Mixed reality for control and visualization of imminent path of vehicles

Rasmus Hogslätt (rasho692)

LiU-ITN-TEK-A–24/026–SE

Friday 28th June, 2024 (12:41)

Summary

With future advancements and adoption of autonomous vehicles, it is of interest for humans to see where vehicles are headed. Meanwhile, advancements of mixed reality, MR, headsets indicate that headsets may eventually become so small and comfortable that they will be suitable for everyday wear. With this background, this thesis explores how visualizations in mixed reality impact situation awareness SA, of a human.

As autonomous vehicles do not require steering controls, but may still be blocking a human's path, it is of interest to explore how virtual controllers in MR can be used to move these vehicles. Therefore, this thesis also explores the impact of MR virtual controllers' usefulness when directing a vehicle as opposed to joystick controllers.

Given this context, the thesis' aim was to explore both of these topics, resulting in the implementation of a system where a remote-controlled, RC, car's imminent path was visualized and controlled by MR virtual controllers. The implementation of this was done in the Rust programming language and deployed on a Meta Quest 3 headset.

Evaluation of the system was done by subjectively based user tests, where the test included three scenarios. These included driving the car with a joystick without visualization and using virtual controllers with and without visualization.

Discussions around the impact of language barriers, hardware choice, alternative methods, and the interest in finding an optimal minimal amount of visualizations are had in regards to the results.

The study concludes that virtual controllers can provide a better user experience than joystick controllers and that visualizations do not prove to be significantly beneficial as opposed to no visualization.

Glossary

ADB Android Debug Bridge. 16

ANOVA Analysis of variance. 32

APK Android Package Kit. 16, 37

AR Augmented reality. 5, 23

Bevy An Open Source game engine written in Rust. It utilizes the entity-component-system, ECS, pattern for efficient data handling. 2, 3, 21, 23, 24, 26, 36, 40, 42

ECS Entity Component System. A memory efficient programming design pattern commonly used in game engines. 1, 20

ESC Electronic Speed Control. 14, 17, 19

GPIO General Purpose Input/Output. 15, 19

HMI Human Machine Interaction. 6, 7, 42

IR Infrared light waves invisible to the human eye.. 17

likert An answer to a statement can be made on the likert scale. The answer made on a scale representing the level of agreement with the statement. 30

Meta Meta, formerly Facebook. The company currently developing the Quest series of VR and MR headsets. 37, 40

MR Mixed reality. 1–3, 5–10, 12, 13, 16, 18, 36, 39–42

OpenXR An open standard for creating XR applications. It is agnostic to what AR or VR device is used. 21, 23

passthrough As per OpenXR's use, the ability to see the physical as captured by cameras on the headset. 16, 23

PWM Pulse Width Modulation. 14, 15, 19

RC Remote control. For example a remote controlled car is called an RC car. 1, 6, 8–12, 14, 16, 17, 19, 21, 24, 28, 29, 37, 39, 41, 42

SA Situation Awareness. 1, 5–7, 12, 29, 30, 38–40, 42

SART Situation Awareness Rating Technique. 3, 28, 29, 31, 34, 38, 42

SLAM Simultaneous localization and mapping. The problem of creating a map of an unknown environment whilst keeping track of the agent's location. 18

SUS System Usability Scale. 3, 28, 31, 34, 35, 38, 40, 42

TCP Transmission Control Protocol. 14

TLX Task Load Index. 3, 28, 31, 32, 34, 35, 39, 40, 42

UDP User Datagram Protocol. 3, 14, 15, 17, 18, 21

VR Virtual reality. 1, 2, 5, 23, 36

Vulkan A cross platform graphics and compute API used in Bevy engine. 36

wand A common, laser-pointer like, method of pointing and selecting things in VR and MR environments. 21

Contents

Chapter 1

Introduction

Autonomous and remote controlled vehicles are becoming more and more mainstream in many industries. This trend means that a growing portion of interactions today are between machine and human as pointed out by Wang, Zhao and Zhu [32]. At the same time, advances in mixed reality MR, a blend of the physical and digital world, open new frontiers in human and machine interactions, making situation awareness SA, a key aspect to consider when developing these systems. This was highlighted in the study by Choi, Ahn and Seo[25], which concluded that increased SA provided by VR in warehouses with forklifts reduced accidents. They also highlighted the importance of improving SA in these environments, as forklifts are among the machines associated with most occupational deaths. In 1994, a taxonomy proposed by Milgram and Kishino [24] defines MR as a continuum ranging from the real environment, augmented reality, AR, augmented virtuality to virtual environments. As of today, most MR headsets are large, heavy, and have rather short battery lives. However, given rapid advances in the field and ample amount of time, the form factor of MR headsets will likely converge towards a device more suitable for all day wear[2]. Therefore, being able to visualize intent of autonomous vehicles to a human, thus hopefully increasing SA, and let a human interact with and get feedback from a machine through the ways of mixed reality is becoming more and more relevant when humans and machines are becoming more integrated in day-to-day tasks.

1.1 Background

Voysys is a company building software for low latency tele-operations, largely focused on remote control of vehicles. A significant part of their market is based on remote control of warehouse forklifts that operate in an environment with human workers. Occasionally, a worker needs to move a forklift or could benefit from knowing where a forklift is headed to avoid colliding. Thus, they want to explore the benefits of utilizing a mixed reality headset, specifically the Meta Quest 3 headset, when it comes to

1. Controlling an autonomous vehicle through a mixed reality interface as opposed to regular means of control

2. Visualize the current trajectory of an autonomous vehicle through mixed reality

By exploring this technique today, the hope is that they can reap the potential benefits of such technique when the hardware is more suitable for all day wear in an environment where human-machine interactions, HMI, are common, such as a warehouses with both autonomous forklifts and humans. Ideally, they also want potential benefits of MR visualizations to be transferable to other vehicle types, such as drones or airplanes, as these vehicles types are involved in other parts of their business.

1.2 Aim

The aim of this project is to develop and evaluate a mixed reality, MR, system for controlling a remote-control, RC, car. This system should allow users to navigate the car to specific physical locations using virtual controls and visualize its trajectory within a MR environment. The project will compare this MR-based control method with traditional analog controls, with and without visual feedback of the car's trajectory from the headset, focusing on user experience, situation awareness, and usability. The goal is to evaluate the effects of these interaction types on the perceived user experience.

1.3 Research questions

To achieve the aim of this thesis, the aim was broken down into three research questions.

1. *How can a MR headset be used to visualize the imminent trajectory of an RC car in mixed reality?*
2. *Does visualization of the RC car's trajectory contribute to an improved perceived situation awareness, SA?*
3. *How do virtual controls in a MR environment, as compared to traditional joystick controllers, perform in terms of user experience, situational awareness and efficiency for directing a RC car?*

1.4 Limitations

This study is constrained by several factors

- The headset being used in the study is a Meta Quest 3, as this is what is feasible given the current options. This infers any technological limitations and features of the Meta Quest 3 headset and its developer tools.
- The use of 1:10 scale RC cars as subject vehicles, rather than full-scale autonomous vehicles that are not easily accessible.
- Virtual controls in this case refer to the use of a hand controller to point to locations in MR.

Chapter 2

Related work

There has not been a lot of research done into the integration of MR into industrial environments and how these relate to SA and impact on HMI. However, similar work was done in 2021 using a Hololens on three type of robots. They visualized the trajectories of these robots in MR but ran into the issue of low accuracy spatial mapping[26]. This emphasises the need to achieve a good spatial mapping between the headset and the car.

Research has also been done evaluating the collaborative aspects provided by virtual and mixed reality elements. They visualized subject persons field-of-views as frustums and eye gazes as lines. The paper concluded that virtual cues in MR provide a feasible option in collaborative environments and that perceived social presence was improved[27]. This research indicates that visualizations in MR can improve the perceived user experience.

There has also been research published in 2013[19] into using the tracked 3D-movements of a pen to let a user set trajectories of vehicles. The study concluded that this interface was effective for trajectory planning, thus providing cues that the virtual controllers in this thesis could be beneficial.

Research by Choi, Ahn and Seo[25] explored factors impacting forklift operators' SA in order to minimize occupational forklift accidents. They concluded that attention narrowing tasks, caused by high cognitive load, descreased SA. Cognitive load is therefore an aspect to consider when aiming to increase SA.

Other research on published in Applied Ergonomics[21] explored how different headsets compared to each other in regards to usability when performing simulated order picking tasks. They also tested various visualization designs. They concluded that the visualization design had a remarkable higher impact on the task performance than the type of headset being used. This suggests that the choice of headset used is of less importance than how the visualization within the MR environment is designed.

Research has also been done in regards to trust of autonomous vehicles and visualizations by comparing visualizations in simulated video environments[15]. It highlighted that trust of autonomous vehicles is an important aspect required for their broader adoption. Results of the study show that providing a user with visualizations of an autonomous vehicles planned trajectories and the system's uncertainties provided more trust in the autonomous vehicle. Thus, creating useful visualizations in MR may increase trust in autonomous vehicles, facilitating their adoption.

Chapter 3

Design of path planning in MR

To achieve the aim of this thesis, a design for the visualization of the vehicle's intent should be designed. This involves visualization of trajectory that provides feedback to the user. A method of controlling the car also needs to be designed. This chapter explains the design of the visualizations, interactions and how they can be combined to allow for testing of the MR system so that the research questions can be answered.

3.1 Visualization design

To achieve the aim, the visualization needs to

- Give feedback on the car's trajectory
- Provide feedback on the car's speed
- Give feedback on what the car could do, such as the maximum turning angles it could achieve

These visualizations would facilitate easier avoidance of collision, and help the user control vehicles. In cases of RC vehicles in a warehouse, it would also allow users to more efficiently plan their walking routes as they would see where vehicles not only were, but were going to be. Ideally, the design would also be possible to slightly modify as to facilitate other vehicle types than cars and forklifts.

