UNIVERSITY OF TARTU
Institute of Computer Science
Computer Science Curriculum

Amirabbas Sherafatian

Teleoperation of Remote Controlled Toy Cars in VR

Master's Thesis (30 ECTS)

Supervisor: Ulrich Norbisrath, PhD

Tartu 2022

Teleoperation of Remote Controlled Toy Cars in VR

Abstract:

Teleoperation, also known as remote operation, is a process enabling a vehicle, robot, or machine to be operated remotely. So that it establishes a two-way communication channel between the operator and the remote-controlled machine, which is called a teleoperator. Teleoperation is linked with several technologies, concepts, and terms such as telerobotics, telepresence, Virtual Reality (VR), and Augmented Reality (AR) and consists of several local and remote components [1].

Many areas benefit from teleoperation. The prominent examples are Autonomous Vehicles (AVs), Remotely Operated Vehicles (ROVs), drones, and entertainment. As a result, humans are able to explore environments and perform tasks via remote machines, without having to be there physically.

In this thesis, while the focus is to provide a minimum viable product, a practical solution to teleoperate toy cars (Donkey Car S1[2]) in VR is introduced. VR is, in fact, a technology enabling users to immerse into an environment simulated with computers and interact with it[3]. As VR users take the advantage of immersive display and feel completely involved in the activity[3], the decision has been made to render the feedback (camera images) from the car in the VR glasses (Oculus Quest 2 - rebranded as Meta Quest 2 in November 2021[4]). As a result, the operator feels like being inside the car and will experience driving with the use of joysticks.

The provided solution has been successful in providing a teleoperation opportunity, while it is open for improvement in some areas. The result shows with around 40 milliseconds of delay, the operator connects to the remote car successfully and is able to operate it. The camera images with a frame rate of 30 fps and a resolution of 320X240 are rendered in the Meta Quest successfully and the operator is able to send the commands to the remote car via joysticks in real-time. These prove that the solution is practical, but there are some areas open for further study and improvement such as camera streaming, network structure, and telepresence.

The provided solution and topics discussed in this thesis yield a basic understanding of teleoperation. Getting to know the concepts and challenges in teleoperation help to step into some areas, where teleoperation is inevitable such as autonomous driving and telerobotics. The thesis will also motivate students and give them a direction in some areas such as teleoperation and VR.

Keywords:
Teleoperation, Video Streaming, Network and Internet Protocols, Teleoperation in Virtual Reality, Digital Twins.

CERCS:P170 Computer science, numerical analysis, systems, control

Kaugjuhitavate mänguautode teleoperatsioon virtuaalreaalsuses

Lühikokkuvõte:

Teleoperatsioon, tuntud ka kui kaugjuhtimine, on protsess, mis võimaldab sõidukit, robotit või masinat kaugjuhtida. See loob kahepoolse sidekanali operaatori ja kaugjuhitava masina vahel, mida nimetatakse teleoperaatoriks. Teleoperatsioon on seotud mitme tehnoloogia, kontseptsiooni ja terminiga, nagu telerobootika, telekohalolu, virtuaalreaalsus (VR) ja liitreaalsus (AR) ning koosneb mitmest kohalikust ja kaugkomponendist [1].

Paljud valdkonnad saavad teleoperatsioonist kasu. Silmapaistvad näited on autonoomsed sõidukid (AV), kaugjuhitavad sõidukid (ROV), droonid ja meelelahutus. Tänu sellele saavad inimesed kaugmasinate kaudu keskkondi uurida ja ülesandeid täita, ilma et nad peaksid seal füüsiliselt viibima.

Kuigi käesolevas lõputöös keskendutakse minimaalse töötava toote pakkumisele, tutvustatakse ka praktilist lahendust mänguautode kaugjuhtimiseks (Donkey Car S1[2]) VR-is. VR on tegelikult tehnoloogia, mis võimaldab kasutajatel sukelduda arvutitega simuleeritud keskkonda ja sellega suhelda[3]. Kuna VR-i kasutajad kasutavad kõikehõlmavat ekraani ja tunnevad end tegevusest täielikult kaasatuna[3], on tehtud otsus renderdada auto tagasiside (kaamerapildid) VR-prillides (Oculus Quest 2 – ümber nimetatud kaubamärgiks Meta Quest 2 novembris 2021[4]). Selle tulemusel tunneb operaator end nagu autos ja kogeb sõitmist juhtkangide abil.

Pakutav lahendus on teleoperatsioonivõimaluse pakkumisel olnud edukas, samas on see mõnes valdkonnas avatud edasiarendamiseks. Tulemus näitab umbes 40 millisekundilise viivitusega, et operaator loob edukalt ühenduse kaugautoga ja suudab seda juhtida. Kaamera pildid kaadrisagedusega 30 kaadrit sekundis ja eraldusvõimega 320x240 renderdatakse Meta Questis edukalt ning operaator saab juhtkangide abil reaalajas käsud kaugautosse saata. Need tõestavad, et lahendus on praktiline, kuid mõned valdkonnad on avatud edasiseks uurimiseks ja täiustamiseks, näiteks kaamera voogesitus, võrgustruktuur ja telekohalolu.

Antud lahendus ja selles lõputöös käsitletavad teemad annavad põhiteadmised teleoperatsioonist. Teleoperatsiooni mõistete ja väljakutsetega tutvumine aitab astuda teatud valdkondadesse, kus teleoperatsioon on vältimatu, näiteks autonoomne sõit ja telerobootika. Lõputöö motiveerib ka tudengeid ja annab neile suuna valdkondades nagu teleoperatsioon ja VR.

Võtmesõnad:

Teleoperatsioon, video voogesitus, võrgu- ja Interneti-protokollid, teleoperatsioon virtuaalreaalsuses, digitaalsed kaksikud.

CERCS:P170 Arvutiteadus, arvutusmeetodid, süsteemid, juhtimine (automaatjuhtimisteooria)

Acknowledgements

I would like to express my gratitude to my supervisor, Dr. Ulrich Norbisrath, who gave me support and assistance in the right direction.

I would like to thank my colleague, Erko Aaberg, who helped me with Estonian translation.

Contents

List of Figures

List of Tables

Listings

1 Introduction

Teleoperation is a technology providing a remote access to a machine to operate. In fact, this area establishes a communication channel between the human and a machine so that the human is provided an opportunity to control a machine remotely for a final task. Technically, the term Teleoperation is used in the academic context to convey operating a machine from a distance, while the distance varies from centimeters to million miles away [20].

Having this in mind, an inevitable part of teleoperation, which is significantly noticeable, is the Human-Machine Interface (HMI), but it is impressive if the technology VR is introduced into this layer. VR is a perfect technology to migrate humans into a virtual environment. In fact, with the help of VR, a computer 3D-based environment is generated that is simulating the real world (I will refer to the VR-based interface as RQ1). Therefore users will find themselves in the simulation and in any possible environment generated, while they are not physically there. VR also enables users to interact with the 3D objects and receive feedback, which ultimately leads to Metaverse. As the paper [21] says, "It is based on the convergence of technologies that enable multisensory interactions with virtual environments, digital objects and people such as virtual reality (VR) and augmented reality (AR)".

Truly, the combination of teleoperation and VR will result in noticeable telepresence. Telepresence is a technical term that appears in the context of teleoperation that helps humans experience being in a real environment virtually. In other words, with telepresence, the operator is transferred to virtual places simulating the real world and is interacting in that place, while they are somewhere else[22, p. 1-5]. As a result, with telepresence, one can feel being inside a machine, see what exactly the machine sees, and interact with that environment.

As the basic concepts are an inevitable part that gives a direction to provide a solution, I discuss the theoretical topics in the beginning. Together with the basic topics, the challenges involved such as Video Streaming and Encoders, Network Structure, Bandwidth, and Latency together with Internet Protocols are discussed. The challenges in teleoperation in VR are also covered in this section.

I also cover the technology Digital Twins which can be considered as a type of teleoperation. Digital Twins are, in fact, digital counterparts of physical objects with a two-way connection established between them. Consequently, any changes to the digital objects will reflect the physical counterpart in real-time.

With this thesis, I provide a minimum viable product to teleoperate toy cars in VR. In fact, the users are shrunk into the toy cars and feel themselves inside them to experience driving. With the practical work, the operators drive the toy cars (Donkey Car S1) remotely via Joysticks (I will refer to the user interaction via the joysticks as RQ2) and video data (camera images) from the cars are represented in the VR glasses (Meta Quest 2) (I will refer to the video streaming as RQ3). The integration of mentioned

components and a communication link is an essential part of the solution that I will refer to the communication link as RQ4. Lastly, with some adjustments, the solution can be applied to other scenarios, as long as the remote machine supports Python and the local component has an Android-based rendering engine.

1.1 Motivation

When it comes to teleoperation, there exist many areas that benefit from it. Since teleoperation aims to control a machine remotely, then it opens a gate to discover new environments, regardless of whether it is underwater, space, or underground meaning that humans will be able to step into places that were already not possible or were hazardous. Additionally, As this work shares the basic concepts of teleoperation, then it is beneficial to topics such as autonomous driving or even telerobotics. Because these are the areas in that teleoperation is inevitable. In the future, it is possible to give the solution a new direction to be integrated into these topics.

There are more areas than the ones mentioned that the solution can be integrated into. For instance, entertainment is another area, where AR can be added to the solution to come up with 3D games. AR is, in fact, a technology that adds computer objects to the real world in real-time - it is different from VR [23].

While this thesis motivates students to study teleoperation-related topics, it is also beneficial for interested students in this direction. This thesis discusses the basics of teleoperation so that it will help with onboarding students by giving them fundamental knowledge and broadening their horizons. Also, the solution showcases the university equipment (Donkey Car and Meta Quest) which are additional stimuli to motivate students. Moreover, since the provided solution is open for improvements, students have the opportunity to join to improve areas such as VR, Telepresence, Video Streaming, and Network Structure, or give them a new direction to autonomous driving.

Lastly, with this work, a VR-based mindset is built. Having this mindset will drive us to think in the direction of VR. This will make a significant difference when stepping into the teleoperation areas. For sure, the difference is noticeable when the operators not only are communicating remotely with the machine, but also feel being there.

1.2 Contribution

The goal of the thesis is to provide a minimum viable product to teleoperate a Donkey Car S1 in the VR glasses Meta Quest. I provide two services, as a remote application, that are running on Raspberry Pi (remote machine). One service is meant to teleoperate the car (I will refer to the teleoperation service as RQ5) and one service is for video streaming. I also provide an Android-based client application that is installed on the Meta Quest. The remote and client applications are connected to each other through a

communication channel using a tp-link router. The solution is discussed in detail in its own chapter (section 3), but before that, I am going through the theoretical chapter 2.

As the theoretical part is necessary to build the ground and understand the context, I dedicate the beginning chapter to the theoretical part. In this chapter, the basic concepts of teleoperation are discussed and I explain how they are connected to the practical solution. Together with discussing the challenges in teleoperation and VR, I also introduce Digital Twins. Digital Twins can be comparable to teleoperation. In fact, they can be investigated as a form of teleoperation.

1.3 Thesis Outline

The thesis is structured as follows:

- **Chapter 2**: in this chapter, I describe the main concepts of teleoperation and how they are related to the practical solution. Also, together with teleoperation architecture, video streaming, network, and teleoperation in VR, Digital Twins are also discussed in the chapter.
- **Chapter 3**: I basically explain the provided solution in detail in this chapter. It is described what are the main hardware components together with the analysis to come up with the architecture. I also introduce the remote application (the services running on the remote machine) together with the client application in this section.
- **Chapter 4**: this chapter presents the result and conclusion. I explain the future work and open areas for improvement in this chapter.

2 Teleoperation

In this chapter, I discuss the basics of teleoperation and will define the terms included in the context of teleoperation. Also, I am going through the classifications of teleoperation together with some of the challenges in the area. Lastly, I will step into the teleoperation in VR and Digital Twins.

2.1 Introduction

Teleoperation means operating a machine from a distance, as the distance can vary from million miles away to centimeters (see Figure 1) [5]. As Figure 1 shows, while the operator is away from the remote machine, sends the commands using the control station and through the network to the remote machine. Afterward, the machine performs the task and will send back the feedback to the operator so that the operator is able to make the correct decisions to interact further with the machine. Teleoperation is mostly used in two scenarios: where it is required by the user to complete a task remotely and where the environment is dangerous or problematic to reach [24].

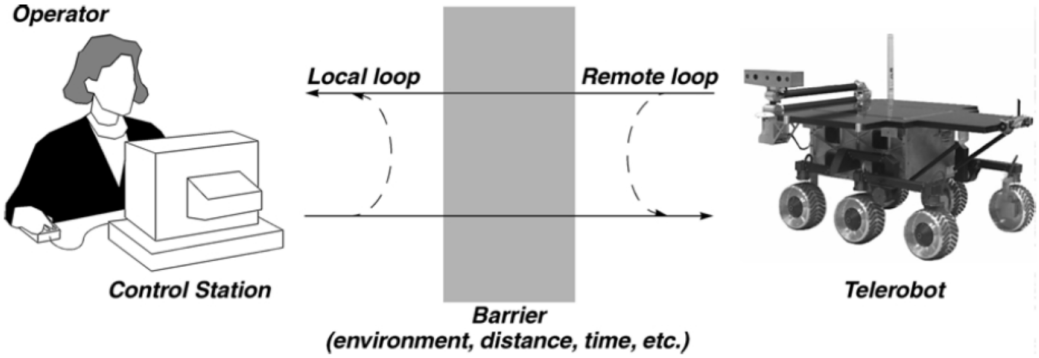

Figure 1. Vehicle teleoperation [5]

Before explaining the main components of teleoperation, first, I describe the definitions.

2.1.1 Definitions

Here the basic definitions that help readers understand the topic and the thesis are explained.

Teleoperation: as the Wikipedia explains: "Teleoperation (or remote operation) indicates operation of a system or machine at a distance. It is similar in meaning to the phrase "remote control", but is usually encountered in research, academia and technology.

It is most commonly associated with robotics and mobile robots, but can be applied to a whole range of circumstances in which a device or machine is operated by a person from a distance.".

Operator: is a person who monitors the remote machine and will send the commands to the remote end. The operator is an inevitable part of teleoperation even if the machine is capable of performing the task autonomously - it is a part of the regulation.

Teleoperator: is the machine in a remote site that is operated by the operator. The sophisticated teleoperator is also called a telerobot.

Sensors and Actuators: sensors and actuators are used in the remote machine. Sensors are to observe the changes in the environment to respond back. If they are not ideal or used inappropriately then the deviation is noticeable. Besides sensors, actuators are the machine components that respond to the commands received at the teleoperator end so the output will be mechanical motion. In fact, they are responsible to move the teleoperator [25, p. 1-8].

telepresence: "Telepresence can be explained as the quality of a teleoperation experience. Ideally, the information from the remote environment (visual, aural, haptic, etc.) is displayed in such a way that the operator 'feels' as if he/she is actually present at the remote environment"[24].

Autonomous Vehicles: the vehicles that can navigate themselves to the destination so that no need for a driver to be inside. The vehicles that are teleoperation enabled are considered semi-autonomous vehicles.

2.1.2 Components

The components of the teleoperation are categorized into local and remote components. As Figure 1 shows, the main components of teleoperation are the operator, the station, and the remote machine. The operator together with the station, with which commands are sent to the machine, sit locally and are considered local components. Besides, the machine in the remote site will receive the commands and operate properly so that the remote machine is considered a remote component [26, p. 37].

In Figure 1, the station acts as an interface including a display and control input. The display is meant to represent the feedback e.g. sensor metrics, and camera, from the remote machine. Once the operator receives the feedback, then further decisions are made and sent to the remote site via control input, which is a part of the interface.

The last main challenging component is the link. In fact, the communication channel is a significant member of teleoperation connecting the local and remote components together. As Figure 1 depicts, several factors impact the communication link e.g. the environment and distance. In most cases, considering the task, it is implemented wirelessly, however, still a wired connection is an option.

2.1.3 Vehicle Teleoperation Applications

Vehicle teleoperation has been employed to tackle the missions in the air, on the ground, and underwater, while the efforts in the area have been started in the 19th century when the first examples came out. One impressive example is the Sojourner landed on the Mars in 1997 for the discovery (see Figure 2).

