Journal of UAB ECTC
Volume 18, 2019
Department of Mechanical Engineering
The University of Alabama, Birmingham
Birmingham, Alabama USA

REAL-TIME VIRTUAL REMOTELY OPERATED VEHICLE

Tyler Gragg, Kevin McFall
Kennesaw State University
Marietta, Georgia USA

ABSTRACT

With the rise of Virtual Reality (VR) in the commercial market, more use cases are being discovered and implemented for this technology in the realm of remotely operated vehicles (ROV). This paper describes the process of creating a system that is able to demonstrate the uses VR can provide over typical ROV control schemes. This is shown through the use of an embedded system capturing live video footage on a four-wheel robot, sending the video wirelessly to a ground station, and displaying that video in VR to the operator. Simultaneously, the ground station is capturing inputs from the operator in VR, sending the controls to the ROV, and reacting in real-time to these inputs.

This method of using VR inputs for the end user allows for a more user-friendly control scheme and provides a familiar and easily customizable interface for the user. Without the constraints of a physical control panel for the ROV operator, changing the layout of the inputs, while keeping the same implementation for sending messages, becomes easier and cheaper.

KEY WORDS: *Virtual Reality (VR), Remotely Operated Vehicles (ROV), OpenCV, Unity*

INTRODUCTION

The use of virtual reality (VR) to operate remote vehicles shows promise to provide a large advantage over traditional ROV operator interfaces. Where physical remotes and work stations provide consistency and feedback, VR versions of this offer flexibility and customization [1]. This project demonstrates a use of customization by creating a virtual work environment and controlling a mobile robot with external devices including a horn, lights, and wheels.

Currently, other fields, such as flight simulators [2] or underwater mapping [3] allow for VR and non-VR control of virtual and remote systems, but every such project has a specific, custom configuration of the operational interface. This project attempts to create a system that is capable of working with any hardware system and that requires minimal adjustment. A simple new protocol is developed to integrate physical actuators with virtual inputs. While the system developed was developed to be adaptable for use in any system; in this demonstration, it is installed in a land-based remotely operated vehicle (ROV), shown in Figure 1.

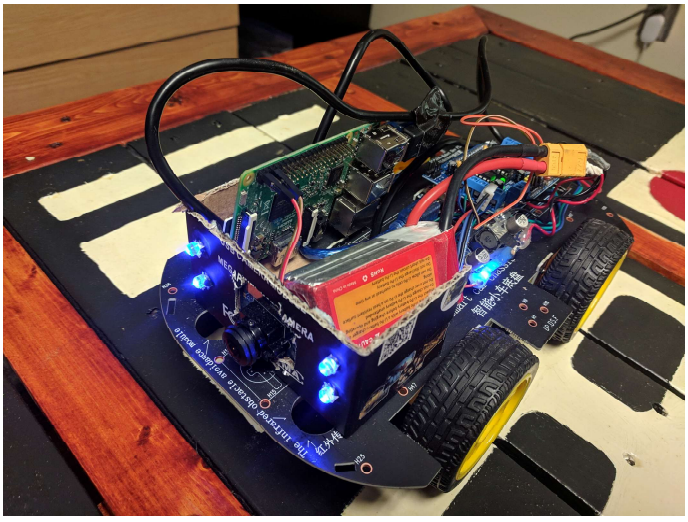

Figure 1. VRBot Proof of Concept

Projects are already exploring the use of VR to operate vehicles such as for underwater robots that can be controlled and operated in VR [4-6]. Another research project allowing users to tour a location using a tele-operated VR Robot [7] has been shown to be successful, but does not provide real-time control. NASA explored control of their complex systems using VR [8] and showed that using a system such as this allowed an operator to maintain improved control of craft operation and to maintain understanding of vehicle surroundings.

In addition, to facilitate adaptation to multiple hardware layouts, this work explores the ability to test control layouts without the need of any physical ROV hardware. Here, the simulation of an ROV and modification of the operator interface can be performed in a fully virtual environment. Within this environment, new features, such as lights or a speedometer, can be shown to the user, tested, and iterated within seconds.

HARDWARE STRUCTURE

The hardware loop in this project is composed of four main components working together though a software stack. The four components, appearing in Figure 2, are the ground station computer, the Virtual Reality headset, the embedded system running on the ROV, and the Arduino interfacing between sensors and other physical devices.