Previous research from 1988 by Sweller [30] developed on the previous work on human information processing. This resulted in the cognitive load theory, essentially stating that a human's brain can only process a limited amount of processes at a time and that all systems introduce some amount of cognitive load. Different MR systems has been shown to produce varying amounts of cognitive load, largely depending on application and implementation, as concluded by researchers at the University of Duisburg-Essen[13]. They showed that how MR is implemented in a system can either increase or reduce cognitive load. A poorly designed MR implementation could therefore be counterproductive in regards to cognitive load, whilst a well design implementation could be

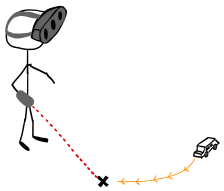
benefitial. A paper studying how the design of a MR-application impacted the cognitive load showed that limiting the number of digital visual elements that are displayed reduced the cognitive load[33]. Thus, they concluded that keeping the number of digital elements low, whilst also not adding unnecessary details, would have a positive impact on the cognitive load. This conclusion means that the visualization design should aim to minimize the number of visual elements and not add unnecessary complexity in order to reduce cognitive load.

In order to achieve high level of spatial awareness when integrating visual digital elements into the physical environment, it is important to have an accurate spatial mapping between the digital and real worlds' coordinate systems. This was outlined in [26] where the spatial mapping of low accuracy limited the level of spatial awareness. This was concluded in experiments where a user was tasked with pointing at locations in the physical environment using virtual controls. This research thus indicates that a MR system that controls a RC car needs to achieve a high enough level of accuracy in the mapping between the headset's coordinate space and the car's coordinate space.

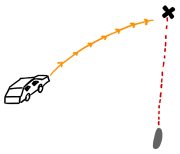
Car video games, such as seen in Figure 3.1 often use arrows or other design elements that indicate the optimal path to drive.

Figure 3.1: Screenshot from the game Forza Motorsport 6.

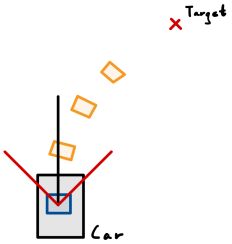
By using inspiration from car video games and the the conclusions from the research by Sweller[30] and the researchers at University of Duisburg-Essen[13], an initial design sketch was done as seen in Figure 3.2.



(a) External view of system. User with headset selects a target. The dotted line represents the virtual pointer used to select said target. The arrowed line depicts a visualization of the path the car is traveling.



(b) Point-of-view as seen from the user.



(c) Top down view of elements.

Figure 3.2: Illustrative view of the system.

As seen in Figures 3.2a and 3.2b, the user would use the right hand controller to point at a location to which the car would drive.

As seen in Figure 3.2c, the design would display the imminent trajectory of the car with squares positioned along its trajectory. Two red lines would indicate the maximum possible steering angles the car was able to turn. Also, a black line, scaled by the current speed of the car, would indicate how fast the car was driving. The simple elements included in this design were based on the prioritization of minimizing cognitive load. These elements could also be extended to simple 3-dimensional elements, allowing for visualization of other types of vehicles, such as drones.

Other visualization element were also explored. For example, arcs and arrows from the car to the target were tested. Ultimately, only one design could be evaluated in the scope of the project. The sketched design was therefore chosen as the squares could be easily extended to further dimensions, which would be useful if other vehicle types were to be tested. For example, a flying drone would require 3-dimensional visualizations. In such case, squares can easily be extended to cubes, whereas 3-dimensional arcs or arrows do not provide many depth cues and are less feasible for 3-dimensional environments.

3.2 Interaction design

The interaction design for controlling the RC car in MR was built on using the Quest 3 headset and its controllers seen in Figure 3.3. These controllers have buttons that can be either pressed or not pressed. They have triggers that can be pulled to various levels and joysticks that can be tilted in any direction. The controllers' directions and positions are also tracked so that the controllers can be used as laser pointers.

Figure 3.3: Quest 3 controllers.

Given these controllers, many types of interactions could have been used. For example, the controllers direction could be used to update the turning angle of the car while using one trigger to control speed forward and another for the speed backwards. The Quest 3 also has hand gestures capabilities, which could also be used to control the RC car.

In order to avoid excessive cognitive overload, the goal was to require as few interactions with the buttons, triggers and joysticks as possible. This led to an initial interaction design. It aimed to use one controller as a laser pointer, and upon activation with the trigger, the target to which the RC car would drive to would be set. The car would then adjust its turning angle to drive there with a fixed speed and the visualizations would be updated. This design would require only one controller's direction and its state of the trigger.

User feedback indicated the desire for also controlling the speed. Consequently, the interaction design was updated so that the target was set upon any activation of the trigger, and the degree to which it was pulled would adjust the speed.

This interaction design, which will be called *virtual control*, only required one controller when interacting with the car, and only one pull of the trigger was needed to update the RC car's speed and setting its target location.

Research question 3, *"How do virtual controls in a MR environment, as compared to traditional joystick controllers, perform in terms of user experience, situational awareness and efficiency for directing a RC car?"*, inferred the need for a joystick based control method too. Therefore, another means of interacting with the car was also designed. This design, which will be called *joystick control*, also required only controller, and its joystick was used adjust the speed forward or in reverse by tilting the stick forward or backwards. Likewise, a left of right tilt of the stick would adjust the car's turning angle. Any visualizations would be updated according to the car's actions.

At this point, two means of controls have been designed, described in table 3.1.

Table 3.1: Interaction designs

Virtual control	Joystick control
1. Point at location 2. Pull trigger • Update target • Trigger adjusts speed 3. Calculate turning angle to reach target 4. Update car's speed and turning angle 5. Update visualization	1. Tilt stick • Set angle • Adjust speed 2. Update car's speed and turning angle 3. Update visualization

In order to address the research questions 1 and 2, *"How can a MR headset be used to visualize the imminent trajectory of an RC car in mixed reality?"* and *"Does visualization of the RC car's trajectory contribute to an improved perceived situation awareness, SA?"*, the visualizations could also be toggled on or off. In this case, the need for updating the visualization was not required, removing step 5 for *Virtual control* and 3 for *Joystick control* in table 3.1.

3.3 Driving modes

Ultimately, given the visualization design and means of interactions, four possible driving modes were possible, as seen in table 3.2.

Virtual control WITH Visualization	Virtual control WITHOUT Visualization
Joystick control WITH Visualization	Joystick control WITHOUT Visualization

Table 3.2: Possible driving modes

As laser-pointer like interactions in MR is a common interface, a line visualizing the controller's direction was shown to the user regardless of any other visualizations. The user was therefore never forced to guess where they were pointing. This line would only be turned of when using the joystick control method.

Chapter 4

Hardware and Data Transmission

Given the design in Figure 3.2, a technical implementation of it was required to allow the testing needed to fulfill the aim of the thesis. By analyzing the design, the entire system could be broken down into separate subsystems by tasks. Three main technical tasks that needed to be fulfilled where identified:

- Overlaying virtual visualizations on the physical environment
- Receiving spatial data from car
- Send steering data to car

This chapter describes the choices of hardware for each subsystem and how they impacted the choice of software tools.

4.1 Hardware choices

At the time of the project's start, in January 2024, there were a few different choices of MR headsets available. Some of the most well known options were the Microsoft's Hololens 2, Meta's Quest 3 and a potential upcoming release of Apple's Vision Pro headset. For this thesis, the Quest 3 was chosen as it had been released a few months prior, allowing ample time for significant bugs to be resolved. The Vision Pro was yet to be released and its developing environments were still unknown. Furthermore, the Hololens 2 was significantly more expensive, whilst not providing additional features relevant to the project.

Yet another advantage of Quest 3 headset was the controllers which already provided built in tracking. One control for each hand was available, but as selecting a location could easily be achieved with only one controller, the other controller could be put on the car, allowing tracking of the car. This way of tracking would likely be easier than implementing other tracking methods, allowing more time to be spent on the visualizations and control of the car, which were the goal of the project. As tracking of the original Quest 3 controllers were largely dependent on the camera of the headset, resulting in loss of tracking in bad viewing angles, the Quest Pro controllers were used instead. These form

their own representation of their environment, thus providing more accurate tracking regardless of what the headset's cameras see.

The Quest 3 headset uses the Android operating system, forcing the project to target this platform. This would, as discussed in the software chapter in 5.1, have consequences for the selection of software tools.

The RC car that was used was a 1:10 scale truck made by Traxxas. It had front wheel steering and four wheel drive. This meant that a servo and a motor needed to receive input from the headset. RC cars, including this car, are typically controlled by sending pulse width modulated, PWM, signals to the electronic speed controller, ESC, for the motor and the servo for steering. The ESC is responsible for regulating speed and direction of the motor based on the width of the PWM pulse received. Similarly, the servo's position is determined by the width of its received PWM pulse. Thus, controlling the car requires two separate PWM pulses. An image of the car is seen in Figure 4.1.

Figure 4.1: RC car without body.

4.2 Data transmission technique

The Quest 3 has WiFi and Bluetooth capabilities, allowing data to be received and transmitted. Several ways of transmitting data was thus available. For example, a WiFi module, such as the ESP8266 as seen in Figure 4.2, could be attached to the car to transmit data using a communication protocol. Another option was to attach a Raspberry PI to the car. As previous experience was had from WiFi modules, and their smaller size as compared to the Raspberry PI, that approach was chosen for this project. For this project, the user datagram protocol, UDP, was chosen over other communication protocols such as the transmission control protocol, TCP. UDP, unlike TCP, does not verify that the data packets are received successfully. To combat this, up to 50 UDP packets were sent each second, in case some were not properly transmitted and received.