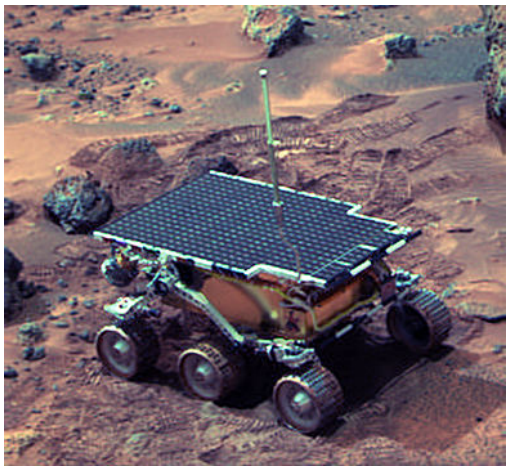

Figure 2. Sojourner - taken from Wikipedia

Air Vehicles. Air vehicles (AVs) as they are also called unmanned aerial vehicles or drones are one of the most popular areas. The applications in this area target the discovery and exploration mostly, while there are military purposes as well like anti-aircraft training. US Army's RP-5 (1941), which flew pre-programmed routes, and the US Air Force Predator, which was being teleoperated from the ground (see Figure 3) are examples in this area[5]. Additionally, there are also applications for other purposes e.g. delivery services, as Amazon has introduced its drone "Prime Air" on this popular ground. These main purposes are not just specific to the air vehicles, ground and underwater vehicles are also employed for similar missions.

Remotely Operated Vehicles. The remotely operated vehicles (ROVs) can be utilized for different kinds of purposes on the ground. They are also considered for the tasks underwater, which in that case are known as remotely operated underwater vehicles (ROUVs). There are applications in different areas such as research, probes, military, and delivery services and these vehicles are beneficial in that. Behind these applications, teleoperated vehicles are introduced into environments that are contaminated or dangerous for humans. For instance, the ROBDEKON project can be referred to, where four research institutions work on robots to get humans out of unsafe zones[27]. Additionally, there are research and projects for industrial machinery in construction that help with

Figure 3. Predator with the operator on the ground for teleoperation [5]

labor shortages and their safety. An interesting recent example is the walking excavator HEAP (Hydraulic Excavator for an Autonomous Purpose)[6] (see Figure 4).

Figure 4. HEAP, the first of its kind for autonomous deployment, in front of its clone Armano, a proof of concept for teleoperated/semi-autonomous operation [6]

2.2 Related work

2.2.1 An Internet Accessible Telepresence

As presented in the paper[7], the authors have provided an internet-accessible, remote-controlled model car that provides the driver's view using a camera on it (see Figure 5). As a result, the car is equipped with a camera and is teleoperated from miles away. The control station is connected to the remote machine via a 56kbps link and receives the video data with the use of an already-made software (NV) for video broadcasting[28]. In

fact, they highly compress the video data in a way that the output is compatible with the NV software so that it can be transmitted to a large number of receivers.

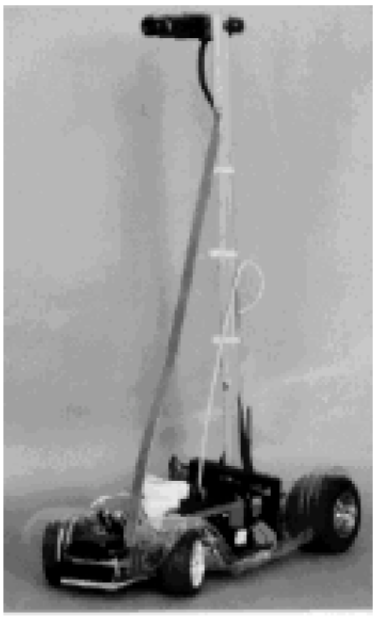

Figure 5. Remote controlled car [7]

As Figure 5 shows, a Sony 999 camera is installed eighteen inches above the car to give a realistic view to the end user. Also, the car is equipped with a Microtek 915 Mhz video transmitter that is used for audio transmission as well. Moreover, there is a controlling system connected to the car to receive the video and audio from the car to transmit, which is considered a server for the remote control station.

Considering the local components, there is a remote control station, which receives the video data and teleoperates the car. A computer application is available in the remote station that tracks the mouse and displays the video. In other words, the control station has two processes running, one of them will display the video and another one send the mouse information as control data. In fact, the car is teleoperated via a mouse. Figure 6 shows their architecture.

It is noticeable that they have used the UDP protocol for video data and the TCP protocol for control data. The UDP protocol is stateless and does not acknowledge if the packets are lost. Thus, the solution should benefit from either some recovery algorithms for lost packages or a mechanism to retransmit the lost packages, otherwise jumps will be noticeable in the video. Unfortunately, the paper has not stated details regarding this issue which is significant.

Also, they state that the video data is highly compressed, which is prominent. The teleoperation solutions should pay attention to the bandwidth to transmit data reduced

in size, while the quality is not sacrificed. Because teleoperation needs continuous data transmission in real-time. Thus, the lack of bandwidth will result in a delay. It is remarkable that, by optimizing the compression, they have reached from about 1.5 fps to 15 fps for video data.

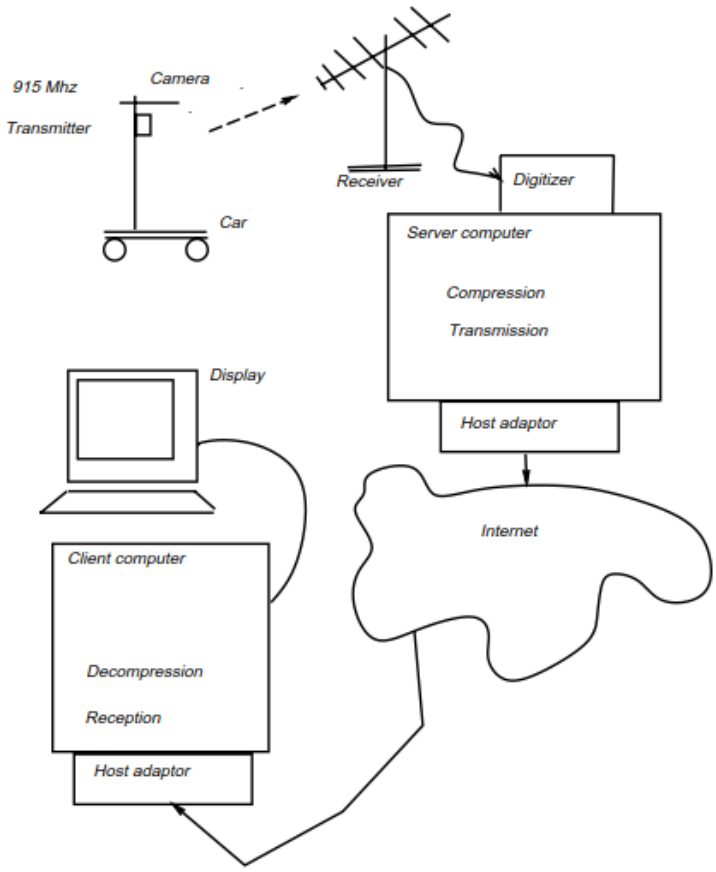

Figure 6. Architecture applied over the internet [7]

2.2.2 NeCS-Car

The NeCS-Car is a project for teleoperation by Networked Control System (NeCS) team in that the operator receives the camera images to remotely control a 200 KG car. The car has a speed of up to 10 M/S and also sends force feedback recognized by a D.C motor, to the operator [8]. This means that the project provides a higher level of telepresence compared to the project mentioned earlier in section 2.2.1.

Considering the remote component, there is a computer and two cameras installed on the car (see Figure 7). The computer has two operating systems that can be considered as two separate PCs connected to each other via ethernet. One operating system (OS) is responsible for camera streaming and another one for steering the vehicle.

Figure 7. NeCSCar - remote controlled car [8]

Considering the local component, in the operator station, there are two PCs, one for remote operation and one for video display, which similarly are connected via ethernet to each other (see Figure 8). All computers have Windows XP as their OS. The operator station, as a master component, eventually is connected to the slave component (remote car) via WLAN (Wireless Local Area Network) and a 100 Mbps switch. This means that they have applied a hybrid network architecture in which two ethernet networks on two sides are connected to each other via a WLAN.

The solution benefits from the UDP and TCP protocols. The UDP protocol is meant for fast data transmission, while the delivery of the message is not guaranteed. In NeCScar, this protocol is used to transmit control data and in case of packet loss, they use a mechanism based on the previous values received to compensate for data loss. As it is stated in the paper[8], around 20 percent of data sent using UDP is lost, but as they mentioned, the controller's performance is not influenced. Behind UDP, they use TCP for camera images. Because TCP will do retransmission of lost packages so that video quality is not affected, although TCP is slower compared to UDP. This project streams MPEG-4 compressed video data and the images have a size of around 40 KB. As they described, the images are sent every 40 MS presenting the delay.

Figure 8. NeCSCar - the operator station [8]

As they are using two separate pairs of client-server systems: one pair for teleoperation and one pair for video data, it defines a way of improvement. By this approach, in fact, the workload for video data and teleoperation is split into two separate systems so that they cannot affect each other. To improve the provided solution, this can be taken into consideration.

Lastly, similar to the project mentioned earlier in section 2.2.1, this project has been built over UDP and TCP protocols, but they use TCP for video data and UDP for teleoperation. This means that TCP and UDP are the protocols that are usually under the focus as the first options when it comes to teleoperation, and depending on the task, analysis, and existing situation, they are used for video data and teleoperation services.

2.2.3 Voysys

Here I am introducing a solution to driving teleoperation from Voysys[9]. The Voysys solution is based on their own software Oden. Oden is a modular software solution that can be applied to different kinds of vehicles to help them with being teleoperated. It can stream high-resolution video up to 8K and is written with C, C++, and Rust together with the rendering library OpenGL.

Oden streamer and Oden VR are the main modules of the software Oden. Their responsibilities are to deal with streaming the video and rendering video data into a display. Oden streamer is the module to receive data from varying sources such as cameras and files to process. This is where the Voysys encoder comes into place to encode the data before transmission. The encoder encodes the data into H.264 standard, while there will be a move to HEVC(H.265) from Voysys. After transmission, the Oden VR module is responsible to decode and render the transmitted data. This module

involves a 3D render engine that can render the data into several environments such as one single monitor, multiple projectors, and VR glasses.

Currently, Voysys is using Nvidia Jetson Nano, which is a small and powerfull computer for video encoding. This computer has a Quad-core ARM A57 @ 1.43 GHz processor, 128-core Maxwell GPU, and 4GB RAM - Oden streamer is running on this computer to process, encode and stream the video. However, they have started to move to Jetson AGX Xavier. This is a bit larger, but more powerful computer than Jetson Nano. It has an 8-Core ARM v8.2 64-Bit CPU, 512-Core Volta GPU with Tensor Cores, and 32 GB RAM. Although it is not mentioned what is the main reason for their migration, it can be guessed that this is the cost of encoders. The encoders need to encode video data so fast, otherwise, the encoders themselves are a source of delay. Thus, having a more powerful computer may help with a higher resolution, less delay, or both.

Besides, they have a dome setup (see Figure 9) and a three-screen setup (see figure 10) to represent the camera images. For each setup, they use a computer to decode the data transmitted to display the video. As a result, the data from the camera on a car (the teleoperator) is directed to their render engine and will be displayed in their setup.

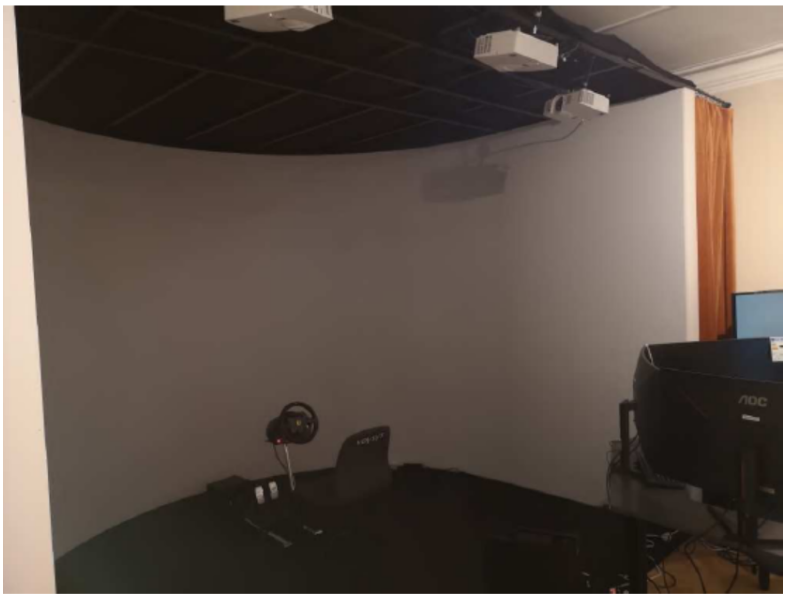

Figure 9. Voysys dome setup [9]

Voysys is a company providing solutions to teleoperation of vehicles to other companies. Thus, companies working on autonomous vehicles can integrate into its solution and, as a result, get their autonomous vehicles teleoperated. Einride[29], as an example, is one of the customers of Voysys that is working on autonomous trucks for transportation.

There are more companies working on providing teleoperation solution to vehicles such as DriveU[30], Ottopia[31], Formant[32], Teksbotics[33] and more. As an example,

Figure 10. Voysys three-screen setup [9]

DriveU has introduced their remote operation solution for robots and it's available on NVIDIA® Jetson™ platforms, which is a leading platform for autonomous machines. In other words, the solution is applied to the robots that come with NVIDIA Jetson platforms and use the robot's existing camera.

The Voysys teleoperation solution emphasizes the importance of encoders. On one hand, the transmission of video data without appropriate encoders is costly and the cost is either having video data with less quality or delay, as the bulky video data needs to be transmitted. On the other hand, encoding video data means workload. So the remote machine should be equipped with appropriate and powerful hardware, depending on the video quality is going to be achieved.

In this section, I discussed the related works. Truly, going through these works help with the analysis of the problem and I emphasized the elements necessary to be taken into the consideration for each work such as encoders and video data compression, protocols needed for data transmission, system architecture, and telepresence. The elements discussed are the issues in common between most of the teleoperation solutions, even though the provided solution in this thesis is based on VR glasses. This is how the work in this thesis differs from the related works discussed here.

2.3 Classifications

Here I am explaining the categories of teleoperation. The teleoperation solutions are classified into three categories of Closed-loop Control, Coordinated Teleoperation, and

Supervisory Control generally (see Figure 11)[10].

2.3.1 Closed-loop Control

In this category, also known as direct teleoperation, there is no autonomy at all and the operator is controlling the machine directly. In fact, with the use of control input as a part of the Human-Machine Interface (HMI), the decisions are sent to the actuators of the teleoperator and will result in a mechanical movement of the slave machine. Then the sensors send back the changes in the remote environment as feedback to the HMI. The feedback, consequently, is represented to the operator via the display in HMI for further decisions. The video data (camera images) almost in all teleoperation solutions is a part of the feedback. This direct teleoperation is feasible when there is no delay or it is compensated. In a case of delay, the strategy "move-and-wait"[34], depending on the task, is a famous approach to follow. We describe this strategy in the following section.

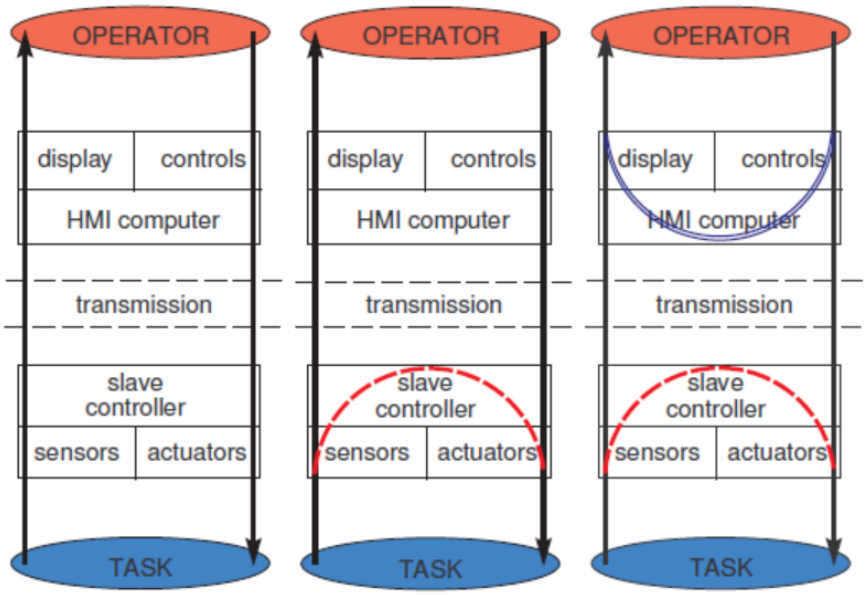

Figure 11. Teleoperation classifications: Closed-loop Control (the most left), Coordinated Teleoperation, and Supervisory Control (the most right) [10]

2.3.2 Coordinated Teleoperation

This type of teleoperation is similar to direct teleoperation, but there exists a remote loop. In this category, the operator controls the actuators of the machine, but additionally, there is a remote loop at the teleoperator end (the red dotted curve in Figure 11 depicts the

remote loop). In coordinated teleoperation still, no autonomy is added to the solution, but there's an additional internal control loop for some particular moments when it is problematic for the operator to operate the machine; for instance, when due to a communication delay, the operator cannot handle operate the remote machine properly. In some cases, there can be more than one operator to operate the teleoperator.