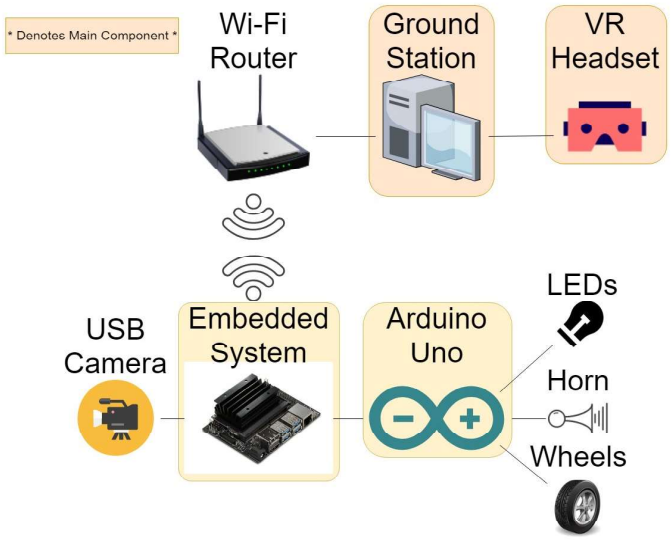

Figure 2. Hardware Structure

The ground station computer performs all of the heavy computation in terms of video processing power, as it is running a VR simulation world where the operator sits in a chair and is able to control the ROV. Connected to the ground station is the VR headset, an Oculus Rift in this case, which provides the real-time orientation and movement data to render and display a VR scene.

The ground station connects to the embedded system through a Wi-Fi link, which is managed by a router broadcasting a local network. As the software is designed to be hardware independent, any small and lightweight embedded system would work, but in this case a Jetson Nano and RaspberryPi were both tested and used. The embedded system pulls images from a USB camera and sends data to an Arduino Uno over a separate USB serial connection. The Arduino takes the information from the embedded system and controls actuators, lights, and a horn on the ROV.

SOFTWARE OVERVIEW

In order to achieve real-time performance, the goal of the software stack is for information to transmit from the ground station to the ROV as quickly and simply as possible to minimize latency. With this goal in mind, the embedded system is mainly responsible for passing information through it without modifying it or creating a bottleneck for the system as a whole.

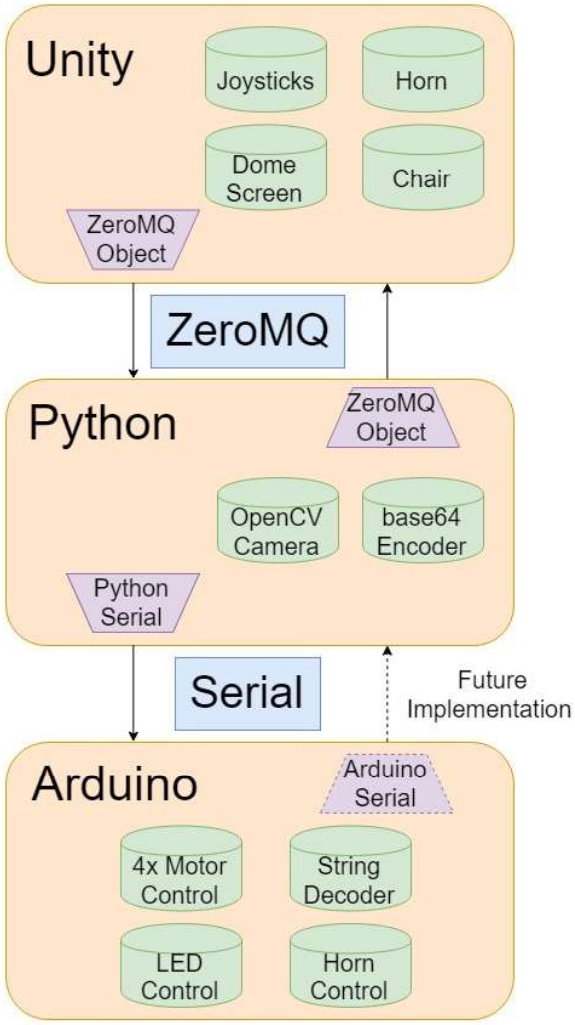

Figure 3. Software Structure

Three main nodes in the software architecture interact with each other through two different protocols. Figure 3 shows the three systems' main components and how they interact with each other. The first system, located within the ground station, is the Unity environment. Unity is a game engine that allows easy creation of a virtual 3D environment. Inside Unity, objects such as buttons and levers interact with each other, but only one object communicates outside of the Unity world. This object handles the passing of information to and from the embedded system on the ROV though the use of network sockets and the ZeroMQ network package.

Once information is sent through the network socket, it is picked up by a Python script running onboard the ROV. This Python process decodes the information from the socket using ZeroMQ and outputs base64 encoded images back through the socket into Unity. These images from the camera are captured, resized, and compressed using the computer vision library OpenCV. Simultaneously, the Python process converts the information from unity into a string encoded with information

that is relayed to the Arduino connected via serial USB. This string message is loaded with all the information about the desired state of the robot. This includes the desired state of the horn, lights, and wheel speeds. Because all this information is being passed through one message, all of these actions can take place simultaneously on the robot. For example, the user is able to simultaneously turn on the lights, and honk the horn.