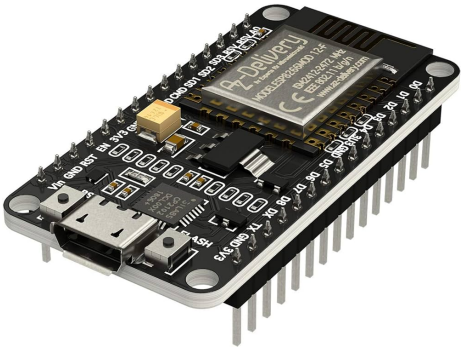


Figure 4.2: The WiFi module used. A NodeMCU ESP8266.

The steering of the car could thus be summarized by letting the headset send a throttle and angle value via UDP. The NodeMCU ESP8266 would then convert these values to PWM and output them to the car via its general-purpose input/output GPIO pins, to control the car's movement.

4.3 System architecture

Given the choice of hardware and selection of transmission techniques, a system architecture was created as seen in Figure 4.3 to provide an overview of the entire system and its subssytems. It shows what data is being sent from and to where and what processing is being performed at each subsystem.

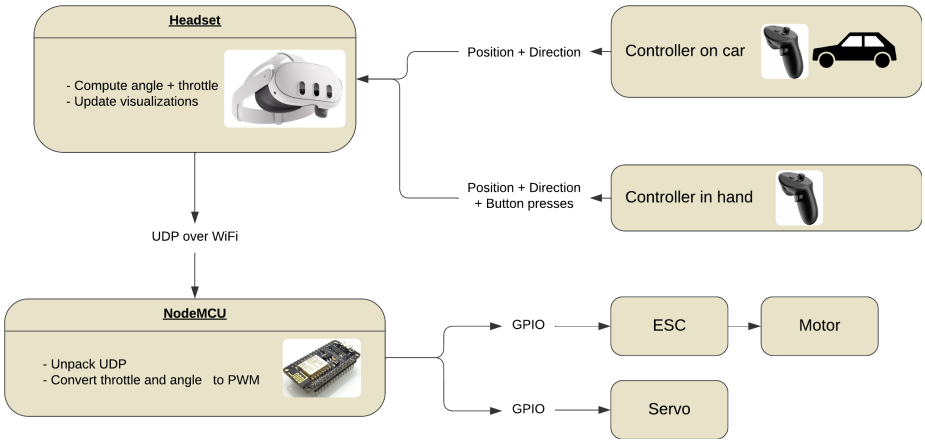


Figure 4.3: System architecture.

Chapter 5

Technical implementation

After having a design of the system and all necessary components and their relationship to one another, the technical implementation could be done. This would result in a working system that could be used for evaluation purposes. The three subsystems that needed to be developed, that is, handling visualization, receiving data from the RC car, and sending data back to it, were implemented separately. With each subsystem working, all systems were integrated into the final system.

5.1 Software tools and programming languages

For this project, the Rust programming language was used, as prior experience of Rust was had in regards to transmitting data and developing systems using the Bevy Engine. Other, more commonly used options for MR development were Unity or Unreal Engine. However, given little experience with these game engines, the Bevy Engine was preferred for this project. However, the Bevy Engine was not yet mature enough to support passthrough for MR development. Therefore, given the open source nature of the engine, the initial plan was to contribute with a passthrough feature implementation to the engine. If this had turned out to be too difficult or require too much time, the plan was to transition into Unity or Unreal Engine at an early stage.

As the system would be deployed on a Quest 3, which runs on the Android platform, this meant that the code had to be compiled into an android package kit, APK. The APK is a file format that packages all parts of an application into one that can be deployed on Android operating systems. If other headsets, such as the *Apple Vision Pro*, had been used instead of the *Quest 3*, the software tools might have changed as the final system would need to target an entirely different platform.

Compiling an APK file in Rust requires cross-compilation. A common tool for this is *Xbuild*. It allows cross-compilation by checking the build requirements of the target by receiving data from the android device bridge, ADB, from the target platform, and adjusting build configurations to that device. Thus, any code written in Rust on the *Windows* or *MacOS* operating systems could be compiled into an APK that could run on the Quest 3 headset.

Further, the WiFi module, NodeMCU ESP8266, can be programmed in

several ways. All of which require a program to be flashed onto the unit. Commonly, Arduino with C/C++ or Python is used to achieve this task. Due to better documentation of the Arduino suite of tools, C/C++ and Arduino was used to program this module.

In short, the planned programming languages used for the project were thus Rust and C++.

5.2 A common UDP package structure

The RC car was controlled using two signals. One for ESC controlling the motor and one for the servo controlling the steering. This meant that two signals had to be sent from the headset to the NodeMCU. To properly be able to debug the contents of the packages, the UDP packages were defined as text strings as "$throttle : X, servo : Y$" where X and Y were integer values in ranges 0 to 100 and 0 to 180 respectively. Having them be strings, rather than integers meant that debugging would be easier, as logging the values and type would be straightforward. However, removing the text strings could have reduced the size of the packages to minimize size. In this case, the size of the packages was not a problem.

With the UDP package structure defined, it was now given that the Quest 3 headset would need to adhere to this structure.

The car's motor's response was not directly mapped to the value sent to the ESC. A value of 50 was meant to produce a neutral response, 0 would be full reverse and 100 full forward. As the action is dependent on the actual circuits in both the car and NodeMCU, responses to varying values were therefore tested and ranges for varying ranges were noted. This resulted in three ranges as defined in table 5.1.

Table 5.1: Throttle ranges

Car response	Throttle ranges
Forward	[0, 38]
Neutral	[39, 48]
Reverse	[49, 100]

5.3 Receiving data from car

A common method for tracking an object in virtual reality and receiving its spatial data is to use fiducial markers[16]. This involves place fiducial markers in the physical world that can be tracked by the headset using relevant software. In this case, that would require direct access to the camera feeds of the Quest 3 headset. At the time of writing, Meta, the company behind the headset does not allow this access due to privacy concerns[7]. Therefore, other methods had to be used.

As the headset is equipped with two spatially tracked hand controllers, as seen in Figure 5.1a, another approach was to attach one controller to the RC car and let all of the steering be handled from the other controller. These controllers are tracked using the IR and color cameras of the headset and IR

lamps on the controllers. This meant that the technique for tracking would be based on Meta's algorithms for tracking. A drawback with this, however, is that the tracked controller is required to always be within the field of view of the headset and in a well lit environment. After testing this approach, it seemed that the tracking would not be sufficiently good when the car, and thus one controller, traveled to far away from the headset.

(a) Quest 3 controllers.

(b) Quest Pro controllers.

Figure 5.1

In order to resolve this issue, the standard controllers in Figure 5.1a were replaced with Meta Quest Pro's controllers in Figure 5.1b. These controllers are compatible with the Meta Quest 3 headset and are self tracked. Each of these controllers were equipped with three cameras at the front, facing different directions, as well as with a more powerful processor. They would then, using Meta's implementation, perform a simultaneous localization and mapping, SLAM, algorithm. SLAM allows construction of a 3D scene given several 2D images of the environment. They would thus localize themselves and communicate and map this to the headset's location. This approach did not require the controllers to always be in the field of view of the headset and were also less dependent on a well lit environment. This approach thus resolved the issues with the standard controllers in Figure 5.1a.

Although this is not the typical way of handling tracking in MR, it was an easy solution that allowed more time being spent on the essence of the thesis, namely the visualization and usefulness of the system, not the tracking technique itself.

5.4 Transfer data to car

When spatial data from the car had been received, including both location and direction, the headset would need to compute the desired angle and throttle. These would then need to be packaged according to the UDP structure described in 5.2.

Transferring the data would be done using UDP packages. Therefore, a UDP-socket was bound to, with the receiver being the NodeMCU attached to the car. The signals, throttle and servo, were then used to construct the package which could be sent to the receiving side. As previously described, due to the inconsistencies of UDP compared to other methods, fifty messages were sent each second. Also, in case the user was not performing any actions, thus not yielding changes in signal, neutral signals, i.e. throttle values of 50 and servo values of 90 were sent. This meant that the car would not continue to receive signals when not prompted by the user.

In order to receive the data on the NodeMCU side, the board had to be configured to receive the incoming packages. This was done by setting the correct local port and listening to it.

When the packages were received, they were unpacked into their respective values, throttle and servo. These values were used to output the proper pulse width modulations, PWM, via the GPIO pins on the NodeMCU board. PWM is the type of signal that the ESC and servo takes. It encodes the value of the signal as the width of a pulse.

The period of a PWM is defined as the time it takes for the wave signal to perform one cycle. For RC car components, this is typically 50 Hz. The duty cycle is then defined as the fraction of time of a period that the wave signal is high versus low. Given that the frequency of the period is 50 Hz, which corresponds to a period length of $T = \frac{1}{f} = \frac{1}{50}$ seconds $= 0.02$ seconds or 20 milliseconds, the duty cycle, i.e., the width of each pulse, can be expressed as a percentage of the period length. The equation for the duty cycle D in percentage is thus derived by equation 5.1.

$$D(\%) = \left(\frac{\text{pulse width (ms)}}{20 \text{ ms}} \right) \times 100\% \tag{5.1}$$

An example of a pulse width modulation is seen in Figure 5.2.