2.3.3 Supervisory Teleoperation

In this type of teleoperation, autonomy is added to the remote end. This is the factor distinguishing this class from the two previous categories. In fact, in this class, still the operator controls the machine, but with the use of high commands. For instance, the operator just sends the coordination of the destination and will not control the machine through the path. Supervisory teleoperation, actually, is a standard form of teleoperation and can be sub-categorized into autonomous and semi-autonomous teleoperation. This type of teleoperation is considered when the delay is inevitable because the commands are limited to high commands only. As a result, the number of commands is reduced and less bandwidth is consumed.

The solution I provide in this thesis falls into the Coordinated Teleoperation category. Actually, the operator controls the car with hand controllers continuously and as long as he sends the commands, the car will move. In other words, as long as the operator pushes the forward button on the joysticks, the move forward command is continuously sent, and meanwhile, if the car gets disconnected since the last command was to move forward then the car will move forward while it is disconnected. In fact, it uses the last command received to continue moving. When the connection is back, then the operator can take control of the car again. This can be considered as an autonomy added to the car.

2.4 Strategies

When there is a delay in the loop, then direct teleoperation is not feasible. In this situation, the tendency is toward having supervisory teleoperation. It means that by adding autonomy to the remote end, the delay is compensated because there is no need to send further commands. As a result, the payload is reduced. There are strategies to apply in case of a delay. Here I am describing two famous strategies.

2.4.1 Move and Wait

The strategy move-and-wait requires the operator to wait until the previous task is completely executed on the teleoperator side[11]. For instance, if the machine wants to reach the destination (the point F) (see Figure 12), the operator needs to send nine commands (see Table 2.4.1), together with the parameters.

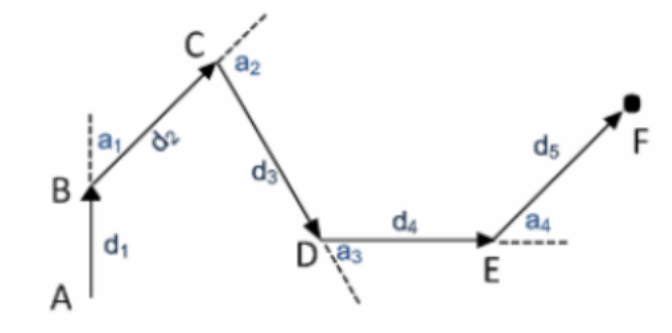

Figure 12. Move and wait - path to travel [11]

At first, the commands are defined as "move forward" (MF), "move backward" (MB), "turn left" (TL), "turn right" (TR), and "stop" (ST). Then at the point of A, by sending the command MF and the parameter d1, the teleoperator goes forward until it reaches point B. At this point when the previous task is done, then two more commands will be executed. The first command is to turn the robot to the left to some degree, and the second command is to make the machine move forward. We need these two commands to be executed with their parameters at the points of C, D, and E. It is important to note that each command will get executed when the previous command has been executed fully.

Table 1. "Move and Wait" commands with the parameters [11]

Command	**parameter**	**location**
MF	Distance d1	A -> B
TR	Distance a1	B
MF	Distance d2	B -> C
TR	Distance a2	C
MF	Distance d3	C -> D
TL	Distance a3	D
MF	Distance d4	D -> E
TL	Distance a4	E
MF	Distance d5	E -> F

As Table shows, there are many commands that need to be sent together with the parameters. This is, in fact, the downside of the move-and-wait strategy.

2.4.2 Speed Control

The aim of the speed control strategy is to reduce the number of commands in comparison to the move-ant-wait strategy. With the speed control, the two commands of MF and MB are replaced with "No Stop Move Forward" (NSMF) and "No Stop Move Backward" (NSMB). Consequently, when the machine receives one of these two commands, it goes all the time either forward or backward until it receives the stop command (ST).

Considering the previous example, with this strategy, the command NSMF is sent at first and the remote machine starts to move forward. Then at the point of B, it needs just one command to adjust its direction. When this task is executed, then the machine goes in the same direction (forward) as before without any further commands for movement. So that it just needs one command at the points of B, C, D, and E. However, when it reaches point F, it needs the ST command to stop.

As the required commands needed are listed in Table 2, this strategy needs six commands to get the task done. As a result, this strategy has reduced the number of commands and payloads. Consequently, fewer commands and payloads will lead to less bandwidth consumed.

Table 2. "Speed Control" commands with the parameters [11]

Command	**parameter**	**location**
NSMF	Speed v	A -> B
TR	Degree a1	B
		B -> C
TR	Degree a2	C
		C -> D
TL	Degree a3	D
		D -> E
TL	Degree a4	E
		E -> F
ST		F

During experiments, I also implemented a client application in Javascript following this strategy [35]. This version of the client application also falls into the category of coordinated teleoperation. Because the only command is sent to make the car move, and after that, it continuously keeps moving until another command is received by the car to turn or make it stop. So that there is autonomy added to the car to keep it moving, but it is not autonomous (the supervisory teleoperation), as it cannot reach the destination by itself. In other words, the user is not sending high commands e.g. the destination coordination, and the car needs the operator through the path. In this version of the application, when the key 'D' is pressed, the car starts moving with the default values

until the key 'S' is pressed and makes the car stop. While the car is moving, it is possible to speed up or down with 'up' and 'down' keys. Also, while the car is going forward or backward, the operator can increase or decrease the angle of the front wheels to the left and right with 'left' and 'right' keys on the keyboard. As it is noticeable this version follows the strategy "Speed Control".

2.5 A General Architecture for Teleoperation Solutions

Here I am introducing a general architecture that is a basic model (see Figure 13) and I compare it to the architecture of the provided practical solution is made. The model, or a variation of it, can be adopted by all types of vehicles teleoperation. When a solution is published, the underlying media for the communication is usually the internet. However, depending on the goal, the solution can be published under a local network - in both cases, the concepts are the same. The solution that I provide is following the same architecture, although the main differences is that instead of the internet browser shown in Figure 13, VR glasses (Meta Quest) are employed that host the user interface. Also, I based the solution in a private network instead of the internet.

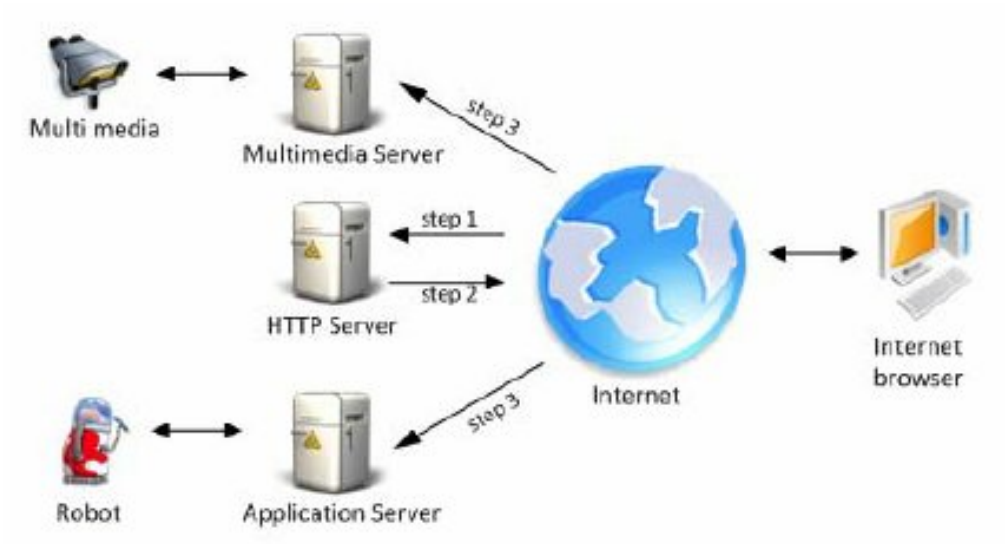

Figure 13. General Architecture [11]

As Figure 13 depicts, the first step is the request to the HTTP Server and the second step is the response from that server. The request (the first step) is sent by the client applications, which here is the internet browser. It means that the user interface is a web application and by this request, the web application is going to be fetched to the client side (the second step). In fact, with the second step, the server is serving the web application so that the user will interact through a web-based interface. As providing

web-based applications are popular, this allows the application to run from a number of devices such as mobile devices. However, in some solutions, the choice for user interface may be something else than web-based forms. Nevertheless, the user interface is a necessary element that needs to send the request to get initialized on the client side.

When the web application (the user interface) is initialized on the client side, then two connections (step 3) are established simultaneously. This means that there are two separate services available (the model shows that the services are on separate servers, while, as a variation, services can also be run separately on a single server). One connection is to the multimedia server to consume the video data. In fact, through this connection the camera images are served to the client application to be presented to the operator - this is considered as feedback from the remote teleoperator. Meanwhile, another connection is established to the application server. This channel is meant to send the commands to the teleoperator end. The provided solution in this thesis is following the same model basically, although there are differences.

Although the architecture of my solution is discussed in section 3.3, the first difference between the general architecture and the architecture of the provided solution is that the application server and the multimedia server are merged into one server. In fact, two services are running on different ports on the car. As a result, the car stays listening on different ports to serve the camera and receive the commands.

The second difference is that there is no HTTP server to fetch the client application. Because the client application in my solution is not a web application and it is developed based on Godot[36]. Actually, it is installed on the Meta Quest as an Android[37] application. This implies that the third difference is having the Meta Quest instead of the client's internet browser.

Godot is a cross-platform and an open source gaming platform. It comes with its own programming language and the final application can be published as an Android application. As a result, I get the client application running on the Meta Quest, which has Android as its own OS.

The final difference is that my solution is in a private wireless network. This means that the components in the solution are not connected via the internet, but through a WLAN.

2.6 Video Streaming

Almost in all the teleoperation solutions, the camera is involved and acts as a sensor to send feedback to the operator's side. So that the operator can see what the robot sees, but since the components are distributed in areas, a wireless connection is utilized to connect the components. This is where the bandwidth does matter. An important factor to consider is that video streaming consumes high bandwidth and as the data get larger (more quality) more bandwidth is needed. This is where video codecs and encoders come

into the place. As a result, more data with less bandwidth will be possible to send to the consumer side.

This challenge is not specific to Donkey Car teleoperation and that is why my solution helps with teleoperation in other areas. For instance, the challenge is also introduced in the area of Telerobotics or Telesurgery (Telesurgery is also an area with applications for Digital Twins, read section 2.9 for Digital Twins). The paper[38], has introduced a robotic arm (applicable to Telesurgery) with a camera installed. They use H.264 compression standard to encode the video data before streaming. On the other side then the video data is decoded and presented to the operator. This is, in fact, the general flow to transmit encoded video data (see Figure 14). There are several video codecs and Figure 15[12] shows the evolution.

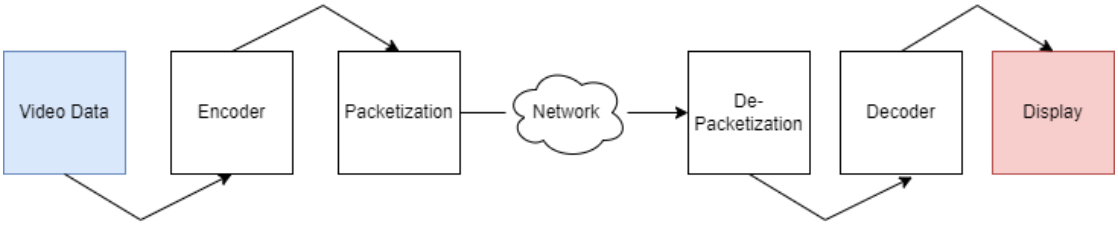

Figure 14. Video transmission flow over the network

2.6.1 H.264

The commonly used codec is H.264 compression standard. This codec is utilized for many cases such as TV broadcasting and live streaming. It is also popular in teleoperation - I already mentioned Voysys also takes the advantage of this codec. H.264 compression is also known as Advanced Video Coding (AVC), which was first published in 2003. It is built on top of earlier standards such as MPEG-2 and MPEG-4, which result in better quality and flexibility[39]. This added quality and flexibility affect the computation time. The codec H.264 supports multithreaded computation, but has some limitations with the performance because the multithreading was emerging at that time and it was not in their mind from the beginning.

2.6.2 H.265

The next generation standard H.265 overcomes the issues with H.264 and is considered as a successor to H.264. H.265 compression aim video resolution maximization by parallelism design[12]. The new compression standard (H.265) is 50 percent more efficient compared to H.264, which means a smaller data can be transmitted and less bandwidth is consumed, but it has not fully dominated H.264.

Development timeline(year)	video coding standard and its evolution
1990	H.261/ITU-T VCEG
1993	MPEG-1/ISO/IEC
1995	H.262/MPEG-2
2000	H.263/MPEG-4(part-2)
2003	H.264/AVC*JVT
2010	VP8 by Google
2013	VP9,H.265/HEVC*JCT-VC
2015	VP10
2018	AV1/AOM
2020	Versatile Video Coding (VVC)

Figure 15. Video codecs evolution [12]

Encoding the video data before transmission helps with reducing the size of the packets and makes it possible to reach better quality, but the cost of encoders is the CPU and GPU workload. Fortunately, the hardware video encoder of Raspberry Pi 4 can encode for H.264 video fast in real-time. In this regard, Chris Griffith has done a beanchmarking[40] and says: "The GPU hardware encoder in the Raspberry Pi can greatly speed up encoding for H.264 videos". He also explains: "It is perfect to use for transcoding live streams as well. It can be accessed in FFmpeg with the h264_omx encoder".

The solution that I provide is not taking the advantage of the mentioned encoders. While during the analysis of the problem I tested JPEG and PNG compression for video streaming, my solution streams the video as JPEG images. Applying the encoders to the solution can be considered an improvement for further study.

2.7 Network

Interconnection of components is an essential part of teleoperation and this has to happen over a networked system. The connection can be established over the internet, Cellular networks, or a wireless local network. A wired network is not suitable for teleoperation, as the components are distributed in a wide area mostly. In any way, there are factors and challenges in common that we need to take into consideration. For instance, the bandwidth and latency are significant, as data needs to be exchanged continuously in real-time. Here we discuss some factors.

2.7.1 Cellular Networks

As discussed in the paper[13], with the first generation of cellular networks (1G) it was possible to transmit voice up to 2.4 kbps, while the next generation (2G) made it possible to send short messages (SMS) and multimedia messages (MMS). 2G could transmit data up to 64 kbps and the bandwidth was between 30 kHz and 200 kHz. When 3G was introduced, the users we able to connect to the internet as they were moving around. With 3G it is possible to transmit data up to 2 Mbps with a bandwidth of 20 MHz. Since in teleoperation transmission of data in real-time is needed and in most cases, video data is being transmitted, these networks are mostly not sufficient. However, it needs to be considered that there may be still areas allowing only for 3G networks. Hence, the solutions should be optimized as much as possible. 4G and 5G networks are more appropriate.

Compared to the past generations, the 4G networks provide more smooth connectivity for multimedia applications, video conferencing, and voice calls. 4G date rate is up to 100 Mbps, but the bandwidth is similar to the 3G (20 MHz). Also, the latency with 4G networks is 100 ms and the coverage is 31 km.

Lastly, the 5G was introduced in 2020 and a jump was seen. The bandwidth and data rate increased to 1GHz and 10 Gbps respectively and additionally, the latency decreased to 1 ms. 5G allows 65,000 connections at the same time and provides a smooth real-time data exchange. Figure 16 depicts the significant parameters for cellular networks.

Parameters	2G	3G	4G	5G
Latency	500–1000 ms	200 ms	100 ms	<1 ms
Bandwidth	200 KHz	20 MHz	20 MHZ	1 GHz
Data Rates	64 Kbps	2 Mbps	1 Gbps	10 Gbps
Coverage Range	1–10 km	1–10 km	31 km	200–300 m
Security	64-bit A5	128-but KASUMI Cipher	128-bit AES, 168-bit DES	D2D and continuous authentication
Scalable Capacity	-	10 bits/s/Hz/km^2	50–500 bits/s/Hz/km^2	50,000 bits/s/Hz/km^2
Network Capacity	-	-	1000 nodes	65,000 nodes

Figure 16. Cellular networks parameters [13]

2.7.2 Transport Protocols

When it comes to the networked systems and data transmission, the two protocols TCP and UDP are heavily used. However, there are factors to focus on before choosing one protocol. TCP and UDP both are staying in the Transport layer, but there are noticeable differences.

