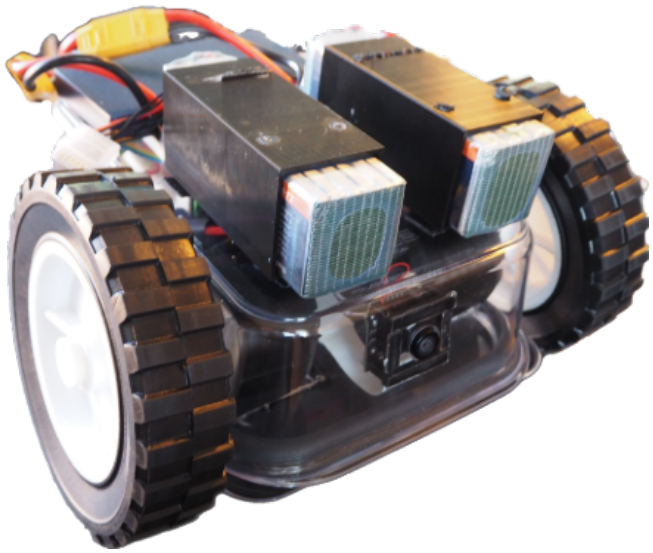

Fig. 6. Vehicle Exterior

The chassis is primarily made from plastic components. This decision was made after initial testing using a steel chassis proved to cause issues with the motors demagnetising internally. While an aluminum chassis could be used, the current plastic version fits budget constraints. 3D printed mounts are used to mount all components to the chassis and can be found for download from [9].

The power system is primarily 22.2 volts and supplied from two 6S LiPo batteries in parallel with a total capacity of 6600 mAh's. The batteries are 50C rated, meaning that they can supply enough current to power both 50 amp motors used. The motors are rated for 190 kV's and are usually used for electric long boards. Using these motors allows us to be unconcerned about vehicle weight as long as the chassis does not exceed 150 lbs. The downside to the use of the motors is their need for speed regulation. To obtain realistic control of the vehicle, the ECSs are programmed using VESC software to limit the motors rotational speed to 20000 rpm and have a max draw of 35 amps each. A throttle curve is also implemented so that users have fine control at slower speeds.

Connecting the Raspberry Pi to the ESCs requires additional components in order to use a PWM waveform as the throttle

signal generated by the control software. The selected ESCs are open source VESC 4.12 unites that allow for many types of input and user defined parameters. On each ESC, an analog signal is used in this case for the speed signal through the use of a built in ADC. To obtain an analog signal from a PWM signal, the use of a logical level shifter is needed to bring the signal from the Pi from 3.3 volts to 5 volts and to separate the Pi from sharing ground with 22.2 volt system. The 5 volt PWM signal is connected to a PWM to Analog converter (PAC) which translates the duty cycle given by the Pi to an equivalent voltage. These converters translate 100 percent duty to 6 volts. The ADC in the ESCs except a max of 3.3 volts so a voltage divider is needed between the PACs and the ESCs to bring the max voltage down to 3 volts. A system wiring diagram can be viewed in Fig. 7.

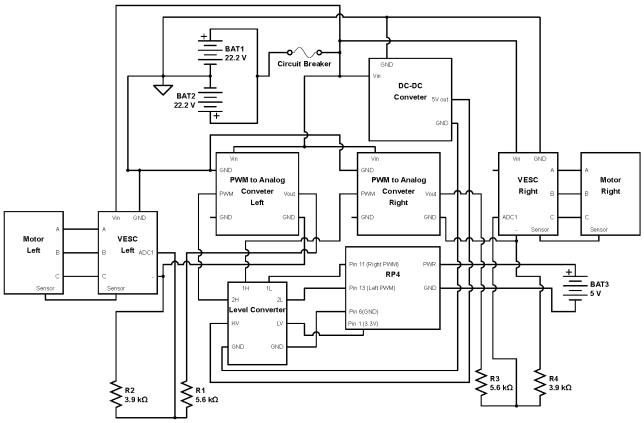

Fig. 7. System Wiring Schematic

Other components of the vehicle include a DC to DC converter, a fan, a 5 volt battery, a circuit breaker, and a Raspberry Pi specific camera. The DC to DC converter is used to power components such as the fan and the level-shifter at 5 volts while using the 22.2 volt batteries as a source. The cooling fan is needed in order to keep the Raspberry Pi from shutting down from thermal issues. A 5 volt battery is used to power the Raspberry Pi separately from the 22.2 volt system allowing the Pi to be used without power being active to the ESCs. An 80 amp circuit breaker is used to as the 22.2 volt systems main power switch and to protect the power systems wiring harness from melting. The camera is connected to the Pi and used to provide the image for the VR feed. An image of the inside of the vehicle where most components are housed can be viewed in Fig. 8.

IV. Conclusion

The team was successfully able to assemble and use the system as well as document our process. We were able to demonstrate that not only can VR be used as a means of controlling remote vehicle operation, but that these cross-discipline systems are engaging to use as well. While some small design details changed, for example, we did not use tank tracks in the vehicle design, the majority of the implementation stayed consistent. Through a lot of time, effort, and communication our team also demonstrated that these