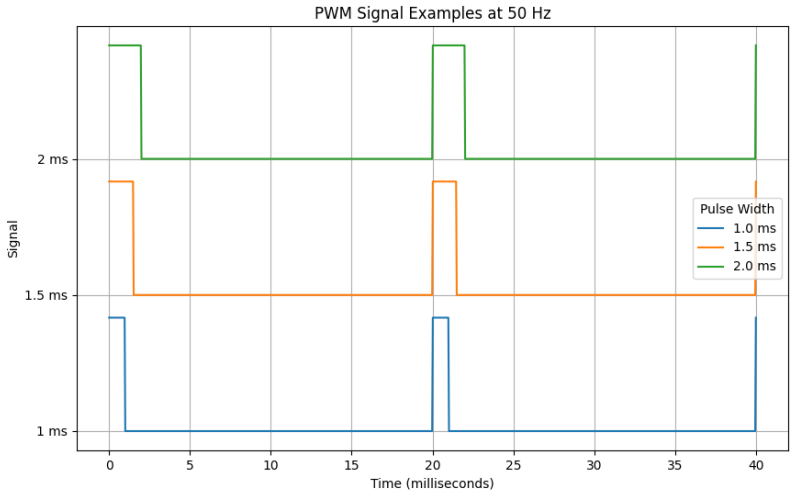


Figure 5.2: Example of PWM. Duty cycle is 100%, 50% and 0% from top to bottom respectively.

This knowledge allowed for a throttle value in range $[0, 100]$ and a servo value $[0, 180]$ to be remapped into the range $[\frac{1}{20}, \frac{2}{20}]$ and be written to the GPIO pin connected to the motor and servo respectively using Arduino's *analogWrite* function.

5.5 Base application in Bevy

Bevy is an open source game engine written in the rust programming language. It has no native editor, thus development of the application was done in written code, as opposed to no-code solutions available in other game engines. It uses an entity component systems, ECS, which means the an application is built using entities, components and systems. This is one approach to handle data in game engines which efficiently utilizes the memory of the system.

Entities are unique identifiers that represent game objects. Often, this is just an integer value. The entities themselves do not carry any more data. Instead, data is stored in components, which can be thought of as data structures without behaviour. These components can be added to entities. The systems handle the logic of the application. They can be thought of as small functions that run in parallel, although schedules can alter whether a certain system should be run before or after certain other systems if needed. A system queries entities with specific components, whose data can be read or written to. A minimal example of an ECS application in Bevy is provided in Listing 5.1.

```
// Entity
struct Entity(u64);

// Component definitions
#[derive(Component)]
struct Person;

#[derive(Component)]
struct Name(String);

// Spawn persons with names
fn add_people(mut commands: Commands) {
    commands.spawn((Person, Name("Rasmus".to_string())));
    commands.spawn((Person, Name("Donald Duck".to_string())));
    commands.spawn((Person, Name("Mickey Mouse".to_string())));
}

// System definition
fn greet_people(query: Query<&Name, With<Person>>) {
    for name in &query {
        println!("hello {}!", name.0);
    }
}

fn main() {
    App::new()
        .add_systems(Startup, add_people) // Add people at
    application start
        .add_systems(Update, greet_people) // Greet each person
    that has a name each iteration
        .run();
}
```

Listing 5.1: Minimal ECS application in Bevy

Commonly, ECS also supports resources. Resources are, like components, data structures, but rather than being attached to entities, are global data that can also be queried for by systems.

Knowing how ECS applications are built in Bevy and that the Bevy application would need to fulfill three requirements; handle visualizations, handle

user interaction and send a UDP package to the NodeMCU, pseudocode for the Bevy application could be developed as seen in Algorithm 5.5.

Algorithm 1 Bevy ECS Application Structure

Step 1: Initialize the ECS World
Setup for UDP packages
Add Resources, Plugins
Spawn world entities with attached components
Step 2: Application loop
while app is running **do**
  System: Set target to drive to
  System: Get position and direction of car
  System: Compute desired throttle and angle
  System: Send UDP package with throttle and angle
  System: Update visualization
end while

5.6 Selecting target

Selecting the location to which the RC car would drive towards, was done using the right controller as a laser pointer, often called wand. In order to achieve this, a transparent plane mesh was positioned at floor level. The location and direction of the controller in the scene could thus be used to raycast against the plane. Data about the controller, such as a location, direction, key presses etc. were handled by the plugin *bevy_oxr*[4], which provides an integration with OpenXR and Bevy. Further, the raycasting was achieved by utilizing Bevy's third party plugin for raycasting, *bevy_mod_raycast*[3]. The intersections location could be retrieved when the trigger button of the controller was pressed, and a target marker could be spawned. If this target marker was already spawned, it's position would be updated to the location of the intersection. Using Bevy's gizmo plugin, the casted ray and the intersection with the plane was visualized. This allowed the user to see where they were pointing the controller.

5.7 Car's angle and distance to target

Given both the target's location and the car's location and direction, the desired values, throttle and angle, sent to the car were possible to compute.

The direction of the car was assumed to be the same as for the controller attached to the car, which was possible to retrieve using *bevy_oxr*. This direction was not necessarily aligned with the floor. A directional vector for the car projected onto the floor was therefor computed according to equation 5.2

$$\vec{v}_{\text{floor}} = \hat{v} - (\hat{v} \cdot \hat{n_{\text{floor}}})\hat{n_{\text{floor}}} \tag{5.2}$$

where $\vec{v}_{\text{floor}}$ is the normalized directional vector aligned with the floor, $\hat{v}$ is the normalized direction received from the controller and $\hat{n_{\text{floor}}}$ is the normalized normal of the floor.

Given both the car's location as vector $\vec{p_{\text{car}}}$ and the target's location as vector $\vec{p_{\text{target}}}$, a directional vector $\vec{v_{\text{ct}}}$ was computed according to

$$\vec{v_{\text{ct}}} = \vec{p_{\text{target}}} - \vec{p_{\text{car}}} \tag{5.3}$$

representing the non-normalized directional vector from the car towards the target. Let $\hat{v_{\text{ct}}}$ be the normalized version of this vector. The angle between $\hat{v_{\text{floor}}}$ and $\hat{v_{\text{ct}}}$ could then be computed according to

$$\theta = \arccos\left(\hat{v_{\text{floor}}} \cdot \hat{v_{\text{ct}}}\right) \tag{5.4}$$

where θ is the positive angle between the two vectors. Given it is a positive angle, calculating whether the target was in front of or behind the car also was also computed according to

$$y = \begin{cases} 1 & \textbf{if} \quad \hat{v_{\text{floor}}} \cdot \hat{v_{\text{ct}}} \geq 0 \\ -1 & \textbf{if} \quad \hat{v_{\text{floor}}} \cdot \hat{v_{\text{ct}}} < 0 \end{cases} \tag{5.5}$$

where a positive value of y meant the target was in front of the car and a negative y meant the target was behind the car. If the target was behind the car, the angle θ was multiplied by -1 as the car would need to reverse to reach the target. These scenarios are seen in Figure 5.3

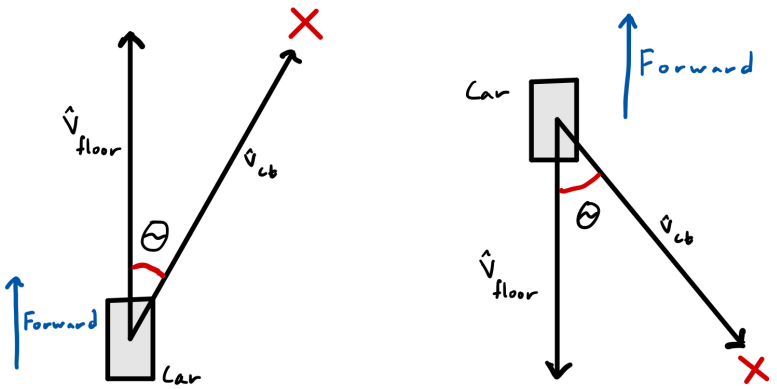


Figure 5.3: Left: Target in front of car. Right: Target behind car.

The distance between the car and the target was given by the norm $|\hat{v_{\text{ct}}}|$.

At this point, three key values have been computed, the angle θ, *distance* and y. Further, a value *trigger* was also retrieved from the controller, a value in range $[0, 1]$ representing how far the user pushed the trigger button of the controller for selecting a target as described in chapter 5.6.

With these values, the servo value sent to the NodeMCU was derived according to equation 5.6

$$S = \begin{cases} \theta \text{ if } y > 0 \\ -\theta \text{ if } y < 0 \end{cases} \tag{5.6}$$

where S is the servo value sent to the NodeMCU.
The throttle value was set according the graph in Figure 5.4.

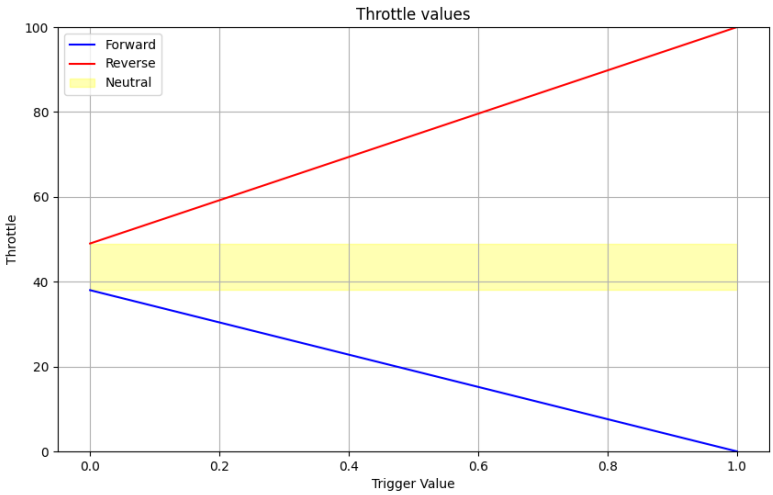


Figure 5.4: *Neutral* if trigger is not pressed or target is already reached. *Forward* if target is in front of car. *Reverse* if target is behind of car.