TCP. The TCP protocol is a stateful (connection-oriented) protocol. TCP establishes a bidirectional channel to send and receive a packet and once the packet is sent the connection is closed. This helps TCP to acknowledge if the packet is received at the destination. If it is not received by the destination, then TCP will retransmit it again. In other words, TCP comes with automatic error handling and guarantees that the packets are received by the other end device. TCP protocol also delivers the packets in order, meaning that as long as the packet is not received at the destination, the next packets will be blocked; however, a timeout can be defined so that if the packet is not received during that period it will be ignored. This makes the use of the TCP protocol for real-time applications problematic, while it is reliable. This is even worse in wireless networks because of a mechanism in TCP for packet losses. Since in wired links packet losses are usually caused by congestion because of their low loss nature, TCP reduces its transmission window size resulting in network utilization. Although, losses in wireless networks are caused by other factors than congestion such as transmission medium, but TCP action to packet losses is still the same congestion mechanism [41].

UDP. Compared to TCP, UDP is stateless. This means that it will not know if the packets are received by the destination and will not block the next packets. This is the benefit of UDP that the next packets will be sent in any way. This is beneficial for real-time applications, while it results in the other end devices not receiving the packets in order. These attributes make UDP faster than TCP, however TCP is more reliable as UDP does not guarantee to deliver the packets and will not retransmit them.

Enhanced UDP. The applications are not limited to choosing either TCP or UDP and the effort is sometimes to build a particular protocol with specific characteristics on top of other protocols. Enhanced UDP (E-UDP) is, in fact, an extension on top of UDP to tackle some issues with UDP. For instance, time stamping and a sequence number supported in E-UDP help destination devices to be aware of missing packets and apply their recovery strategies. Also, E-UDP comes with a "heartbeat", which is a signal to the destination periodically to find if the remote device is still alive. It also provides a mechanism to adapt the bandwidth [42].

High-Low level command. After reviewing the protocols, it is also needed to know the levels of commands in teleoperation to proceed with one protocol. There are two levels of commands in teleoperation: high-level commands and low-level commands[43].

By high-level commands, the operator is defining the ultimate goal related to the task and will not guide the remote machine toward that goal. For instance, the operator sends coordination then the teleoperator takes it as a destination to reach. Picking up or putting down an item, drawing a circle on a paper or closing the window, and more, are examples of these kinds of commands that the operator sends. Then it is the remote machine that

knows how to do that autonomously. These commands are highly used in supervisory teleoperation, as there is a high level of autonomy in this classification. The high-level commands need to be guaranteed that they are received by the teleoperator and they are also not frequently sent. Therefore, the TCP protocol is an appropriate choice to send these commands.

Besides, the low-level commands are to move the teleoperator to get the task done. The operator is moving the teleoperator for that specific task. For instance, if the robot machine wants to reach a destination, the operator should move the teleoperator forward or backward and be careful of obstacles on the way to give the vehicle a direction. In other words, the operator directs the teleoperator appropriately through the path to the destination, as there is no autonomy and the task cannot be done autonomously. The commands of this type are sent frequently and higher bandwidth is needed for low-level commands. If some low-level commands are lost, the machine can continue working, however, the performance is degraded. Thus, For low-level commands, the UDP protocol can be taken into account.

My solution mainly benefits from the low-level commands to teleoperate the Donkey Car. This means that as long as operators pushes the joysticks buttons, the control data is sent to the Donkey Car and it will move. Once they stop pushing the button, then the car will stop. As the delay minimization was out of scope for my solution, for both control data and video data transmission I take the use of WebSockets[44] (WebSockets are based on TCP) to guarantee that the video data is received by the client and the video quality is preserved. To reduce the delay, using the UDP protocol can be taken into consideration. With further investigation, it is also possible to come up with a hybrid protocol or a UDP-based protocol like Enhanced UDP mentioned earlier.

2.7.3 QoS

The Quality of Service (QoS) is a technology that makes it possible to define prioritization. In fact, depending on the task, sometimes some specific services or traffic have more priority than others. For instance, in some cases, Audio is prioritized over Video, and in some cases vice versa. By using QoS and defining the priorities we can manage the traffic and the bandwidth efficiently. the ITU-T defines QoS as "the collective effect of service performance which determine the degree of satisfaction of the user of the service".

In teleoperation, depending on the task, maybe sometimes the video data is more important than the control data or other types of feedback such as auditory or force feedback; Contrarily, maybe sometimes other types of data are prioritized over the video data. In any case, QoS is an option and a technology provided to manage the traffic and use the bandwidth efficiently. QoS may be more useful when the traffic goes out of the private network. However, in local networks, considering applications running locally

and dealing with different types of traffic such as audio and video, still QoS helps with the bandwidth. Although, some people argue that in local networks it is easier to add more bandwidth.

Applying the QoS appropriately requires precise analysis and QoS has classifications helping with the analysis, but going deep into this topic is out of the scope of the thesis.

2.8 Teleoperation in Virtual Reality

With virtual reality (VR), the human is allowed to virtually enter a remote environment, which is known as a virtual world. The focus of VR, besides receiving feedback, is on the human‘s interaction with a virtual world and this makes teleoperation more challenging. In VR-based teleoperation, the operator, while has worn a virtual reality headset, interacts with 3D objects in VR space. Virtual reality headsets are, in fact, a type of head-mounted display (HMD) presenting the interface, through which the operator interacts with the virtual world. The VR setup allows the operator feels in the remote environment by receiving not only the visual feedback, but also the haptic feedback. From Precision Microdrives[45]: "Haptic/Tactile feedback (or haptics) is the use of advanced vibration patterns and waveforms to convey information to a user or operator". Compared to the simple camera-monitor combination in conventional teleoperation, VR creates a high level of presence and makes an opportunity for having a higher level of telepresence.

2.8.1 General Architecture

As Figure 17 shows, the general architecture for VR-based applications is divided into two major components - the remote and local components, which are connected via a network. The network can be established on the internet, local networks, or other technologies like Bluetooth[14].

The remote component locates in the real world and contains minor components. This component is responsible to live in the real world and the sensors are important parts of it to send the feedback to the operator. As Figure 17 depicts, for instance, the remote component can be equipped with the Position, Force, Vision, and more sensors. The camera is also considered as a sensor to collect the data from the environment (camera images) to send as feedback. Behind the sensors, there is a module to pass through the received commands from the operator to the teleoperator. This module sitting in the remote component is referred to as the Drive Unit.

The local component is divided into VR I/O Devices and VR Engine. VR I/O devices are means to make the operator capable of interacting with the real world. With these devices, the operator sends the command to the remote component (Drive Unit) and receives the feedback transmitted via the sensors. These are making a presence of being in the real world for the operator. Through these devices, the operator can receive visual

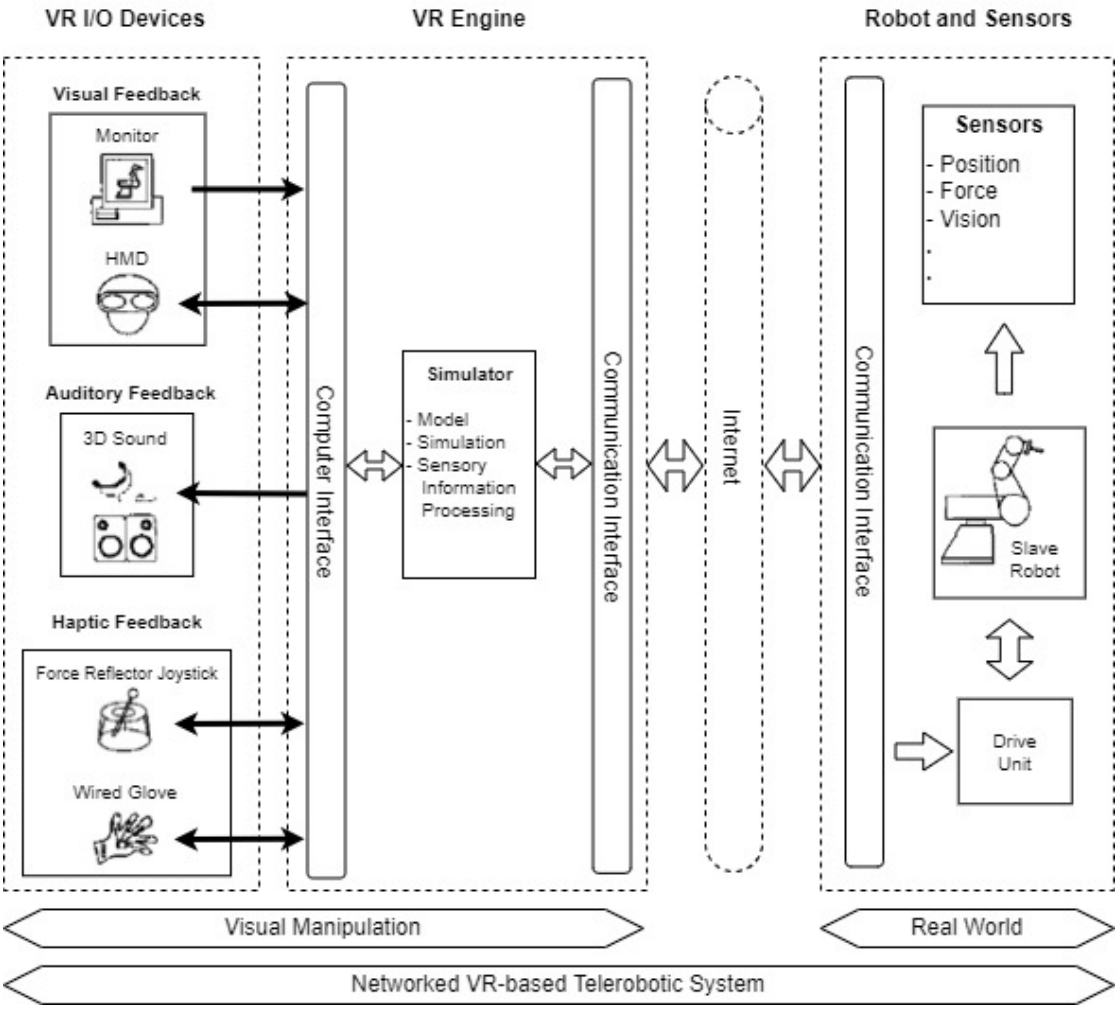

Figure 17. General diagram of a VR based teleoperation [14]

(Monitor and HMD), auditory (3D Sound), and haptic (Force Reflector Joystick and Wired Glove) feedback.

The VR engine also resides locally. This engine can be considered a manager and is responsible to receive the commands, translate, and send them to the remote components. It is also responsible of receiving the feedback data, translating them again, and providing it to the I/O devices. The VR engine also is responsible to build virtual environments and rendering the scenes and 3D Objects.

Although I explained the major components of VR-based teleoperation and described how they are connected, there might be some differences in the components utilized and how they are networked in some of the solutions. For instance, in this paper[46] they have developed a framework for VR-based solutions to interact with the real world. Compared

to the general architecture, they have used an Arduino Uno as a remote component that is connected to a VR I/O device (HMD), which supports Android. In their solution, the remote component is connected to the local component (HMD) via Bluetooth. Considering the VR engine, they take the use of Unity3D[47] running on Android. Unity3D is a popular Game Engine that helps with building virtual environments and rendering 3D scenes and objects.

The Arduino stays listening to receive a message from the potential modules that can be connected to it like a gyroscope. Once the message is received, it is sent to the Unity3D to translate so that by wearing, for instance, an HMD, the feedback received to the VR engine, is represented to the operator. As a result, the operator can see the changes in the real environment and feel like being in the real world. This means that their framework is acting as a bridge for the peripheral modules that can be plugged into both sides (Arduino and VR engine) of the framework. Then the feedback from the remote end will go through their bridge and is represented to VR I/O devices.

In their experiment, they connected a gyroscope to the Arduino and an HMD to the VR engine. Then while the operator had worn the HMD, he had the Arduino on his leg tied. Thus, when he was walking in the virtual world, he was given the presence of walking in the real world.

In my practical solution, Donkey Car is considered a remote component. As the car comes with Raspberry Pi, this is considered a drive unit. It is responsible to receive the request, process, and making the car move. Raspberry Pi also acts as an interface to send the feedback to the local component.

In regard to the local component, the VR glasses and the joysticks used in the solution are, together, considered as the local component. The VR glasses and the joysticks are the VR I/O devices and this is while the VR glasses have Android as their OS. Android is a platform for the VR engine (Godot). Godot is employed in the solution as the VR engine to build the environment, through which the operator receives feedback and interacts with the remote component.

Considering the connectivity, my solution networks the components locally. There is a WLAN set up, to which the components are connected.

2.8.2 The Challenges in VR

Compared to the conventional teleoperation approaches, teleoperation in VR is more challenging. While the challenges already reviewed are still held, some more are added by this variation. Here we are explaining some of them[14, 48].

Telepresence. The telepresence of the real world is one of the major challenges. This feature provides an opportunity for the operator a higher degree of presence in the real world. Because together with the visual sense it includes the senses of hearing and touch. For telepresence of the real world, not only the virtual environment should be

built similar to the real environment, but also there should be a mechanism implemented to catch, transmit, process and represent the haptic and auditory senses as well as visual sense[14].

The solution that I provide is not directed to represent the haptic or auditory feedback, although it is taking the advantage of VR glasses. VR glasses are representing the built 3D-based environment and video data so that the solution provides a high level of telepresence, but the area of telepresence can be improved. In other words, regarding telepresence, the operators will find themselves inside a virtual 3D model and see what the car sees so that they can feel shrunk in the car, but the car can be equipped with some sensors to send the auditory and haptic feedback to be represented to the operators. As a result, the operator will also hear the environment and the joysticks will vibrate when the car runs into the obstacles on its way. This can be defined as an exciting improvement to be considered for future work.

VR I/O Devices. There are devices in the market that help with representing the feedback to the operator and the user interaction, but they are costly. Also, the integration of these devices is another issue. In other words, the link between these devices and the system together with the data transformation in real-time should be taken into consideration.

In this regard, my solution takes the benefit of the joysticks that come with VR glasses. As a result, the input from the joysticks reaches to the car.

3D Environment Modeling. The process of building and rendering the 3D models is taking a long time. There are applications helping with the process such as AutoCAD[49] and Blender[50], but depending on the application, this process needs a long time and expertise. It also needs powerful PCs for rendering 3D models. During the experiments, I built three environments that are being presented in the section 3.4.2.

Time Delay. Each component adds some delay to the solution. For instance, when VR I/O devices such as joysticks and wired gloves are integrated, they need to send and receive data over the network in real-time. Also, the data needs to be translated. Therefore each component adds a delay to the solution to some extent and it is inevitable.

The VR engine is also the component that adds delay. The efficiency of the VR engine is essential, as it is dealing with rendering 3D models, together with receiving, processing, and transferring data to connect the components together.

My solution makes use of Godot. It is a great gaming platform supporting 2D, 3D, and VR. Godot is a powerful engine and it is a respected rival for Unity3D and Unreal[51]. In section 3.2.3 I explain why I chose Godot.

2.9 Digital Twins

Here I am introducing Digital Twins (DT), as it can be investigated more as a form of teleoperation. The challenges and concepts for teleoperation that I mentioned earlier are still true in the context of DT, as they have characteristics in common.

The DT concept first was introduced informally by Michael Grieves in 2002 and NASA created the "Digital Twin" term in 2010 and explained it as "a multi-physics, multi-scale simulation of an asset, incorporating high-fidelity modeling and simulation and situational awareness in real-time"[52]. Saddik in his paper explains "Originally developed to improve manufacturing processes, digital twins are being redefined as digital replications of living as well as nonliving entities that enable data to be seamlessly transmitted between the physical and virtual worlds. Digital twins facilitate the means to monitor, understand, and optimize the functions of all physical entities and for humans provide continuous feedback to improve quality of life and well-being."[53]. DT is, actually, a multidisciplinary approach for representing a physical object, system, or process in the digital world, which is utilized for further enhancement, design improvement, and diagnosis purposes. In other words, the advances in Artificial Intelligence (AI), the spread of broadband, computational power, embedded sensors, Internet of Things (IoT), Big Data processing, Machine Learning (ML), and cloud computing in the past decade have led to the emergence of DTs[54]. DTs are highly used in industry and some use cases are listed in Figure 18 [15].

2.9.1 Components

As Figure 19 depicts, DTs consist of three main components: Physical Entity, Digital Representation (DT), and the link between them[15]. Compared to teleoperation, these elements are aligned with the main components of teleoperation discussed earlier.

Physical Entity. Compared to teleoperation, this element resides in the real world and can be any object, process, or system. For instance, the physical entity can be defined as a remote machine, which is equipped with sensors, actuators, and a drive unit. So that the data, from the environment, is collected on this end and represents the state of the remote machine in the physical world. Afterward, the data is sent through the link.