Once the serial messages are in the Arduino, a short decoding method is called that takes the full string and extracts the important information from it. From there, the motors, connected to an Arduino motor shield, are sent a PWM signal and instructed to turn forward or backwards at a particular speed. The horn, a piezo quartz crystal buzzer, is also connected and sent a tone to play, emulating a car horn. Finally, a string of LEDs wired together form the headlights that can be toggled on or off.

UNITY ENVIRONMENT

Unity provides a framework with which to create virtual environments for a ground station operator to interact with the ROV in the real world. By creating various 3D objects in Unity, an entire virtual command station is created with rules that govern how they move and fit together.

Along with the base Unity assets, the SteamVR package is imported into the project providing a number of useful scripts and objects to facilitate quick and easy VR development and provides a Player object that automatically connects the VR headset to Unity. Along with the headset connection, SteamVR's scripts allow for turning wheels, pulling levers, or flipping switches by setting a few parameters in the Unity interface. These base interactions were used to create a simple control scheme where the user is presented with two joysticks, one on each side of the chair the operator sits in. The user's chair and control joysticks are shown in Figure 4. Pushing either joystick forward turns the corresponding set of wheels on the ROV forward. Likewise, pulling the joystick back will rotate the wheels in the opposite direction. This control method allows the user to turn around the ROV's center and pivot with ease.

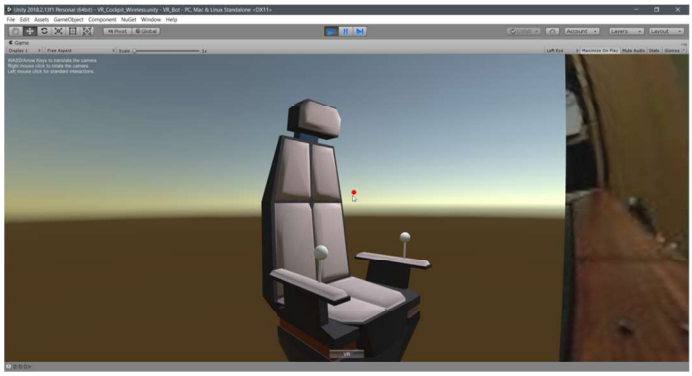

Figure 4. Unity Control Panel

The Unity environment attempts to give the illusion to the ground station operator that they are inside the ROV controlling it. Figure 5 displays a screenshot of the live video feed on a large virtual concave screen placed in front of the user in VR displaying the live video from a 180° camera. The screen simulates a world for the user representing a glass window from onboard a small craft. The operator is placed in a chair that automatically resizes to fit their body to the arm rests, and in front of the user is a button to emulate a horn for the ROV.

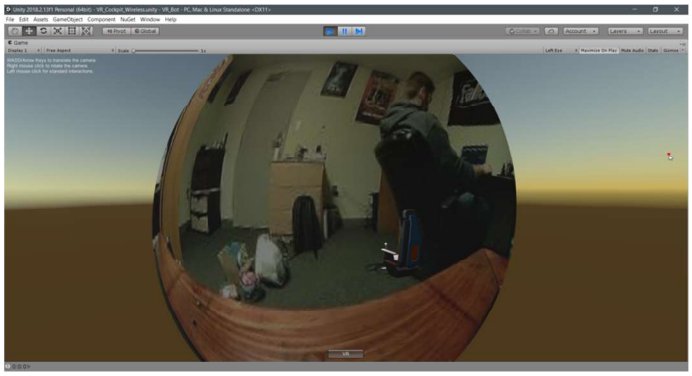

Figure 5. Unity Live Video

ZEROMQ IMPLEMENTATION

ZeroMQ is a networking library designed to make implementation of socket programming simple, easy, and scalable. This project uses ZeroMQ as a method for reliably and quickly sending text-based messages around a local network. Using ZeroMQ, real-time video can be streamed wirelessly, and commands can be distributed regardless of the target language or architecture in which they exist.

This implementation of ZeroMQ called for two nodes, a ground station and embedded system, to communicate with each other simultaneously and exclusively. From these requirements, a PAIR socket was chosen. A PAIR socket allows for two independent IP addresses to be bound together through a specific socket number. When a PAIR socket is implemented, no other traffic may communicate over that socket, and messages sent will only ever go to these two paired nodes. For this implementation, it was very convenient to know that the two nodes talking over the socket were only ever going to send messages to each other.