5.8 Passthrough in Bevy

Bevy's third party mixed reality plugin *bevy_oxr* did not support passthrough at the time of implementation. An implementation of this was therefore done and merged with the plugin. As this was not the goal of the thesis the technical implementation is not detailed here, however, a brief overview is given.

bevy_oxr is written in Rust and utilizes OpenXR, a standard to implement mixed reality features that can run on varying types of AR and VR devices[9]. This is a standard written in C, requiring the Rust implementation to utilize C bindings, i.e. calling C functions from within Rust. OpenXR has an extension for this called *XR_FB_PASSTHROUGH* detailed in their documentation[11]. In short, a *passthrough layer* is created. It acts as a viewport though which the user can see the physical world as captured by the device's cameras.

As the headset was running on Android and developed by Meta, an *Android manifest*, detailing essential info had to be created with properly set permissions and features[1]. Specifically, the *com.oculus.feature.PASSTHROUGH* feature had to be required.

5.9 Visualization

The planned visualization seen in Figure 3.2, included a neutral color blue square representing the car's location. Three orange squares were then placed and rotated along the circle tracing out the car's imminent trajectory. Any number of orange squares could have been added, however, not enough time was available to test scenarios with varying amounts. An arbitrary choice of three squares was therefore made. Two red lines would also indicate the maximum possible steering angle. Lastly, a black line with a length corresponding to the amount the throttle trigger's status would extend in the car's travel direction. For further discussion on choice of visualization and alternatives, see chapter 7.5.

The RC car utilized the Ackermann steering model. This model solves the problem of the inner and outer wheels needing to trace out circles of different radii in a turn. This ensures that all wheels have traction throughout when turning. Had the car been using the bicycle model instead, where both wheels trace out circles of the same radius, one of the wheels would lose traction, making the steering less predictable. This meant that the visualized angle could be computed according to either the Ackermann model, resulting in two angles, or the bicycle model, resulting in one angle.

The angles for Ackermann steering can be computed according to equation 5.7[28]

$$\cot(\theta_o) - \cot(\theta_i) = \frac{W}{L} \tag{5.7}$$

where θ_i and θ_o are the steering angles of the inner and outer wheels, respectively, W is the distance between the wheels on the same axle, and L is the wheelbase of the vehicle.

Furthermore, the angle using the bicycle model can be computed according to equation 5.8[34]

$$\tan(\theta) = \frac{L}{R} \tag{5.8}$$

where θ is the steering angle, L is the wheelbase and R is the turning radius. The bicycle model is a simplification of the Ackermann model.

For this visualization, the bicycle model was implemented, inferring that there would be some difference in the visualized trajectory and the actual trajectory. The RC car used had a maximum steering angle of about 25°. Further, its wheelbase relative to the distance between wheels along the same axle was long compared to other vehicles. This meant that any errors caused by the mismatch of steering model would remain small.

When implementing the visualizations, the car's location, $\vec{p_{\text{car}}}$ direction, $\hat{v_{\text{car\_direction}}}$, desired angle, θ, *throttle* value, the target's location, and distance to the target were known. Creating the blue square as seen in Figure 3.2c could therefore be done by spawning a blue square plane in Bevy at the location and orientation the same as the car's location and direction.

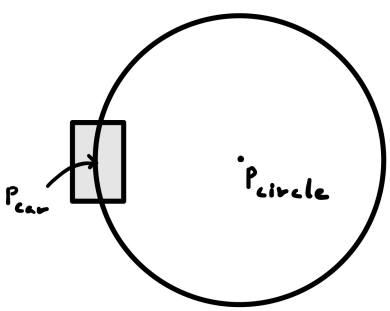
The locations and orientations of the orange squares were not given. Initially, they were created on top of the blue square. The car's right vector was then computed by

$$\hat{v_{\text{right}}} = \hat{v_{\text{car\_direction}}} \times \hat{n} \tag{5.9}$$

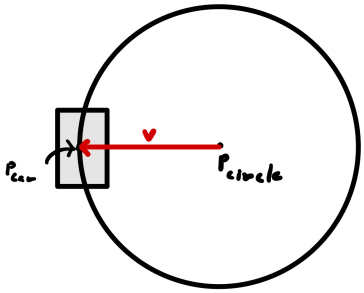
where $v_{\text{car\_}\hat{\text{direction}}}$ is the normalized direction of the car and $\hat{n}$ is the normal of the floor. The radius R of the turn was then computed by

$$R = \frac{L}{\tan(\theta)} \tag{5.10}$$

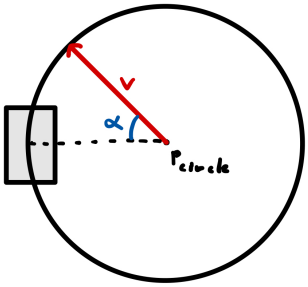
where L was the car's wheelbase. The radius was used to scale the right vector of the car. Depending on whether the target was in front of the car and if it steered left or right, the center of a circle, $p_{\vec{circle}}$ could be computed by either summing or subtracting the scaled right vector from the car's location. The circle's arc would trace out the path that the car would travel given the current angle as seen in Figure 5.5.



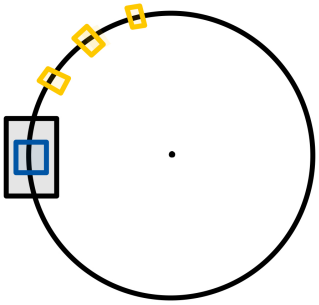
(a) Circle arc representing trajectory of car.



(b) Vector v from circle's origin to car's location.



(c) Rotated vector v.



(d) Orange squares placed along circle arc. Note that the circle was not visible to the user.

Figure 5.5: Circle arc representing trajectory of car.

A vector, $\vec{v}$, from the circle's position to the car's location $p_{\vec{\text{car}}}$ was then created by

$$\vec{v} = p_{\vec{\text{car}}} - p_{\vec{\text{circle}}} \tag{5.11}$$

as seen in Figure 5.5b.

Vector $\vec{v}$ was then rotated around the circle's origin by some angle $\alpha = i \cdot \phi$ where i was the current square being rotated and ϕ an arbitrary angle. Angle α was used to construct a quaternion, $\mathcal{Q}$, for the rotation around the y-axis according to equation5.12.

$$\mathcal{Q} = (\cos\frac{\alpha}{2}, 0, \sin\frac{\alpha}{2}, 0) \tag{5.12}$$

The rotation of $\vec{v}$ was then applied by

$$\vec{v} = \mathcal{Q} \cdot \vec{v} \tag{5.13}$$

as seen in Figure 5.5c.

The endpoint of vector $\vec{v}$ was now positioned along the circle's arc and could thus be used to set the square's location. Further, applying the same rotation to the forward vector of the blue square's directional vector resulted in the desired direction of the current orange square. This is illustrated in Figure 5.5d.

The maximum angle and throttle visualizations were implemented using Bevy's gizmo plugin. This allowed lines to be drawn by specifying starting and and points. The starting point of these lines was given by the car's location $\vec{p_{\text{car}}}$. The endpoints of the angles were then computed by rotating the forward vector of the car around the y-axis using the maximum angle of 25°. Thus, rotations were applied twice with -25°and +25°. The same logic as described in equation 5.6 and Figure 5.3 was applied. The throttle's endpoint was computed by scaling the car's forward vector with the throttle's value multiplying it by -1 in case the car was reversing.

At this point, the fully implemented visualization looked as illustrated in Figure 5.6.

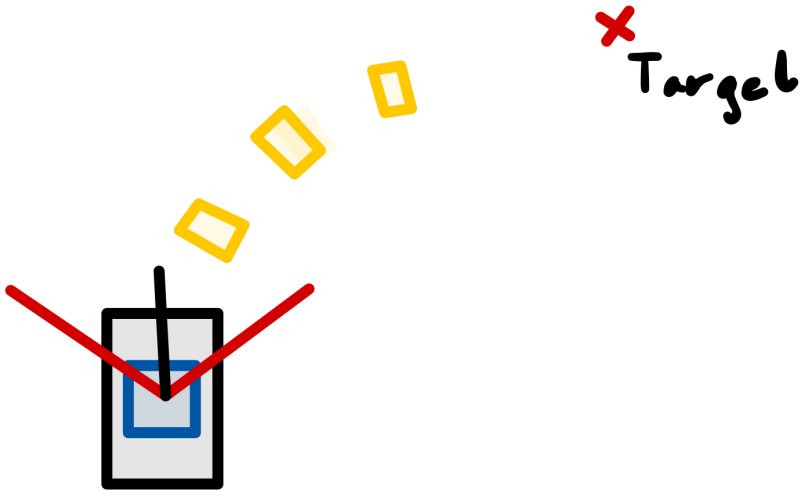


Figure 5.6: Complete illustration of implemented visualization.

5.10 Result of implementation

After all individual components of the system were developed and integrated into the complete system, the visualization looked as shown in Figure 5.7. As the resolution of the screen captures from within the headset was low, they have been upscaled using free online artificial intelligence tools to better convey the result of the implementation. For the original low resolution images, see appendix B and Figure B.1.