Link. Similar to teleoperation, a link, through which the data is exchanged between the physical entities and DTs is an inevitable element. One option to establish the connection is through the internet, while there are other options available such as 4G and 5G networks. The network challenges in regard to teleoperation are still valid here since there is still a need to exchange data in real-time. In addition, depending on the application, there may be several DTs interacting with each other so there should be a connection established between them as well.

Industry	Applications
Manufacturing	• Design verification • Predictive maintenance • Process optimization • Safety management • Equipment utilization, etc.
Energy	• Power monitoring and management • Failure analysis • Grid operation and maintenance, etc.
Smart cities	• Transportation monitoring • Urban planning • Strategy evaluation • HVAC (heating, ventilation and air conditioning) control, etc.
Farming	• Planting optimization and monitoring, • Machine and products tracking • Pesticide monitoring, etc.
Building	• Progress monitoring • Budget control and adjustment • Building quality assessment • Worker safety monitoring • Resource allocation and waste tracking, etc.
Healthcare	• Health monitoring • Personalized medicine • Medical resource allocation, etc.

Figure 18. Digital Twins applications in the industry [15]

Digital Representation. DTs are built based on the received (dynamic) data and static data. The dynamic data is collected and received from the physical world, which is updated in real-time so that the digital counterparts of the physical objects will continuously represent the current state of the objects in the real world. Static data is related to the definition of the physical objects, which is needed to build the DTs. Also, historical data, which is the result of performed actions falls into the category of static data. Based on these two categories of data, the DTs are built, decisions are made, and will be sent back to the physical objects. In fact, the DTs are continuously updated during the physical counterpart lifecycle and will adjust the physical entity in real-time.

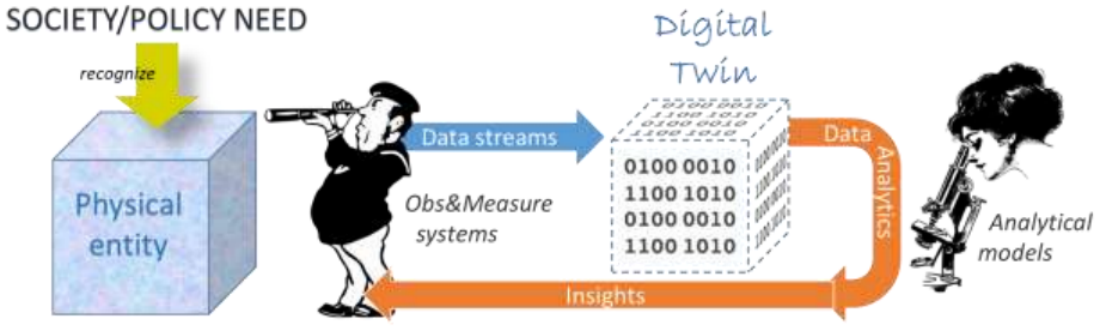

Figure 19. Digital Twins Components and approach [15]

2.9.2 Conceptual Model

Here I am introducing a conceptual model for the integration of physical entities and DTs[13]. The model can be considered as a base model, but depending on the applications it may be adjusted. As Figure 20 shows, the model consists of layers: "Physical Space", "Communication Network", "Virtual Space", "Data Analytics and Visualization", and "Application".

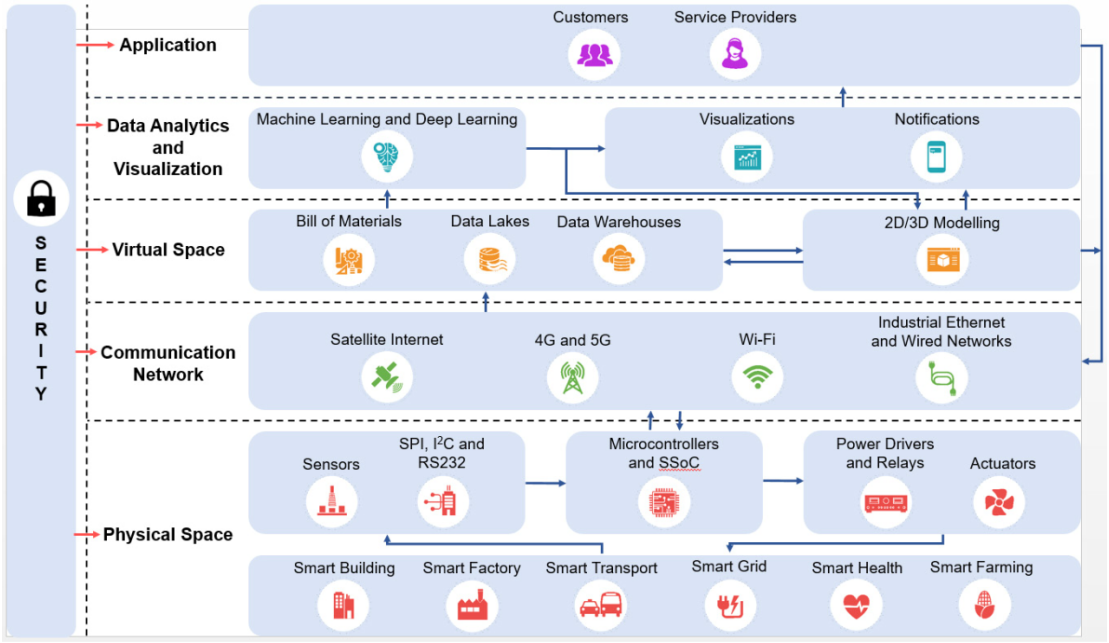

Figure 20. Digital Twins Conceptual Model [13]

Physical Space. In this layer, as Figure 20 represents, physical objects together with sensors and actuators (the remote component) exist. The sensors will collect the parameters from the environment and then they will be sent to the upper layer.

Communication Network. This layer is the second layer, which is on top of the physical layer. The purpose of this layer is to receive the data from the bottom layer (physical layer) and transmit it to the upper level (the virtual space) efficiently. This layer is considered a bridge between the physical and virtual space. Here the connection can be established, as discussed earlier, through the internet, cellular networks, and LAN. Considering the advancement in cloud technology, one option can be based on the cloud, as the virtual space is not necessarily in the same location as the physical space.

Virtual Space. The purpose of this layer is to process and store the data together with modeling. The data, which is provided by the sensors and the communication layer, is processed in this layer and stored for further use. Since a lot of data is received in this stage, some data reduction techniques may be applied such as encoding/decoding techniques, aggregation techniques e.g. MapReduce, or data fusion algorithms. Afterward, based on the available data, dynamic 3d models (digital twins) are created. In fact, sensors continuously send the information, and data is updated in this stage. Then, the model is dynamically built and updated so that the changes in the physical environment and physical entities reflect the 3D models.

Data Analytics and Visualization Layer. After data preparation in the bottom layer (the Virtual Space layer), this layer can access data for analysis purposes. In this layer data analytics together with machine learning and deep learning algorithms can be employed, and afterward, the result can reflect on the 3D models or be represented through dashboards to management. As a result, the data analytics in this layer will provide a dynamic 3D model representing the current state of the physical entities through which it is possible to predict possible failures, maintenance requirements, and design improvements.

Application. Since a large amount of data is transmitted and processed, it can help with the improvement and enhancement of the physical entity, prediction of failures, design improvement, and many other applications in industry or other domains. For instance, in regard to teleoperation, we can name autonomous driving.

The solution that I provide is not related to DTs directly, but as explained in this section DTs can be considered as a form of teleoperation. For instance, teleoperation needed for autonomous driving can be an application for DTs. Autonomous driving in any way needs teleoperation because there will be problematic or difficult moments when the operators need to take control of the machine. Also, in some countries, it is

a part of regulation to have an operator ready as a backup. Autonomous driving falls into the category of supervisory teleoperation since the remote machine will receive a high command e.g. the destination coordination that it needs to reach. This means that the remote machine with the use of sensors and path-finding algorithms and techniques will travel through the path autonomously by itself. Considering the DTs model, the autonomous vehicle, can be considered as a physical entity, which its sensors collect data from the environment as the vehicle moves. Then, the data goes through the layers and will be processed until the digital twin of the vehicle is made and updated continuously. Complex big data analytics and ML processing algorithms are conducted and as a result, the operator not only can assess the current state of the vehicle, but also the statistics provided by dashboards, charts, and graphs for further decisions. Also during this procedure, some decisions can be made automatically to be sent back to the vehicle[13].

Another example is the teleoperation of robotic arms which is applicable in Telesurgery. In the paper[55], a framework to control a remote robot with DTs in VR has been introduced. They particularly have examined their framework with an operator remotely controlling a robotic arm.

3 Practical Solution for Toy Cars

In this section, I introduce a solution to teleoperate a Donkey Car S1 via the joysticks, which is based on VR glasses (Meta Quest 2). The solution can be adjusted for different kinds of remote machines and VR glasses. First, I explain the hardware components available for the solution, since they are dictating the architecture to some extent. Then, after analyzing the solution, I describe the architecture and the software programs.

3.1 Hardware Components

Here I am describing the existing hardware components for the solution.

3.1.1 The Donkey Car S1

Donkey Car S1 (see Figure 21) is equipped with a Raspberry Pi 4[56], and a camera (Wide Angle Raspberry Pi Camera). It is manufactured by Robocar ltd in Hong Kong and they explain: "It is one of the most popular self-driving projects used by academics, technology companies and research institutions. It is made for you to learn AI through a hands-on and engaging experience. Start learning, along with the comprehensive education package we offer!"[16]. The car also makes several software programs available. For instance, an open source software program in Python that provides basic services together with a web-based interface to communicate with the car and a mobile application to get the car teleoperated fast.

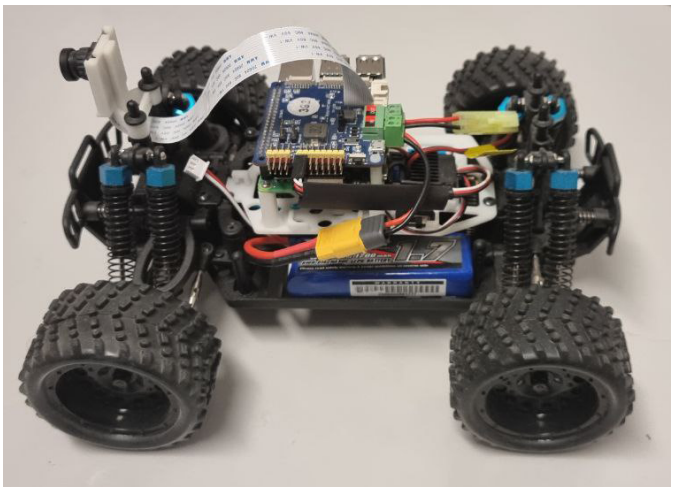

Figure 21. My Donkey Car S1

Raspberry Pi 4. Raspberry Pi (see Figure 22) installed on Donkey Car comes with a Quad-core CPU "Cortex-A72" (armv71) 32-bit SoC @ 1.5 GHz (see Figure 23) and 2 GB of RAM. The memory allocation for the CPU and GPU is also shown in Figure 24.

Figure 22. Raspberry Pi 4[16]

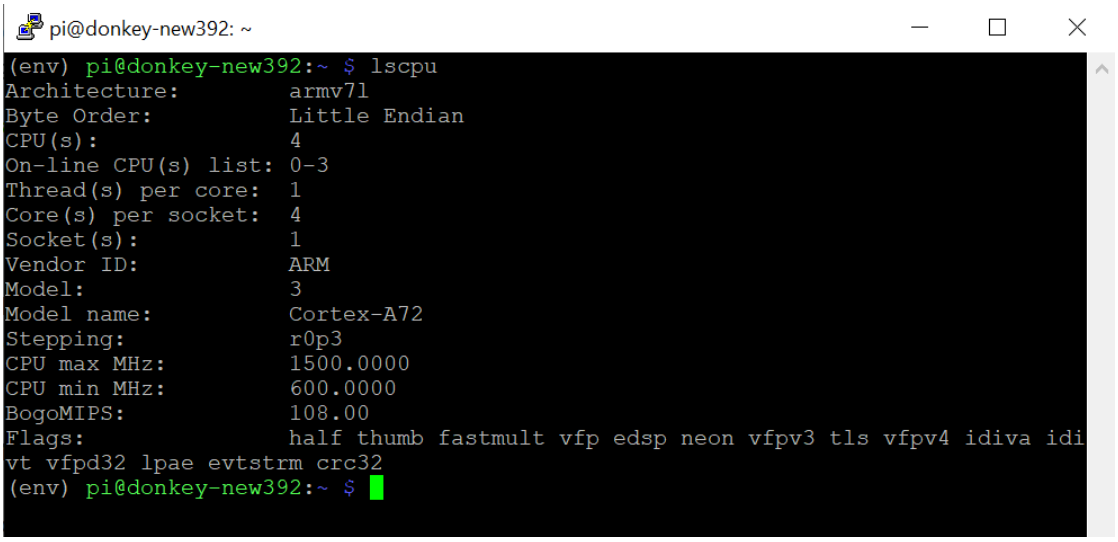

Figure 23. Raspberry Pi 4 - CPU

```
pi@donkey-new392: ~
(env) pi@donkey-new392:~ $ vcgencmd get_mem arm && vcgencmd get_mem gpu
arm=896M
gpu=128M
(env) pi@donkey-new392:~ $
```

Figure 24. Raspberry Pi 4 - CPU and GPU memory allocation

Also, Raspberry Pi is connected to the Robo HAT MM1 from Robotics Masters[57]. This is actually a controller board that acts as a PWM (Pulse Width Modulation) driver to supply the power and vary the speed. Pico Technology[58] explains: "PWM (Pulse Width Modulation) is an efficient way to vary the speed and power of electric DC motors". Figure 25 shows the Robo HAT MM1.

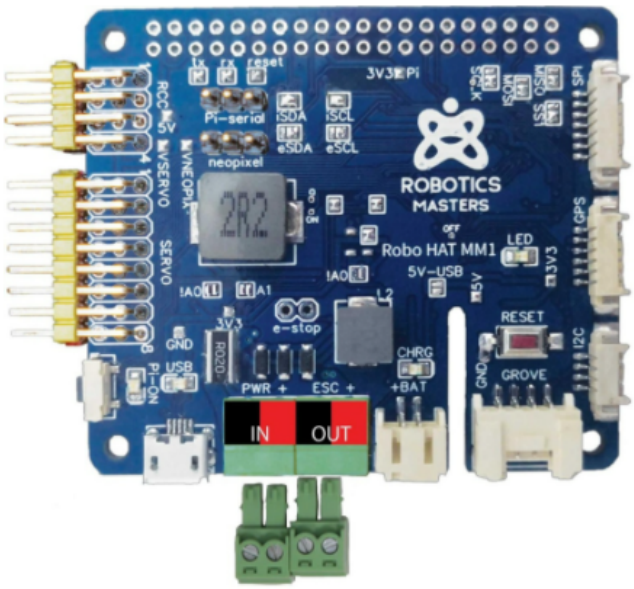

Figure 25. Robo HAT MM1 [17]

3.1.2 Meta Quest

The solution takes the benefits of VR glasses (Meta Quest) (see Figure 26). Meta Quest is considered a local component in the solution and is a perfect suite of innovations providing a superb opportunity to build virtual environments. It has Android as its OS and the processor (Qualcomm® Snapdragon™ XR2 Platform) offers high AI capability. It also comes with 6GB of RAM and the display features 1832 x 1920 pixels per eye. The display is also capable of supporting 90 Hz[18].

Figure 26. Meta Quest [18]

3.1.3 TD-W8960N

The components of the solution (the local and remote components) are connected via a WLAN. To build a WLAN, the router TD-W8960N from tp-link is employed (see Figure 27). This router supports wireless standards IEEE 802.11b, 802.11g, 802.11n, a speed of up to 300 Mbps, a frequency of 2.400-2.4835 GHz, and QoS [19]. This router has been used because routers with a frequency of 5 GHz sacrifice the coverage, although they provide a faster speed. Choosing a 2.4 GHz router allows driving the car in a wider range. The article [59] states: "Wi-Fi routers operating on the 2.4 GHz band can reach up to 150 feet indoors and 300 feet outdoors". It also explains that the 5 GHz routers can reach up to one-third of these distances.

3.2 Analysis

Here I explain how the problem is analyzed to come up with the architecture in the following section, although the introduced hardware components dictate the architecture to some extent.

3.2.1 Communication Link

To choose an appropriate communication link, the requirements and considerations should be explained (RQ4). As the operator needs to connect to the remote machine to consume the services, the communication link is significant. Since there is a client to connect to a server (remote machine) for its services, the client-server communication and architecture are noticeable here. Also, it is prominent that real-time communication

Figure 27. TD-W8960N from tp-link [19]

is happening and the delay should be as less as possible. There are several options available and tested for the communication link, which is explained as follows.