Along with the type of sockets being bound to, a message queue must be established. A protocol must be developed for situations such as when the embedded system sends two consecutive messages to the ground station before either is read. As this project was designed to be as close to real-time as possible, it was decided that newer messages should take precedent over old messages. Because of this, a CONFLATE option is passed to ZeroMQ with a value of one. This configures the ZeroMQ object to only ever store one message at a time, and, if a new one comes in, it should replace the old.

RESULTS AND FUTURE IMPLEMENTATIONS

Feedback was solicited from several operators in an attempt to confirm proper software operation and test the VR interface's ease of use. Initial reactions indicated the VR controls were easy to pick up, but difficult to master. The

ROV's top speed of approximately 3 m/s is fairly hard to operate in an enclosed space. A second limitation of the project is the requirement of a stable local Wi-Fi network for continuous operation of the vehicle. The router used provided a limited range of roughly 30 ft. with no object in the path. As the ROV ranges further from the Wi-Fi router, the slower video feed makes operation of the craft more difficult.

Future plans for the project include the integration of neural networks and more advanced computer vision into the system. A plan to provide object tracking on the live video involves the user identifying a desired object to track by defining a bounding box on the dome screen in VR. The bounding box would be passed to the embedded system, where either a neural network script or a computer vision algorithm would be used and track the object as it moves in the operators view.

In addition to object tracking, a PID implementation for speed control is desired for the ROV. By adding an inertial measurement unit and/or wheel encoders to the embedded system, ROV velocity can be calculated and controlled. Once the current velocity is measured, it can be compared against a desired user-defined speed and fed into a closed PID loop to drive the error toward zero.

Finally, creating a customizable user interface for the user is desired. For the user to feel natural when operating any kind of craft, they need to be able to customize the interface for comfort and familiarity. Creating tools for the operator to make adjustments to the size, placement, colors, and orientation of their workspace is important to allow them to feel in control of the ROV.

ACKNOWLEDGEMENTS

This project would not have been possible without the help and support from Dr. Kevin McFall, Kennesaw State University's Chair of Mechatronics Engineering. The University of Washington's annual hackathon, Dubhacks, also played a large role in providing an interest in Virtual Reality in addition to initial access to an Oculus Rift. Final thanks also go out to some of the initial testers and friends that were there to offer feedback for this project, Trevor Stanca, Brian Roney, and Jessica Brummel.

REFERENCES

[1] Dorneich, Michael Christian, Stephen Whitlow, William Rogers, Karen Feigh, and E. Robert. "Adaptive user interface for semi-automatic operation." U.S. Patent 8,977,407, issued March 10, 2015.

[2] Chin, Cheng Siong, Nurshaqinah B. Kamsani, Xionghu Zhong, Rongxin Cui, and Chenguang Yang. "Unity3D serious game engine for high fidelity virtual reality training of remotely-operated vehicle pilot." In *2018 10th International Conference on Modelling, Identification and Control (ICMIC)*, pp. 1-6. IEEE, 2018.

[3] Martins, Alfredo, José Almeida, Carlos Almeida, Bruno Matias, Stef Kapusniak, and Eduardo Silva. "EVA a Hybrid ROV/AUV for Underwater Mining Operations Support." In *2018 OCEANS-MTS/IEEE Kobe Techno-Oceans (OTO)*, pp. 1-7. IEEE, 2018.

[4] Hine, B., Carol Stoker, Michael Sims, Daryl Rasmussen, Phil Hontalas, T. Fong, J. Steele et al. "The application of telepresence and virtual reality to subsea exploration." In *Second Workshop on Mobile Robots for Subsea Environments*. 1994.

[5] Bonin-Font, Francisco, Miquel Massot Campos, and Antoni Burguera Burguera. "ARSEA: A Virtual Reality Subsea Exploration Assistant." *IFAC-PapersOnLine* 51, no. 29 (2018): 26-31.

[6] Candeloro, Mauro, Eirik Valle, Michel R. Miyazaki, Roger Skjetne, Martin Ludvigsen, and Asgeir J. Sørensen. "HMD as a new tool for telepresence in underwater operations and closed-loop control of ROVs." In *OCEANS 2015-MTS/IEEE Washington*, pp. 1-8. IEEE, 2015.

[7] Oh, Yeonju, Ramviyas Parasuraman, Tim McGraw, and Byung-Cheol Min. "360 vr based robot teleoperation interface for virtual tour." In *Proceedings of the 1st International Workshop on Virtual, Augmented, and Mixed Reality for HRI (VAM-HRI)*. 2018.

[8] Nguyen, Laurent A., Maria Bualat, Laurence J. Edwards, Lorenzo Flueckiger, Charles Neveu, Kurt Schwehr, Michael D. Wagner, and Eric Zbinden. "Virtual reality interfaces for visualization and control of remote vehicles." *Autonomous Robots* 11, no. 1 (2001): 59-68.