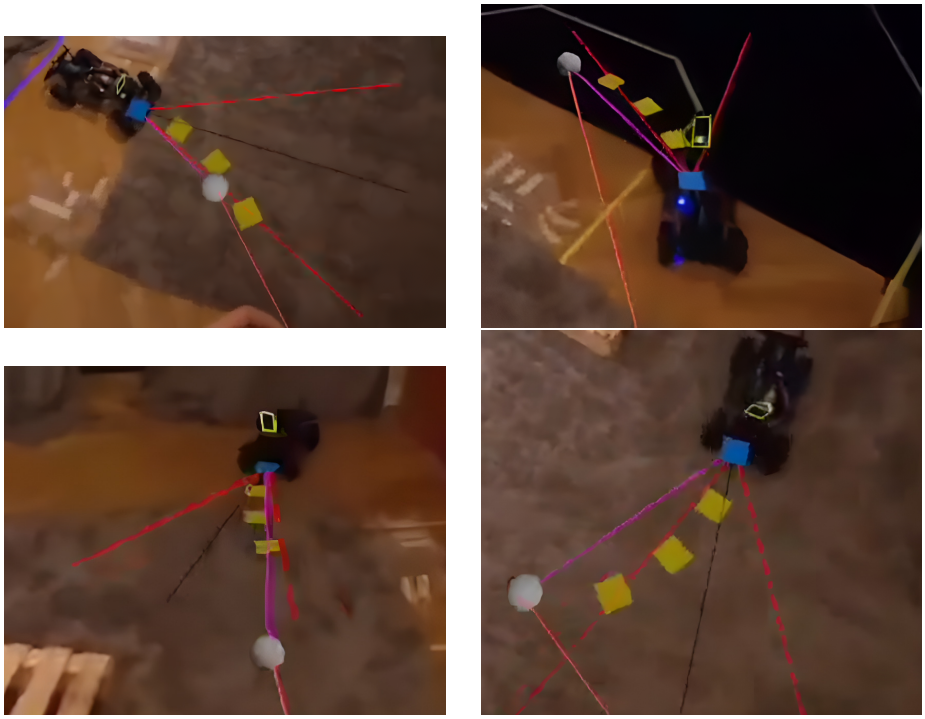


Figure 5.7: Visualization after finished implementation.

Chapter 6

User studies

Throughout the design and technical implementation of the system, knowledge of system capacities and limitations was gained. This was used to design a method for evaluating the implemented system's usefulness. This entailed planning the design of user tests and conducting user tests. Specifically, an obstacle course was constructed, and three different questionnaires would be developed. These questionnaires where SART, SUS and TLX questionnaires. The purpose of these tests was to result in data which could be used to answer the research questions in chapter 1.3.

6.1 Conducting user studies

With a course set up and questionnaires prepared, users participating in the tests were informed about the tasks they would perform. These were the three scenarios:

1. Navigate RC car with a joystick without visualizations
2. Navigate RC car with virtual controls without visualizations
3. Navigate RC car with virtual controls with visualizations

After all three tasks were completed, the subject was asked to respond to three questionnaires meant to provide useful data about his experience, one for each task.

In order to avoid a user getting used to controls, which could impact their experience in the following tasks, randomization was implemented. This means that the order in which each task was performed was changed for each user.

6.2 User participants

Finding user participants was mainly done by finding people on the street that were willing to participate. This resulted in a a somewhat mixed group of people from different backgrounds. Out of nine people, eight were Swedish and one was French. The questionnaire was developed in English, making this possible. However, it also meant that it was difficult to find young and very old

participants as the language barrier would likely cause problems when answering questionnaires. Most people were therefore between twenty to 50 years old.

6.3 Construction of obstacle course for tests

In order to test the system, an obstacle course, in which the user would navigate the RC car through, had to be developed. The purpose of the course was to test the system's usefulness. This would by done by subjective assessments with surveying methods. The time spent to complete the course was thus not of much importance. Instead, an emphasis on the level of confidence in controlling the car would be emphasised. A short obstacle course was therefore developed as seen in Figure 6.1 with the path that the user were to navigate along is outlined.

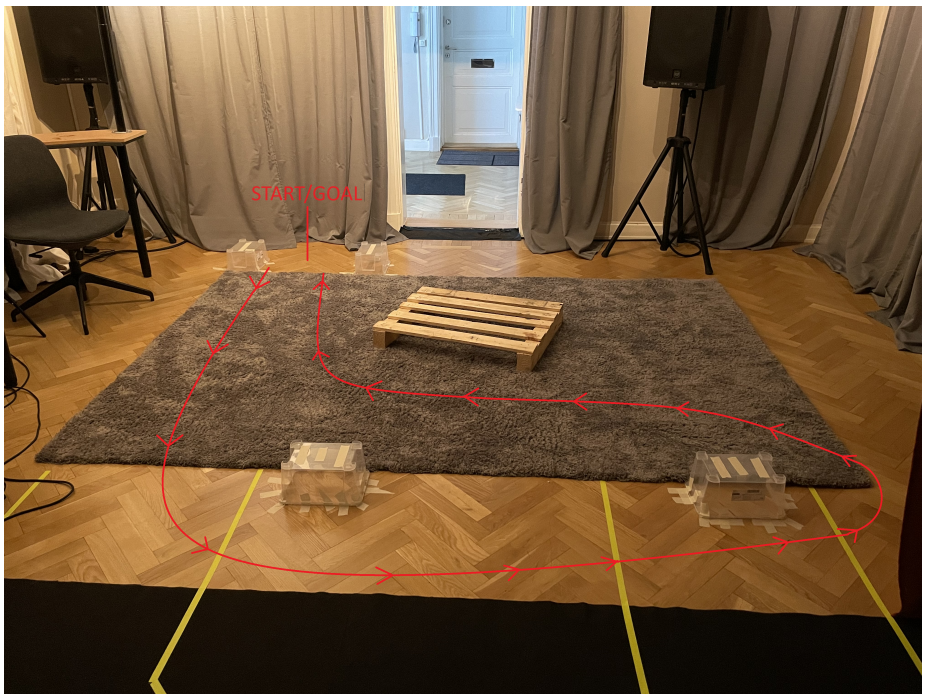


Figure 6.1: Obstacle course with outlined path from start to finish.

The RC car's turning radius was very large. This meant that the tight turns of the course would add difficulty by forcing the user to reverse the vehicle. Despite the obstacle course being short, there were still significant challenges presented.

6.4 Situational Awareness Rating Technique, SART

Situation awareness, SA, a concept introduced by Endsley[23], examines how well a person understands what is going on in an environment and how it changes with time and other factors. The idea is that SA, can be thought of in three

levels. These are *perception*, *comprehension*, and *projection* as seen in Figure 6.2.

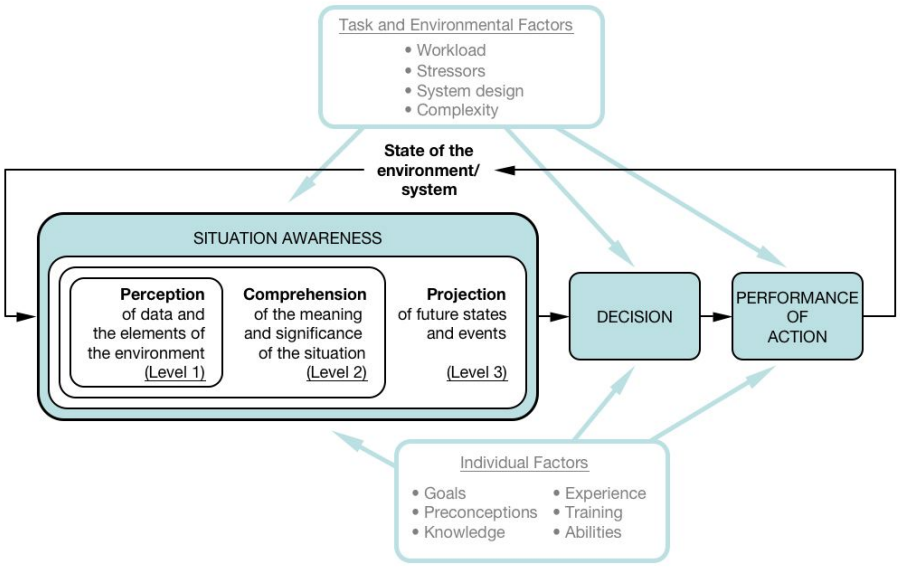


Figure 6.2: Levels of situation awareness.

According to this model, a *prediction* level of SA allows the user to anticipate what will happen given the situation. A *comprehension* level would only allow the user to interpret perceived information to understand the situation, but not anticipate future events based on it. A *perception* level would only allow the user to notice and recognize information cues in the environment, although would not fully be able to comprehend the situation based on these cues.

There are objective, subjective and semi-objective ways to measure SA.

Semi-objective ways are based on comparing a subject user's subjective thoughts to an objective truth. SA global assessment technique, SAGAT[22], is one such technique that involves querying the person by interrupting the task being evaluated.

Another semi-objective method is based on real-time probing, as proposed by Debra G. Jones and Mica R. Endsley [17]. Instead of interrupting the person during the task, the person is verbally queried during the execution of the task without interruptions.

Another semi-objective method, WOMBAT or HUPEX, used for selecting suitable pilots for pilot training, is instead based on querying the subject after the task was performed[14].

A commonly used subjective measure of SA is based on having the person perform the task and then answer a questionnaire, as opposed to before or during the task. One method for this is called the Situation Awareness Rating Technique, SART[31]. This involves having the user perform the evaluated task and then, upon completion, answering a questionnaire. The questionnaire commonly includes ten questions across three dimensions. Each question is answered on a likert scale in range 1-7. The first three for demand, the following three for supply and the last four for understanding.

The demand dimension answers how much or how many tasks there were that required the participant's attention.

The supply dimension answers how well the user was able to fulfill the demands of the system.

Further, the understanding dimension answers how well the user understood the environment and the situation. The situational awareness is then calculated according to equation 6.4 as proposed by Taylor [31],

$$d_{\text{Situation\_Awareness}} = d_{\text{Understanding}} - (d_{\text{Demand}} - d_{\text{Supply}}) \tag{6.1}$$

where $d_{\text{Situation\_Awareness}}$ is the final score and $d_{\text{Understanding}}$, d_{Demand} and d_{Supply} are the respective sums for each dimension.