The first option I tested is making hotspots with Donkey Car S1 or the Personal Computer (PC). Donkey Car S1 tries to connect to the configured network when it is switched on. If it cannot connect, then it makes a hotspot as a way for other devices to connect to the car. Besides, it is possible to make a hotspot via PC and configure the car to connect to it. With the experiments I did, I experienced that either Donkey Car hotspot or the hotspot via the PC are beneficial options only for short distances and to get started. The article [60] also states that the hotspot signals can be reached up to 33 feet. Thus, a hotspot is not a good option as the coverage is strictly limited.

The second option for the communication link is based on 4G or 5G technologies and through the internet. This option allows the car to go far away, but it is more complicated and costly. Truly with internet-based communication, the bandwidth should be taken into consideration more and several extra factors may affect it such as the mobile provider. Also considering Donkey Car comes with Raspberry Pi, Raspberry Pi needs a component to allow a sim card to be inserted. As the goal of the thesis is to provide a minimum viable product, this option can be defined for further study.

Lastly, the more appropriate option is to connect the components via a WLAN. Because with this option it is possible to simplify the problem to achieve the goal of the thesis. Besides, this option, as tested, is more reliable than the hotspots mentioned and the car can travel longer distances. As a result, the car is configured to connect to the provided WLAN using a router (TD-W8960N) from tp-link.

3.2.2 Remote Services

Remote services running on the car and how they are implemented is another topic to analyze. Donkey Car comes with an open source software program in Python and provides fundamental services needed. The open source software provides a service through the TCP protocol to teleoperate the car and I take the benefit of this service. Besides, there is another service provided by the open source software to consume the camera over the HTTP protocol. In general, these two services are enough to achieve the goal of the thesis, but to be more flexible and for learning purposes, I implemented a service for camera consumption. Similar to the camera service provided by the open source software, my service is based on TCP and the lost packages are retransmitted, but my service is implemented from scratch and is more flexible. It allows adjusting the resolution and the quality of the camera images together with the frame rates. Also, this service paves the way in the future for applying a preferred encoder more easily, as the open source software lacks the documentation so more investigation would be inevitable (I spent many hours reading the open source software to find out how to use the camera service, set the parameters, and how and in which format the message should be structured). I did not build the service over UDP because it is stateless and it is not retransmitting the lost packages. As a result, the quality may be degraded.

Regarding the compression and encoders, my service transfers camera images in JPEG compression. The service benefits from the Picamera[61] package and it internally supports compression e.g. JPEG, PNG, and raw formats. I also tested PNG compression, but as PNG uses lossless compression[62], the camera images have a size of around 90 KB. With the same resolution (320X240), JPEG compression produces camera images with a size of around 40 KB, even though it is a lossy compression[62].

I am not taking the advantage of encoders, while the Picamera package supports h.264 streaming. The thought behind the implementation of this service was to stream frame by frame, not real video streaming. By migrating the approach to real video streaming, it is possible to apply streaming in h.264[63] as the Picamera package supports it, but it needs more investigation to be applied to the server and client applications. The server side implementation can be done more easily, compared to the Godot side (the client application), because of the Picamera support. Currently, the service does not take the advantage of encoders and it can be defined for further study, while with encoders, it is possible to reach a higher quality. However, it should be tested how much the encoder will add workload, as the cost of encoders is CPU and GPU.

Lastly, since there is only one Raspberry Pi available on the car (only one platform is accessible), it is inevitable to share the hardware for both services and run two services on different ports on the same OS (platform).

3.2.3 Client Application and Telepresence

The client application and Telepresence are other topics to analyze. Operators need to be provided with an application so that they can teleoperate the car remotely. Besides, a higher level of telepresence makes the operators experience being involved in the solution.

The client application is based on Godot and installed on Meta Quest. Godot is a gaming platform and is a worthy rival for other famous gaming engines such as Unity and Unreal. Godot supports 2D and 3D, is cross-platform, runs fast and is lightweight (this makes it a better fit for not powerful PCs), and has Android support. Godot also can be integrated with Microsoft IDEs (integrated development environments) such as Visual Studio and Visual Studio Code. Moreover, while it has its own programming language (GDScript), it supports C# and C++. Godot has great support for request-response communication over TCP and HTTP via built-in objects. Godot supports VR and the applications can be exported as android packages (APK) so that they can be installed on Meta Quest. Godot is also open source and this makes the work done in this thesis open to other students.

With the help of Meta Quest, a great level of telepresence is achievable. Meta Quest makes the operator immerse into the display and its combination with Godot provides the potential for a high level of telepresence. However, building virtual environments that nicely fit the needs is a challenge and needs its own expertise to come up with flawless 3D models. A higher level of telepresence is achievable by sending haptic or auditory with the use of sensors from the car to be represented to the operator. This is related to the immersion intent that makes the operator feels like being in the environment. The book[64] explains: "Immersion is the objective technology that has the potential to engage users in the experience. However, immersion is only part of the VR experience as it takes a human to perceive and interpret the presented stimuli. Immersion can lead the mind but cannot control the mind. How the user subjectively experiences the immersion is known as presence.".

3.3 Architecture

Here I describe the architecture of the solution (Figure 28 represents an overview of the architecture).

As Figure 28 shows, the remote and local components are connected through a WLAN via the router TD-W8960N from tp-link.

On one hand, there is Donkey Car S1 as the remote component. The car hosts two services running on different ports. One service is running on port 8887 for teleoperation. The responsibility of this service is to receive the command and make the car move. Another service (custom camera service) that is responsible to transmit the video data is running on port 8080. These two services are described in section 3.4.1.

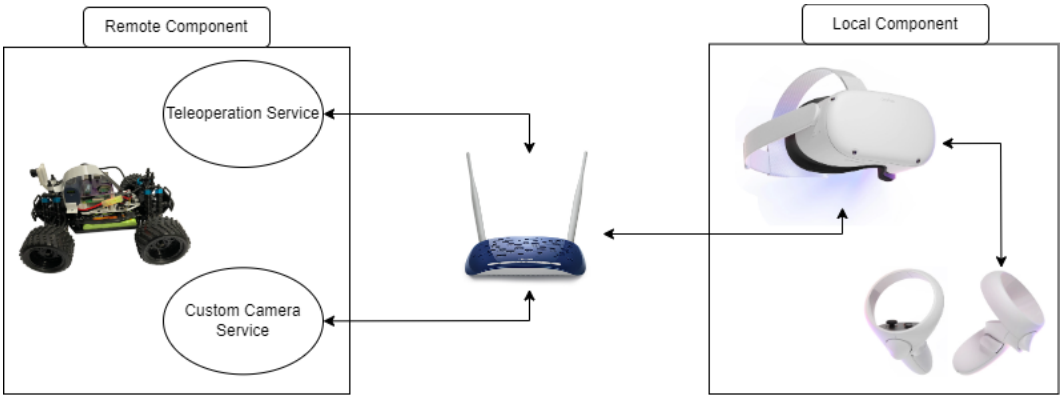

Figure 28. Overview of the architecture

On the other hand, there is Meta Quest as the local component. Meta Quest hosts the client application, which connects the operator to the remote car. As Android is the OS of the Meta Quest, an Android-based application has been written with the use of Godot. This client application is introduced in section 3.4.2. Meta Quest communicates with the custom camera service and renders the video data received so that the operator receives the feedback (video data) in the VR glasses. Also, Meta Quest communicates with the joysticks internally to receive the commands, since the engine is located in the VR glasses. Afterward, the commands from the Meta Quest are sent to the teleoperation service on the car to make it move.

3.4 Software Programs

The solution involves two applications: remote application and client application, which I explain in this section.

3.4.1 Remote Application

The remote application itself contains two services running on the remote car. Before diving into the remote application, it is necessary to introduce the OS of Raspberry Pi.

Raspberry Pi Operating System. Raspberry Pi has a Debian-based OS. Debian is a Linux distribution and the OS of Raspberry Pi (previously called Raspbian)[65] is based on that. Raspberry Pi OS has been optimized for Raspberry Pi hardware and it is the recommended OS for normal use (see Figure 29). This gives a perfect opportunity to go with any programming language we prefer. Also, it allows connecting to the remote machine via SSH. SSH connections allow configuring Donkey Car to connect to the provided WLAN, executing the services, and many more.

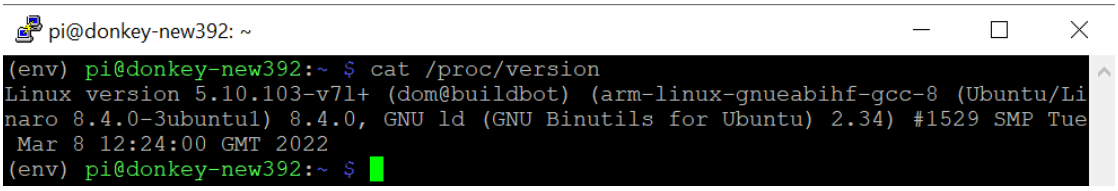

Figure 29. Raspberry Pi 4 - OS

The Donkey Car S1 Open Source Software. Donkey Car comes with an open source software[66] written in Python that I take the benefit of it to run one of the services needed on the remote car (teleoperation service). The open source software is quite mature and it provides web-based API services needed for camera consumption and teleoperation of the car, while the downside is the lack of documentation. To consume the API services of the open source software, when Donkey Car is configured properly and connected to the established WLAN, a Python file needs to get executed via SSH connection. Once the file gets executed, then a service on the car runs and stays listening. Afterward, camera and teleoperation services are exposed and consumable.

The camera API serves the video data (images) continuously through an HTTP Connection. During my experiments, I implemented a Javascript application (available in Github[67]) and consumed the camera API to present the video data, but for the final solution, I came up with a new service written from scratch for camera consumption. During my experiments, I also modified the open source software to add a feature to serve the images not only in HTTP, but also through WebSocket connections (the customized open source software is available in Github[68]). As the last modification to the open source software, it was necessary to make it release the camera because I wrote a separate service to serve the camera. Hence, I added a switch to the open source software allowing it to run the software while the camera is bypassed and not used. This allows other services, which are running at the same time, to consume the camera. Figure 30 shows the execution of the customized open source software and the usage of the switch bypassing the camera.

Behind the camera API, the open source software provides another API for teleoperation. This API, through the WebSocket connections, allows making the car move (RQ5). The API mainly accepts two parameters: "angle" and "throttle". Each of these parameters can be between -1 and 1. If the value for the property "angle" goes toward -1, then the front wheels are set to drive the car toward the left and if it goes toward 1, then the front wheels are set to go toward the right. Considering the property "throttle", it is mainly the speed. The value 1 indicates the maximum speed forward, the value -1 indicates the maximum speed backward and the zero makes the car stop. These values tell that if the car receives a command to go forward (e.g. throttle = 0.5) or backward (e.g. throttle = -0.5) it will go in that direction by itself until it receives a command in that the property "throttle" is zero to make the car stop. This is considered as autonomy

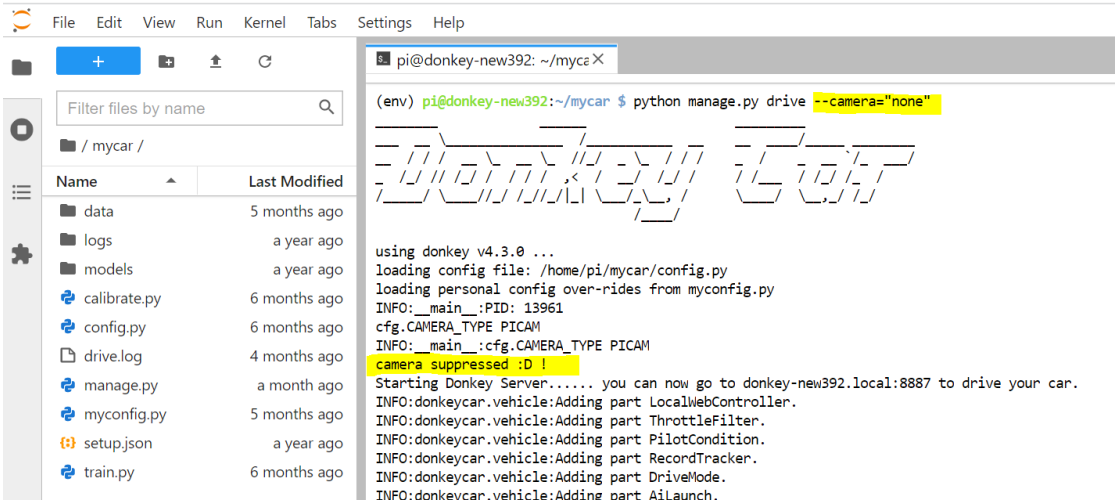

Figure 30. The execution of the customized open source software and the switch bypassing the camera

added to the car that classifies the solution into the category of coordinated teleoperation.

To sum up, my solution utilizes the built-in open source software, as one of the running services on Donkey Car is the teleoperation service. For this purpose, the customized built-in open source software gets executed without camera consumption.

The Custom Camera Service. Together with the teleoperation service, another service needs to run on the remote machine at the same time. This is a custom service for camera consumption that I wrote in Python (available in Github[69]) (RQ3). The purpose of this service is to provide video data (images) continuously. By having this service running, the client applications are able to get connected to the remote car and consume the camera through WebSocket connections. The diagram of the service is shown in Figure 31 (the left part - Server).

As Figure 31 shows, I used three packages: Picamera, Tornado, and Tornado Websocket. The Picamera package makes it possible to consume the camera through exposed methods and by having the Tornado and Tornado Websocket packages, two services are exposed on port 8080. One of these services is accessible through the HTTP protocol, which serves a picture that has been captured when the request is received. In fact, this service can be considered as a shutter button for the camera so that when a request is received, a picture is captured and served. Behind this service, another service stays listening for WebSocket connections. As a result, client applications can request this endpoint through WebSocket connections. When a request reaches this endpoint and a WebSocket connection is established, the client, as shown in Figure 31, is stored in a pool. This allows multiple clients can connect to consume the camera at the same

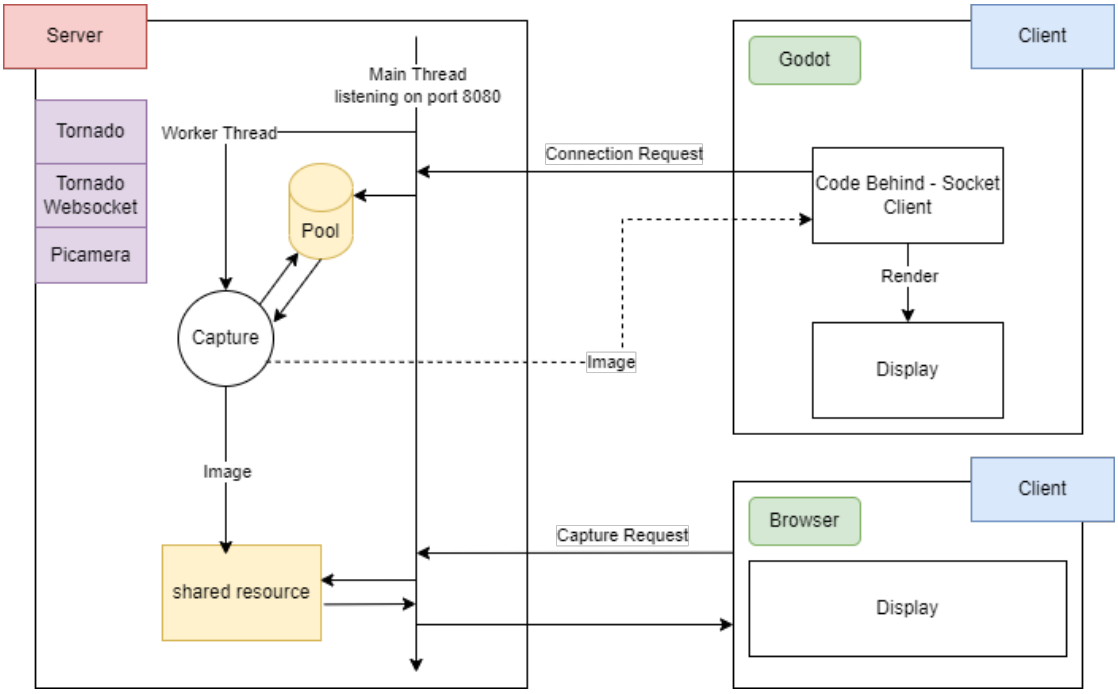

Figure 31. Custom Camera API and the client applications consuming the API

time. Considering all the clients are stored in the pool, there is a worker thread spawned by the main thread that has access to the pool. This thread is continuously capturing pictures and if there are clients available in the pool, it will send the images to the clients through the WebSocket connection already established. This thread also shares the newly captured image with the main thread so that when a request is received through an HTTP connection, the new image, which is just taken, is ready to be served. The main job of the worker thread is shown in the Listing 1.

```
def start_camera(loop):
    global wsVideoClients
    global is_service_stopped
    global image_stream
    global camera

    asyncio.set_event_loop(loop)

    while is_service_stopped == False:
        camera.capture(image_stream, format='jpeg', use_video_port =
            True, quality=100)

        if len(wsVideoClients) > 0:
            img = image_stream.getvalue()
```

```
            for wsClient in wsVideoClients:
                wsClient.write_message(img, binary=True)

        image_stream.truncate()
        image_stream.seek(0)

    print("camera stopped!")
```

Listing 1. The main job of the worker thread

As Figure 31 shows, two client applications (as an example), one via an HTTP connection (the Browser) and another one via a WebSocket connection (the Godot application), are connected at the same time. Consequently, they receive the video data and represent it to their end user via their own display. The images are compressed as JPEG and the resolution is 320X240, while the quality is set to maximum (100) and the frame rate is 30 fps. When an HTTP request for a camera image is received, the size of the image is also printed out. It shows each image has a size of around 40 KB. For improvement, an encoder instead of JPEG compression can be applied to reduce the size of the images. Since I wrote the service from scratch, it is flexible to adjust settings such as image resolution and frame rate. It also paves the way for applying encoders.