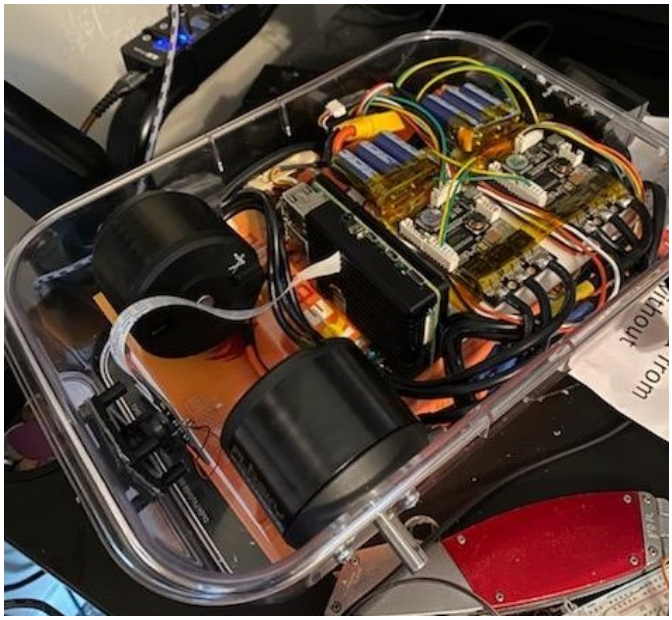

Fig. 8. A Close Up of Vehicle Hardware

systems can be created remotely, with the only in-person interaction necessary being the combination of separate system components. The implication of this showed that while being blocked on a particular technical aspect, a development team could outsource certain pieces of the project, and still create a functional system. It is also worth noting, in the new remote requirements of COVID-19, developing components separately was a safer as well. Our team demonstration acts as a foundation for other developers, in areas where having more engaging and interactive displays can benefit the completion of certain remote tasks.

A. Lessons Learned

Virtual reality isn't currently well supported for live camera service, so this process was very time intensive to debug. Motor and VESC integration is difficult, and had a lot of bugs that were related to hardware and software problems. Creating custom spaces in VR requires additional software, Blender, which we had not accounted for in our original proposal. Another incident our team learned was that automating the system with a Bluetooth controller wasn't possible during the time period. Due to the way the Raspberry Pi boots, the Bluetooth modules setup last during this process, and therefore any script that relies on the Bluetooth modules will not execute during boot. This prevented us from fully automating the system. This system is currently only safe if the operator is not moving around in space. Lagging WIFI speeds was not something we predicted, and as a result, occasionally the project performs poorly with slow WIFI speeds. This can however be remedied with a hot spot. It is beneficial to anyone reproducing this project, to take these into account when designing a new system.

B. Next Steps

As proof of concept, our project is a good foundation for the use of VR in teleoperation. Moving forward, our team discovered that enabling the controller to work over a WIFI connection, rather than Bluetooth, has promising implications. If the vehicle could be controlled via WIFI our project can be controlled anywhere in the world. This is a substantial extension of the project's functionality and could make the system more applicable in a variety of environments. For example, if large spaces need to be surveyed, that extend beyond the immediate area or reach of Bluetooth, the range would no longer be a limiting factor. Beyond this, additional features, and Augmented Reality functionality would further enhance the capabilities of the system in specialized use cases. One example with the integration of Augmented Reality, is in surveying various types of geological features. Augmented Reality could inform a driver about what rocks, flora, and fauna, might be detected on the camera feed. This can inform user actions, and potentially, actions the vehicle can take depending on the feedback. For example, if a special type of rock is detected, in reaching out in VR to grab the object, a corresponding robot arm could reach out in the real world, to collect an object. This of course would need more immersive camera streams, point cloud mapping, and additional hardware, but the foundation of the project would be the same. There are many ways this project can be adapted and is highly usable in many industrial applications.

REFERENCES

[1] S. D. Sawyer, "Outdated controls instrumentation in nuclear power plants - strategies for extending useful life," in *2003 IEEE Nuclear Science Symposium. Conference Record (IEEE Cat. No.03CH37515)*, vol. 5, 2003, pp. 3617–3621 Vol.5.

[2] H. Yamada, N. Tao, and Z. DingXuan, "Construction Tele-robot System With Virtual Reality," in *2008 IEEE Conference on Robotics, Automation and Mechatronics*, 11 2008, pp. 36–40.

[3] D. Etherington. (2017, 10) MIT's remote control robot system puts VR to work. [Online]. Available: https://techcrunch.com/2017/10/02/mits-remote-control-robot-system-puts-vr-to-work/

[4] D. Whitney, E. Rosen, D. Ullman, E. Phillips, and S. Tellex, "ROS Reality: A Virtual Reality Framework Using Consumer-Grade Hardware for ROS-Enabled Robots," in *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2018, pp. 1–9.

[5] Byeong-Hyeon Moon and Jae-Won Choi and Kun-Tak Jung and Dong-Hyun Kim and Hyun-Jeong Song and Ki-Jong Gil and Jong-Wook Kim, "Connecting motion control mobile robot and vr content," in *2017 14th International Conference on Ubiquitous Robots and Ambient Intelligence (URAI)*, 2017, pp. 355–359.

[6] "Operating system images." [Online]. Available: https://www.raspberrypi.org/software/operating-systems/

[7] "Complete resource list." [Online]. Available: https://uofutah.sharepoint.com/:x:/s/SeniorProject2020/ERQggGwD3T1GqkDkzCyIo9wBD1VwtkLc7yjZxsO7-zZxoA?e=yweOul

[8] K. Castle. (2020, 10) Oculus Quest 2 Review. [Online]. Available: https://www.rockpapershotgun.com/2020/10/13/oculus-quest-2-review/

[9] "Project Site." [Online]. Available: https://uofutah.sharepoint.com/sites/SeniorProject2020

[10] "Telipito es karbantarto." [Online]. Available: http://mail.bekescsaba.hu:8080/mjpg/video.mjpg

[11] R. H. Espinosa, "Joystick API Documentation," 8 1998. [Online]. Available: https://www.kernel.org/doc/Documentation/input/joystick-api.txt

[12] A. Spirin, "pyPS4Controller." [Online]. Available: https://pypi.org/project/pyPS4Controller/