Commonly, SART is preferred over the semi-objective methods due to its simplicity. Further research by Mica R. Endsley, Stephen J. Selcon, Thomas D. Hardiman and Darryl G. Croft have also shown that a user's perceived situational awareness can be of more importance than the true situation awareness[18].

The developed questions can be seen on page 3 in Appendix A.

6.5 System usability score, SUS

The System Usability Scale, SUS, was introduced by Brooke in 1986[12]. It is a method for evaluating usability of a general system. It uses ten statements that a subject participant grades on a Likert scale of 1 to 5 or 7 based on their level of agreement with the statement.

The ten statements are the same, regardless of system, making SUS applicable to many systems. This allows comparisons across varying systems, but lacks the ability to adapt to specifics of any given system. The statements used can be seen on pages 4 and 5 in the Appendix A.

After answering the SUS questionnaire, a final score is calculated. Questions 1, 3, 5, 7 and 9 contribute with the grade's value minus 1 and questions 2, 4, 6, 8 and 10 contribute with 5 minus the grade's value. The accumulated value is then multiplied by 2.5 as can be seen in equation 6.2,

$$\text{SUS Score} = 2.5 \left(\sum_{i \in \{1,3,5,7,9\}} (G_i - 1) + \sum_{i \in \{2,4,6,8,10\}} (5 - G_i) \right) \tag{6.2}$$

where G is the grade value for the statement with the number i. This results in a score between 0 and 100.

This scoring technique allows for quick evaluations of a system's usefulness with few participants, and rapid iteration at the cost of specificity in regard to the evaluated system.

6.6 Task load index, TLX

The Task Load Index, TLX, is a tool developed in 1988 by Hart and Staveland at NASA's Human Performance Group[20]. It is a subjective method for assessing perceived cognitive workload and gauging a system's usability. TLX lets users to rate the demand for a task across six dimensions: mental demand, physical

demand, temporal demand, frustration level, effort, and performance. These dimensions are rated on a scale from low to high, with 21 levels.

Scores can be directly analyzed, commonly referred to as RAW-TLX. They can also be weighed on the basis of what demands are deemed most important for the given workload. This is done by letting the participant pick the most relevant demand across the 15 pairwise combinations of the various demands. The number of times a dimension is chosen is used to weigh, by averaging or summing, the raw scores to get a final score. This weighing scheme is called NASA-TLX and was the final version of TLX that Hart and Staveland developed.

For this project, RAW-TLX was chosen over NASA-TLX. This was done in order to reduce the amount of time and effort spent on user surveying, thus lessening the burden placed upon the participants. This would allow more studies to be done in the same amount of time. The RAW-TLX's lesser data would also be easier to analyze, although at the cost of metrics that could perhaps have been useful.

The questionnaire for the study, with the questions from Hart and Staveland's questionnaire, [20] can be seen on page 6 of Appendix A.

6.7 Variance analysis

In order to determine the results of the questionnaires, an analysis of variance, ANOVA, would also be conducted. ANOVA is used to determine if there is a statistical difference between three or more groups. In this case, there were three driving scenarios, making ANOVA a feasible method of analysis[5].

ANOVA assesses the null hypothesis that the means of different groups are equal, against the alternative hypothesis that at least one group mean is different. This is done by comparing the variances within the groups to the variances between the groups. ANOVA results in an F-statistic, which is calculated as

$$F = \frac{\mathrm{MS}_{\mathrm{between}}}{\mathrm{MS}_{\mathrm{within}}} \tag{6.3}$$

where $\mathrm{MS}_{\mathrm{between}}$ is the mean square between the groups, and $\mathrm{MS}_{\mathrm{within}}$ is the mean square within the groups. These are calculated as

$$\mathrm{MS}_{\mathrm{between}} = \frac{\mathrm{SS}_{\mathrm{between}}}{\mathrm{df}_{\mathrm{between}}} \tag{6.4}$$

$$\mathrm{MS}_{\mathrm{within}} = \frac{\mathrm{SS}_{\mathrm{within}}}{\mathrm{df}_{\mathrm{within}}} \tag{6.5}$$

where $\mathrm{SS}_{\mathrm{between}}$ is the sum of squares between the groups, $\mathrm{SS}_{\mathrm{within}}$ is the sum of squares within the groups, $\mathrm{df}_{\mathrm{between}}$ is the degrees of freedom between the groups, and $\mathrm{df}_{\mathrm{within}}$ is the degrees of freedom within the groups. These are given by

$$\mathrm{SS}_{\mathrm{between}} = \sum_{i=1}^{k} n_i(\bar{X}_i - \bar{X})^2 \tag{6.6}$$

$$\mathrm{SS}_{\text{within}} = \sum_{i=1}^{k} \sum_{j=1}^{n_i} (X_{ij} - \bar{X}_i)^2 \tag{6.7}$$

$$\mathrm{df}_{\text{between}} = k - 1 \tag{6.8}$$

$$\mathrm{df}_{\text{within}} = N - k \tag{6.9}$$

where k is the number of groups, n_i is the number of observations in the i-th group, $\bar{X}_i$ is the mean of the i-th group, $\bar{X}$ is the overall mean, X_{ij} is the j-th observation in the i-th group, and N is the total number of observations.

If the F-statistic is greater than the critical value from the F-distribution at a given significance level, the null hypothesis is rejected, indicating that there are significant differences between the group means. The F-critical value can be found using the inverse cumulative distribution function of the F-distribution:

$$F_{\text{critical}} = F^{-1}(1 - \alpha, \mathrm{df}_{\text{between}}, \mathrm{df}_{\text{within}}) \tag{6.10}$$

The p-value togehter with the F-statistic helps determine the significance of the results. The p-value is the probability that the observed data would occur if the null hypothesis were true. It is calculated from the F-distribution with the appropriate degrees of freedom. The p-value is compared to a significance level α that is commonly 0.05, to assess the null-hypothesis. This is done according to

- If $p \leq \alpha$, the null hypothesis is rejected, indicating that there is a statistically significant difference between the group means.
- If $p > \alpha$, the null hypothesis is not rejected, indicating that there is no statistically significant difference between the group means.

6.8 Results

The scores from the user tests were computed for each user. The average of these scores are seen in table 6.1 and the standard deviations of these are seen in table 6.2.

Table 6.1: Average scores

Average scores	**SART**	**SUS**	**RAW-TLX**
Joystick	18.56	61.39	63
Virtual control without visualization	24.33	78.06	55.11
Virtual control with visualization	25.67	77.50	54.56

Table 6.2: Standard deviation

Standard deviation	**SART**	**SUS**	**RAW-TLX**
Joystick	0.42	0.57	3.62
Virtual control without visualization	0.76	1.15	3.52
Virtual control with visualization	0.8	1.12	3.66

The average score for each question of each questionnaire and the test scenario are shown in Figures 6.3, 6.4 and 6.5, respectively.

The F-statistic and p-values were also calculated for SART and SUS using ANOVA, in order to determine the significance of the scores. This was not done for the TLX results as those values are not meant to be interpreted as mean values. These values are seen in table 6.3. The F critical value for nine participants and three groups was also calculated, resulting in a F critical value of 3.40 at the significance level 0.05.

Table 6.3: ANOVA analysis

ANOVA analysis	**F-statistic**	**P-value**
SART	3.66	0.039
SUS	0.14	0.87

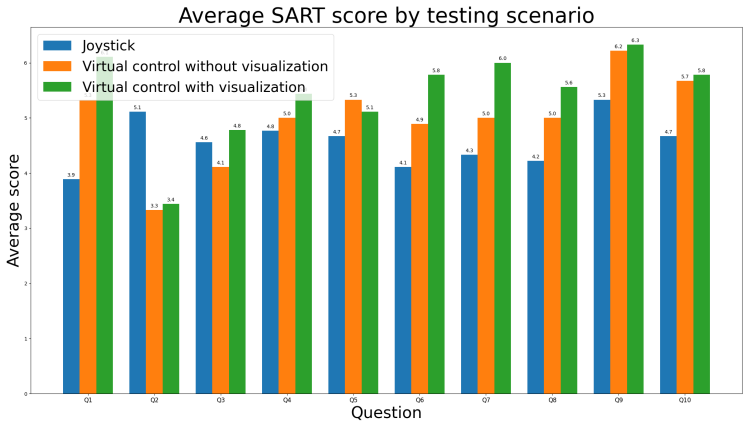


Figure 6.3: Bar graph of the average score per question in the SART questionnaire.

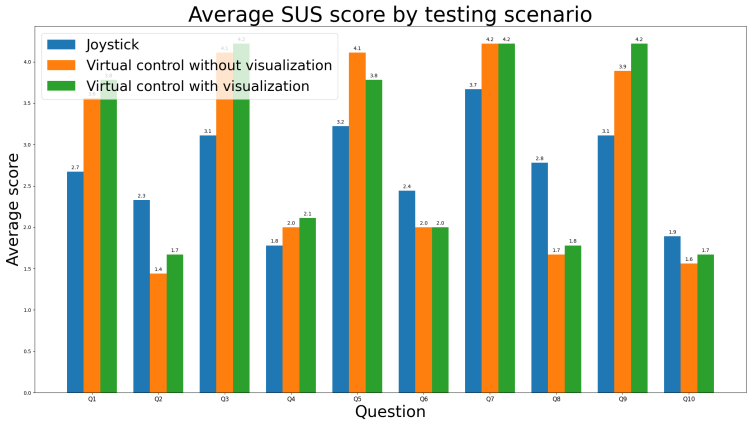


Figure 6.4: Bar graph of the average score per question in the SUS questionnaire.