Figure 32 shows the execution of the custom camera service when it is running on port 8080. Also, it shows the size of an image, as an HTTP request for an image to capture has been sent. Afterward, a client application established a WebSocket connection so that it shows the number of the client applications connected (it is 1 in this case). Lastly, it shows the single connected client application gets disconnected and the service is stopped.

3.4.2 Client Application

The client application is in fact connecting the operator to Donkey Car. Then the operator can consume the services running on the car to teleoperate it. In fact, the operator receives feedback (camera images) and then based on the feedback, the decisions are made and the car is teleoperated.

During the experiments, I developed several client applications in Javascript[35, 67, 70, 71] and in Godot for windows (available in Github[72, 73]), but I implemented the final version of the client application in Godot (3D) and C#. The client application is ultimately built as an Android package (.apk) and installed in Meta Quest (available in Github[74]). Godot applications also need a plugin that adds support for running Godot applications on Meta Quest[75].

The final version of the interface is based on a free HDRI image (a 360° High Dynamic Range Image), which is set as a background and taken from Poly Haven[76] (see Figure 33) (RQ1).

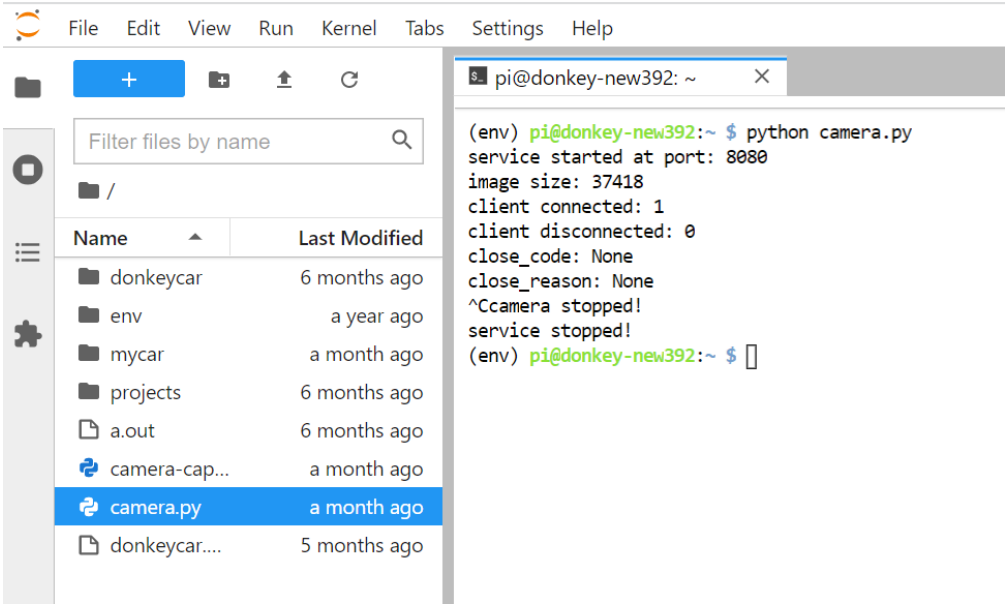

Figure 32. The execution and the log of the custom camera service

Figure 33. The interface has an HDRI image as a background

Behind this main interface, I also built two more interfaces that can be installed in Meta Quest. One of them contains a simple screen representing the camera images (available in Github[77]) (see Figure 34), and another one (available in Github[78]) is based on a free 3D model taken from 3DEXPORT[79] (see Figure 35) that the camera resides inside the 3D model and the video data is being represented in front of the 3D model.

As creating the virtual environments in 3D is one of the challenges that I discussed earlier, this area is also open for improvements. Because it needs its own professionals to design the 3D models from scratch and based on the needs hence free ready-made models are just to show the feasibility of the implementation. Also, the interfaces are in 3D. This means that while the operator has worn Meta Quest, can feel being in the model and have a 360° environment, but the figures attached are in 2D and unable to present all aspects of the environment.

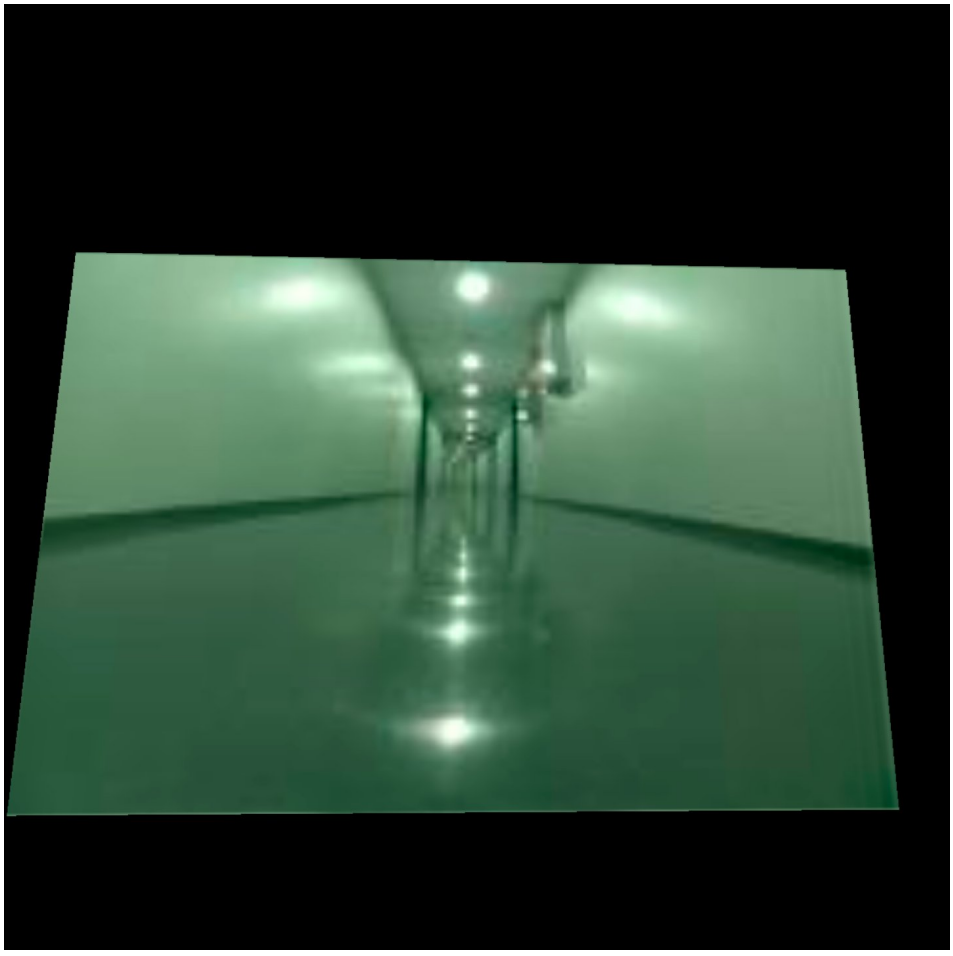

Figure 34. The interface contains a screen

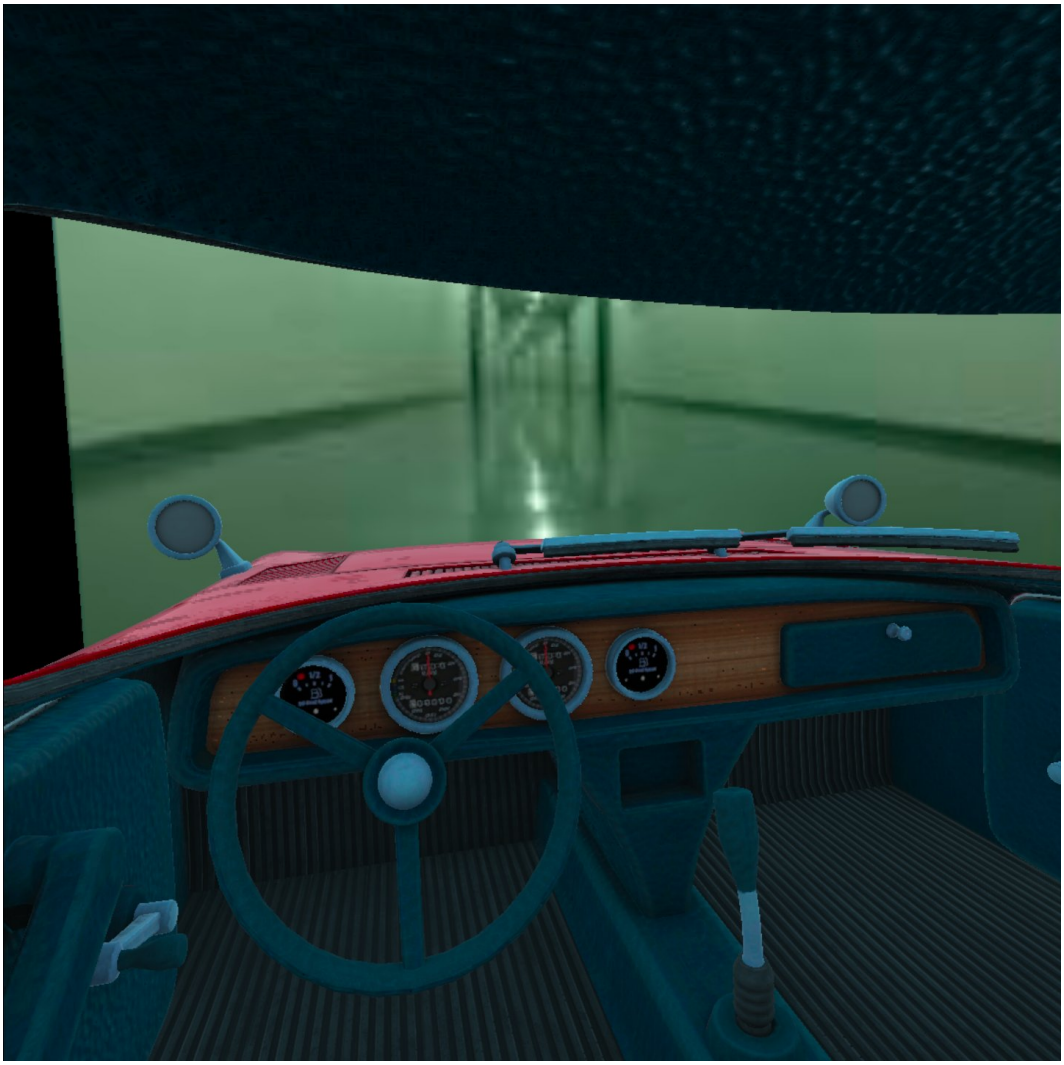

Figure 35. The interface contains a free 3D model

When the client application gets executed, two WebSocket connections are separately established. One of them connects the operator to the custom camera service running on the car to continuously receive the camera images and another connection is to connect the operator to the teleoperation service.

Once the connection to the custom camera service is made, the service starts transferring the video data. As a result, the client application receives real-time camera images and renders the images to display to the operator. The Listing 2 shows the function (in Godot-based client application) getting the call once a message containing the image is received and renders the image. The last lines print out the milliseconds it takes for each camera image to be received and rendered. The output, as shown in Figure 36, shows that the average time is around 40 ms.

```
long lastTimeMilliseconds = default(long);
public void _on_received_data()
{
    var img_bytes = ws.GetPeer(1).GetPacket();
    var image = new Image();

    var error = image.LoadJpgFromBuffer(img_bytes);
    if (error != Error.Ok)
        GD.PushError("Couldn't load the image.");

    var texture = new ImageTexture();
    texture.CreateFromImage(image);
    this.Texture = texture;


    long currentTimeMilliseconds = DateTime.Now.Ticks / TimeSpan.
        TicksPerMillisecond;
    if (lastTimeMilliseconds == default(long))
        lastTimeMilliseconds = currentTimeMilliseconds;
    else
    {
        GD.Print($"I/O duration for camera in ms: {
            currentTimeMilliseconds - lastTimeMilliseconds}");
        lastTimeMilliseconds = currentTimeMilliseconds;
    }
}
```

Listing 2. The function getting the call once the video data is received by the client application

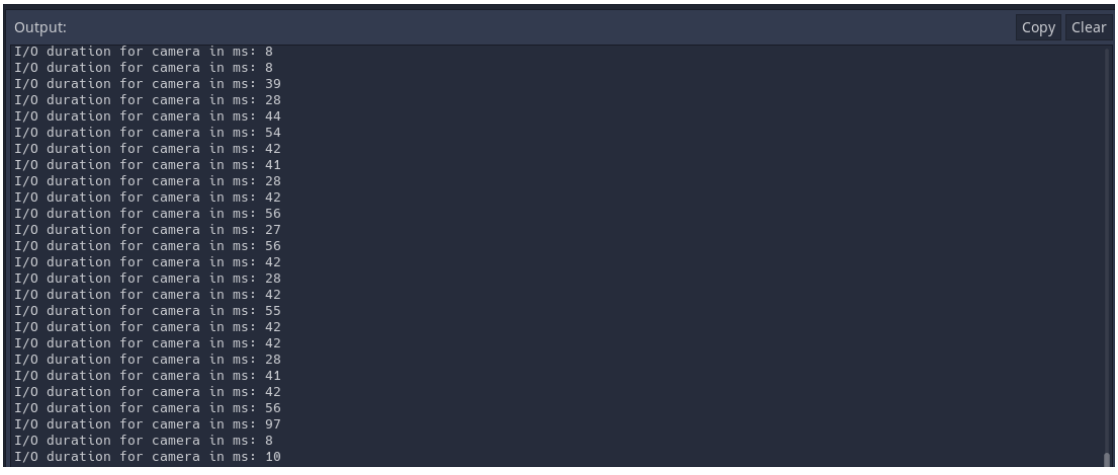

Figure 36. The log showing the delay for each image

Another connection established allows the operator to send the commands via the joysticks to make the car move (RQ2). As long as the operator pushes the Up button, the car will go forward and once the operator stops pushing it, the car will stop moving. The same is true for the Down key to make the car goes backward. While the car goes forward or backward, the operator with another joystick can push the Left and Right keys to give the car a direction. The teleoperation API remembers the last command sent so that when the operator stops pushing the Up or Down keys, another command should be sent to make the car forget the last command and stop. It means that when the car is going forward or backward, if it gets disconnected for any reason, it will keep going in the last direction it has received. There were several evaluation sessions with my supervisor that the car got disconnected so that it was going in the last direction that it had received.

Godot follows a hierarchical approach to structure the scene and a scene is defined as a group of nodes. Nodes, actually, are the building blocks of the game which are organized as a scene. For instance, if a character is needed to be in a game, a node of type KinematicBody should be added to the scene. The type of KinematicBody is a special type that is meant to be user-controlled. Then to add some characteristics such as movement or texture, another special type should be added. This time it should be added under KinematicBody. A node of type Sprite is added for such properties. Figure 37 shows the scene tree of the client application in Godot. As it shows, a root node for 3D scenes (Spatial) is added. In the second level, a node of type ARVROrigin is necessary for VR-based applications. In the last level, nodes for the camera (the user viewport) and joysticks together with Sprite3D are added under ARVROrigin. This is how the viewport for the operator is structured and it is possible to capture the user interaction with the joysticks. The Sprite3D node here is meant to be a screen to represent the video frames (camera images) to the operator.

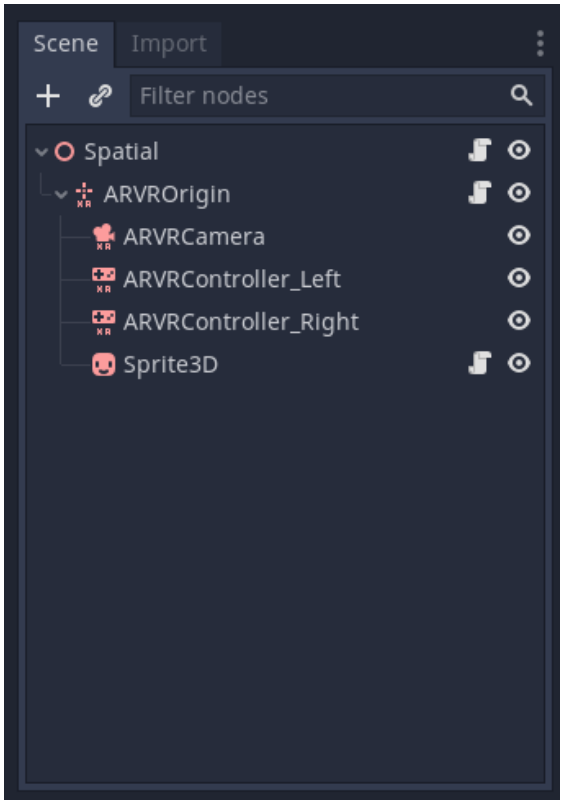

Figure 37. Godot Scene Tree of Nodes

3.5 Amount of Work

I spent 416 hours in total for thesis development.

Learning Godot Basics(2D and 3D) and Environment Integration:
56h

Client Applications in Javascript and Godot (2D, 3D):
80h

Providing the User Interfaces (the Virtual Environments):
40h

User Interaction via Joysticks:
16h

Raspberry Pi Investigation:
24h

Video Streaming Service:

56h

Getting Used to Donkey Car - Problems with Batteries, Network, Donkey Software:
32h

Teleoperation Service and Studying the Open Source Software:
40h

Donkey Car Open Source Software modification:
32h

Experiments, Tests, Debug, and Improvements:
40h

4 Conclusion

The goal of the thesis was to provide a minimum viable product and in the thesis, together with the basics and challenges of teleoperation, I introduced a practical solution to teleoperate a remote-controlled car (Donkey Car S1). The focus of the thesis was on VR and I covered this topic in the theoretical section; the practical solution was also based on VR. Here, I am explaining to what extent I implemented the requirements and what was my experience during the thesis. I will also describe future work.

4.1 Requirements

RQ1: regarding the user interface, an Android-based application is installed on Meta Quest that provides the operator with three interfaces, but as creating flawless virtual environments in 3D needs its own expertise. This area is open to building environments that align with the needs. I fulfilled this requirement successfully.

RQ2: in regard to user interaction via the joysticks, convenient use of joysticks is provided and the operator is able to drive the car with joysticks. I fulfilled this requirement successfully.

RQ3: regarding video streaming, I implemented a custom camera service based on TCP protocol that serves the camera frame by frame. It transfers images in JPEG compression with a resolution of 320X240, maximum quality, size of around 40 KB, and a frame rate of 30 fps. As it uses TCP protocol, it is reliable and the video quality is not degraded. Applying encoders (H.264 compression) is defined for the future because the effort was put into other areas such as providing three interfaces, and implementing a custom service from scratch. Providing the service from scratch is a great benefit because of the flexibility in defining the image settings such as resolution and frame rate, even though the approach is to stream frame by frame. The decision for this approach was to come up with a working solution first and then start improving. By migrating to real video streaming it is possible to benefit from the Picamera package support for H.264 streaming. The Picamera package support for H.264 streaming, makes it to apply H.264 encoding on the server-side more easily compared to the client side - investigation needed. I fulfilled this requirement successfully, as video data are transmitted frame by frame however, it can be improved by migrating to a real video streaming approach.

RQ4: in regard to the communication link, the components are integrated via an established WLAN and the delay is around 40 ms. The link is made via a 2.4 GHz router so that an indoor area up to 150 feet or an outdoor area up to 300 feet is covered. This established link is a better choice than hotspots, as they cover up to 33 feet. Another

option for the communication link is through the internet (via cellular networks) is defined for future work. During the thesis, I had several evaluation sessions with my supervisor and during those sessions sometimes the car got disconnected. The first assumption was the wifi conflicts. This may be defined as a further investigation to apply networking techniques to improve the communication link and to make the channel isolated. I fulfilled this requirement, but under particular circumstances sometimes the car got disconnected.

RQ5: regarding teleoperation service, I used the Donkey Car open source software to steer the car. This service is based on TCP and the commands from the operators' interaction via the joysticks are forwarded to this service. The service successfully receives the messages and allows teleoperation in real-time. I fulfilled this requirement successfully.

4.2 Godot

Godot is a perfect platform for making games and applications. It greatly supports VR. My supervisor recommended Godot and after that, I found it easy to learn. As Godot can be integrated with Microsoft IDEs and C#, it was straightforward to proceed with Godot, but the integration of Godot with Microsoft IDEs is challenging. Also still, C# is not recommended officially for production. It is a good approach to get started in C#, if there is some background in that regard. Because it is possible to start the project so fast, but it is better to migrate to GDScript after getting familiarized with Godot. Because there are some technical attributes that are supported only in GDScript. Lastly, the documentation is rich enough, but if there are some specific cases, there may be some trouble reaching a popular community or finding some clues pushing you toward the right path.

4.3 Donkey Car

Donkey Car is an ideal remote-controlled car. Having access to a toy car that comes with Raspberry Pi, makes a great potential for so much learning and fun. Donkey Cars come with a web-based interface that makes it possible to easily have access to the car and start driving, but I also should say that the documentation for Donkey Car is not rich enough, the community is not popular, and students should be ready for a lot of investigation. Nevertheless, driving Donkey Cars is exciting and I am quite confident that experience driving a Donkey Car with my solution in Meta Oculus will bring incredible excitement.

4.4 Future Work

While my solution to teleoperate Donkey Cars is practical, there are areas that can be improved or be defined for future work.

- The user interface can be defined for future work. Building 3D models and virtual environments needs its own expertise and for a better user experience and a higher level of telepresence, they should be built based on the needs. Also, the interface can be equipped with some techniques similar to path-planning algorithms to guide the operators while they are steering the car.

- Telepresence is an open area for future work. For instance, the remote machine can be equipped with sensors to provide, for instance, auditory and haptic feedback. Then representing the received feedback to the operator will take the telepresence to the next level. For instance, the operator can hear the remote environment or the joysticks will vibrate when the car runs into some obstacles.

- The custom camera service is an open area for improvements. With further study, it is possible to apply encoders to the service or change the approach to real video streaming and take the benefit of the supported h.264 encoders in Picamera package. Also, migration to other protocols than TCP or creating a new protocol as a mix of existing protocols can be defined for further study, as it helps with transmitting the video data faster.

- Improvement in network structure will help with the delay issue. By applying network techniques or considering other communication link options such as the internet and cellular networks, less delay may be noticeable and result in more smooth teleoperation.

- Architecture improvement can also be led to higher performance and less delay. As mentioned earlier in the related work section, similar to the NeCS-Car project[8], it can be tested if by having two separated systems dedicated for each service (custom camera service and teleoperation service), the higher performance and less delay may be achieved.

I fulfilled all requirements for an MVP. The prototype has been tested successfully in the Delta building by my supervisor and various colleagues. We all are looking forward to using it to attract more students into the exciting field of autonomous driving as well as advancing research to use VR in teleoperation. I hope that a lot of students will enjoy driving with and improving my teleoperation solution.

References

[1] Teleoperation: Set to revolutionize intermodal operations. https://en.wikipedia.org/wiki/Teleoperation.

[2] Donkey Car. https://www.donkeycar.com/.

[3] Ning-Ning Zhou and Yulong Deng. Virtual reality: A state-of-the-art survey, 2009.

[4] Oculus Quest 2. https://store.facebook.com/quest/products/quest-2/.

[5] Terrence Fong and Charles Thorpe. Vehicle teleoperation interfaces, 2001.

[6] Dominic Jud, Simon Kerscher, Martin Wermelinger, Edo Jelavic, et al. Heap - the autonomous walking excavator, 2021.

[7] A.E. Kaplan, S. Keshav, N.L. Schryer, and J.H. Venutolo. An internet accessible telepresence, 1997.

[8] Zeashan Hameed Khan. Wireless network architecture for long range teleoperation of an autonomous system, 2010.

[9] Daniel Olsson. Techniques for selecting spatially variable video encoder quantization for remote operation, 2020.

[10] S. Lichiardopol. A survey on teleoperation, 2007.

[11] Jean Vareille. Internet-based teleoperation: A case study - toward delay approximation and speed limit module, 2007.

[12] Anitha Kumari and Narendranath Udupa. A study of the evolution of video codec and its future research direction, 2020.

[13] A. R. Al-Ali, Ragini Gupta, Tasneem Zaman Batool, Taha Landolsi, Fadi Aloul, and Ahmad Al Nabulsi. Digital twin conceptual model within the context of internet of things, 2020.

[14] Cheng-Peng Kuan and Kuu-Young Young. Challenges in vr-based robot teleoperation, 2003.

[15] Stefano Nativi, Blagoj Delipetrev, and Max Craglia. Destination earth: Survey on "digital twins" technologies and activities, in the green deal area, 2020.

[16] Robocar ltd. https://www.robocarstore.com/products/donkey-car-starter-kit.

[17] 10botics. https://courses.10botics.com.

[18] Meta. https://about.fb.com/news/2020/09/introducing-oculus-quest-2-the-next-generation-of-all-in-one-vr/.

[19] tp-link. https://www.tp-link.com/us/service-provider/dsl-router/td-w8960n/.

[20] Teleoperation. https://en.wikipedia.org/wiki/Teleoperation.

[21] Stylianos Mystakidis. Metaverse, 2022.

[22] Matjaž Mihelj and Janez Podobnik. Haptics for virtual reality and teleoperation, 2012.

[23] Scott A. Greena, XiaoQi Chen Mark Billinghurst, and J. Geoffrey Chase. Human-robot collaboration: A literature review and augmented reality approach in design, 2008.

[24] Mehmet Ismet Can Dede and Sabri Tosunoglu. Parallel position/force controller for teleoperation systems, 2007.

[25] Princeton Brown. Sensors and actuators: Technology and applications, 2017.

[26] B.P. DeJong, J.E. Colgate, and M.A. Peshkin. Mental transformations in human-robot interaction, 2011.

[27] Janko Petereit, Jürgen Beyerer, Tamim Asfour, Sascha Gentes, et al. Robdekon: Robotic systems for decontamination in hazardous environments, 2019.

[28] Ron Frederick. Experiences with real-time software video compression. In Proceedings of the Packet Video Workshop, 1994.

[29] Einride. https://www.einride.tech/.

[30] DriveU. https://driveu.auto/.

[31] Ottopia. https://ottopia.tech/.

[32] Formant. https://formant.io/.

[33] Teksbotics. https://www.teksbotics.ai/.

[34] W.R. Ferrell. Remote manipulative control with transmission delay, 1965.

[35] Javascript Client Application Consuming Built-in Camera Service - Over HTTP - Speed Control Version. https://github.com/sherafatian-amirabbas/DonkeyCar-WebInterface/tree/SpeedControl.

[36] Godot. https://godotengine.org/.

[37] Android. https://www.android.com/.

[38] Mohsin Khan and Jason Gu. Web based teleoperation architecture and h.264 video encoder, 2012.

[39] IAIN E. Richardson. The h.264 advanced video compression standard. Wiley, 2011.

[40] Chris Griffith. Raspberry pi hardware encoding speed test. https://codecalamity.com/raspberry-pi-hardware-encoding-speed-test/, October 2021.

[41] K.H. Walse and D.R. Dhotre. Wireless network: Performance analysis of tcp, 2007.

[42] Roberto Oboe. Web-interfaced, force-reflecting teleoperation systems, 2001.

[43] Raul Wirz, Raul Marin, Jose M. Claver, Manuel Ferre, et al. End-to-end congestion control protocols for remote programming of robots, using heterogeneous networks: A comparative analysis, 2008.

[44] Websocket. https://en.wikipedia.org/wiki/WebSocket.

[45] Precisionmicrodrives. https://www.precisionmicrodrives.com.

[46] Matias Selzer and Martín Larrea. Anaru, a virtual reality framework for physical human interactions, 2015.

[47] Unity. https://unity.com/.

[48] Qingjin Peng. Virtual reality technology in product design and manufacturing, 2007.

[49] Autocad. https://www.autodesk.com/products/autocad/overview.

[50] Blender. https://www.blender.org/.

[51] Unreal. https://www.unrealengine.com/en-US/.

[52] Sebastian Richard Newrzella, David W. Franklin, and Sultan Haider. 5-dimension cross-industry digital twin applications model and analysis of digital twin classification terms and models, 2021.

[53] Abdulmotaleb El Saddik. Digital twins: The convergence of multimedia technologies, 2018.

[54] B. R. Barricelli, E. Casiraghi, and D. Fogli. A survey on digital twin: Definitions, characteristics, applications, and design implications, 2019.

[55] Ievgenii A. Tsokalo, David Kuss, Ievgen Kharabet, Frank H.P. Fitzek, and Martin Reisslein. Remote robot control with human-in-the-loop over long distances using digital twins, 2019.

[56] Raspberry Pi 4. https://www.raspberrypi.com/products/raspberry-pi-4-model-b/.

[57] Robotics Masters. https://roboticsmasters.co/.

[58] Pico Technology. https://www.picotech.com/library/application-note/some-power-pwm-drivers-for-electric-dc-motors.

[59] Bradley Mitchell. Life Wire. https://www.lifewire.com/range-of-typical-wifi-network-816564, 2020.

[60] Difference between wi-fi and hotspot. http://www.differencebetween.net/technology/difference-between-wi-fi-and-hotspot/.

[61] picamera. https://picamera.readthedocs.io/en/release-1.13/.

[62] Advanced Recipes. https://picamera.readthedocs.io/en/release-1.13/recipes2.html.

[63] Camera live streaming using h264 format. https://www.codeinsideout.com/blog/pi/stream-picamera-h264/.

[64] Jason Jerald. The vr book, 2016.

[65] Raspberry Pi OS. https://www.raspberrypi.com/software/.

[66] Donkey Car Source. https://github.com/robocarstore/donkeycar.

[67] Javascript Client Application Consuming Built-in Camera Service - Over HTTP. https://github.com/sherafatian-amirabbas/DonkeyCar-WebInterface/tree/master.

[68] Customized Donkey Car Free Open Source Software. https://github.com/sherafatian-amirabbas/donkeycar.

[69] Custom Camera Service. https://github.com/sherafatian-amirabbas/donkey-oculus/blob/interior_car_environment_custom_camera_api/PiCamera-API/camera.py.

[70] Javascript Client Application Consuming Added Web Socket Camera Service - Customized Built-in Source. https://github.com/sherafatian-amirabbas/DonkeyCar-WebInterface/tree/wsVideo.

[71] Javascript Client Application Consuming Custom Camera Service. https://github.com/sherafatian-amirabbas/DonkeyCar-WebInterface/tree/CustomAPI.

[72] Godot Client Application (2D) for Windows Without Camera. https://github.com/sherafatian-amirabbas/GodoPrototype.

[73] Godot Client Application (2D) for Windows Consuming Custom Camera Service. https://github.com/sherafatian-amirabbas/GodoPrototype/tree/CustomVideoAPI.

[74] Godot Client Application (3D) Consuming Custom Camera Service - Final Version. https://github.com/sherafatian-amirabbas/donkey-oculus/tree/interior_car_environment_custom_camera_api.

[75] Godot Oculus Mobile GDNative Module. https://github.com/GodotVR/godot-oculus-mobile-asset.

[76] Poly Haven. https://polyhaven.com/.

[77] Godot Client Application (3D) Consuming Added Web Socket Camera Service - Customized Built-in Source - Clean Environment. https://github.com/sherafatian-amirabbas/donkey-oculus/tree/clean_environment.

[78] Godot Client Application (3D) Consuming Added Web Socket Camera Service - Customized Built-in Source - Based on a Free 3D Model. https://github.com/sherafatian-amirabbas/donkey-oculus/tree/interior_car_3d.

[79] T gt2000 68 F 3D Model. https://3dexport.com/free-3dmodel-t-gt2000-68-388281.htm.

Appendix

I. Licence

Non-exclusive licence to reproduce thesis and make thesis public

I, **Amirabbas Sherafatian**,
(author's name)

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

   reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   Teleoperation of Remote Controlled Toy Cars in VR,

   supervised by Ulrich Norbisrath.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Amirabbas Sherafatian
16/05/2022