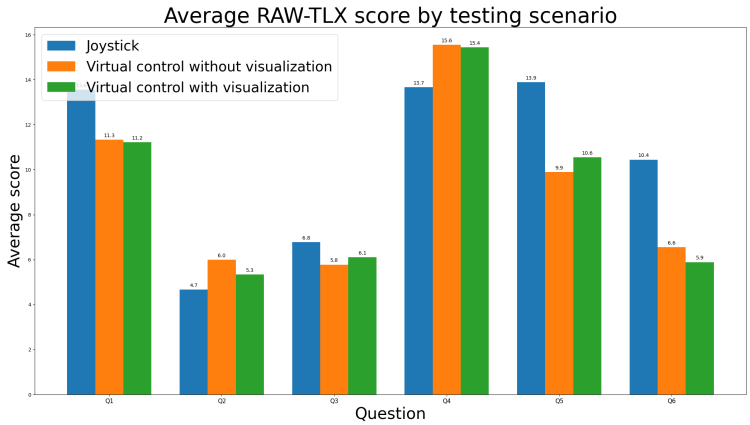


Figure 6.5: Bar graph of the average score per question in the TLX questionnaire.

Chapter 7

Discussion

The results of the gathered data from questionnaires are discussed and the validity of these in regards to subjectivity and language barriers are discussed. There is also discussion about the source of the thesis and alternative methods that could have been chosen to answer the research question in chapter 1.3. The purpose of this was to aid the the reader in understanding the outcome and validity of the results from the user tests in chapter 6.8.

7.1 Method

The method implemented in this thesis utilized the Bevy engine and Rust, mainly due to Voysys, the company hosting the research, expressing a preference for Rust, a sentiment also shared by the student conducting the study. Alternative engines could also have been used, such as *Unity* or *Unreal Engine*, both commonly used for MR applications. Opting for one of those would have largely changed how the actual implementation of the system would be done. Perhaps these engines could have provided already built visualizations elements that could have been easily integrated into the visualization. However, familiarity with these engines was lacking, especially in regards to how transfer of data between the car and headset would be achieved.

In regards of how virtual controls can be useful for directing a car or not, there are many types of virtual controls. In this case, a laser pointer-like approach was chosen, as this type of input is commonly used in VR and MR headsets already. Alternative virtual controls could also have been tested. For example, the *Apple Vision Pro* does not have physical controllers, but instead relies on hand-tracking and gestures. This type of input was also possible on the *Meta Quest 3* headset, but is not a common input interface across other headsets. Changing the type of virtual control would of course also change how the implementation of the system was made.

Opting for other headsets would perhaps allow more access to the headset camera's data directly. This would have made it possible to utilize fiducial markers for tracking by placing fiducial markers on the car and in its environment and track their locations, rather than retrieving the spatial data from the *Quest Pro* controller. An alternative approach that could have been explored was to research if the Vulkan frames could have been intercepted, and thus bypassing

Meta's privacy policies. If this would have been possible, the estimated time needed to implement this would have been too long, given that the goal of the thesis was not how to track the RC car. Thus, retrieving spatial data from the controller directly was preferred in this case.

Using other headsets could also infer that other operating systems would be targeted. For example, if the *Apple Vision Pro* had been used instead, their *Vision OS* would need to be targeted instead of Android's APK for *Android OS*.

7.2 Visualization alternatives

Many different types of visualizations could have been implemented. Adding more varying visualizations would have been interesting. However, given the difficulty of finding participants for evaluation, extending the time of testing and answering questionnaires by adding more tests would escalate the issue of finding participants. Therefore, only one visualization was tested. The chosen visualization that would be evaluated would be one that was likely to reduce cognitive load. Therefore, a simple, yet intuitive visualization was prioritized over complex ones.

The visualization implemented could have been made using arrows or lines instead. Using strictly 2D elements such as lines may be less feasible for other applications if visualizations were to be used in other applications such as for airplane or submarine trajectories. By using a 2D-plane meshes, introducing a third dimension to the visualization could easily be done by extending it to a cube instead. Meanwhile, the shape can be kept simple in order to avoid unnecessary complexity that may contribute to excessive cognitive load. Using lines can also be done in three dimensions, although may infer difficulties in depth perception. Using mesh based visualizations offered the benefit of allowing shading algorithms to be applied that can provide depth cues.

It would also have been interesting to add more elements, such as telemetry data. For example, the RC car's battery level, speed, angle, and more could have been shown to the user. Further, lines could have been drawn on top of the squares used. Although adding more visual elements would have been interesting, adding more elements would likely make it more difficult to interpret the results. If there are too many elements in a visualization, it is likely hard to deduce which element contributed to what aspect in the usefulness of the entire visualization. It would therefore be of interest to remove each element in the implemented visualization one at a time when performing the study to determine what aspect was of most benefit.

In a scenario where trajectories of many vehicles, such as a warehouse with many autonomous forklifts, at once may be needed, reducing the amount of visualizations for each vehicle would likely be beneficial, as too much data at once may overwhelm the user. Therefore, it would be interesting to explore if there is an optimum where the amount of visual elements do not overwhelm the user while still being useful for many vehicles at once.

7.3 Data, Subjectivity and Language

The user tests conducted in this thesis resulted in purely subjective data. Therefore, it would be of interest to conduct further tests to gather more data, which would minimize any single particant's impact on the results. The developed SART questions could also have been developed differently, while still following the guidelines of SART questions. This may have resulted in different results. However, the guidelines of using *demand*, *supply* and *understanding* type of questions were followed. Since there are no standardized questions for SART, questions specific to this study were developed. Assessing the relevance of these questions compared to others is challenging because both are subjective. Additionally, there is no objective metric available to compare the subjective scores.

All of the questionnaire questions used were written in English, while most, albeit not all, participants' native languages were Swedish. Therefore, a language barrier could have impacted their answers in the study. This should be kept in mind when interpreting the result. However, participants were free to ask for translations if needed, and as long as the essence of each question was conveyed, it likely would not impact the results when looking at the relative scores of each scenario. Thus, analyzing the scenarios' scores relative to each other is likely more beneficial than looking at each scenarios' score alone.

7.4 Results

As seen in table 6.1 and 6.2, the SART score for the joystick scenario was remarkably lower than for the other scenarios, while the low standard deviation indicated a high level of agreement of the participants. This indicates that SA was lower when only a joystick was used to control the car. For both virtual control scenarios, the scores were remarkably higher, although the added visualizations appear to provide slightly higher SA. The p-value of the SART test, 0.039, is lower than 0.05. This means that the difference between the means is significant and that the null-hypothesis is rejected in this case. This rejection of the null-hypothesis is also supported by the F-statistic of 3.67 being higher than the F-critical value of 3.40anova.

As for the SUS scores, the joystick scenario seems to be less usable as compared to both other scenarios. Interestingly, the scenario without visualizations is slightly higher than the scenario with visualizations. However, given the slightly higher standard deviation in the scenario without visualizations, they may be considered equal. Adding these visualizations therefore does not seem to indicate a higher usability of the system. Commonly, SUS scores of 68 are considered average usability and anything above this is considered okay[10]. Although this is very general, this implies that the joystick scenario is not very usable, whereas both other scenarios are above average usability. Note, however, that these values are very general and can vary dependent on systems. The p-value of the SUS test, 0.87, is also higher than 0.05, meaning that the difference between the mean values are not significant. The F-statistic is also lower than the F-critical level, indicating that the variance between samples in the same scenario were high relative to the variance between the mean values for each scenario, and that the null-hypothesis is not rejected. It is thus hard to draw any conclusions from these scores alone[6]. It would thus be of interest

to increase the number of participants in the test in further studies.

The RAW-TLX scores are not meant to be used as averages. Instead, a look at the values in Figure 6.5 is made for each question. The first question implies that the mental demand required was higher in the joystick scenario as opposed to the other scenarios which were rather similar.

The second question implies that the physical demand was higher when using virtual controls, which makes sense as pointing with the entire arm was required to control the car. The joystick scenario would only require the thumb to move, requiring less physical demand.

The third question asked about perceived time pressure. All scenarios have similar responses, which makes sense as participants were informed that they should not feel any time pressure during the tests. This question is therefore not very applicable to these tasks.

The fourth question asked about how successful they felt during the task. Participants felt less successful when using the joystick compared to the other scenarios. One interpretation of this is that they may have crashed more during this task. Another interpretation could be that they initially imagined it would be easier than it was, as joysticks are more common in everyday tasks than virtual controllers.

Question 5 asked how much effort was required for the task. The joystick seemed to require more effort than the other scenarios, which may be related to the same reasoning discussed for the fourth question.

Question 6 asked how frustrating the task was. The joystick scenarios was notably much more frustrating than both other scenarios. However, in this case, adding visualizations to virtual controllers seemed to make the experience less frustrating than when using virtual controllers without visualizations.

Overall, the RAW-TLX scores, with the temporal demand disregarded, it appears that the chosen visualization did reduce cognitive load when performing the task of driving the RC car. The visualization thus achieved the goal of minimizing cognitive load, which was considered an important factor in reducing occupational accidents as concluded by [25].

Overall, the results show that users preferred the virtual controllers more than the joystick. Interestingly, there was not much difference between the two virtual controller scenarios other than perceived SA. However, the results are subjective metrics. Further, given that only nine participants were found, it is possible that there were more tests where the scenario *virtual controls with visualization* was tested before the *virtual controls without visualization* scenario. This could have allowed users to get used to the RC car's behavior by the time the visualizations were toggled off.

Given this, it does seem like virtual controls in MR can improve the user experience in relation to SA and the usability of the system when it comes to directing a car. However, given the similar performance between *virtual controllers without visualization* and *virtual controllers with visualization*, user studies show that there was some, albeit little, benefit in relation to SA when using visualizations.

7.5 Future work

Future research that would be interesting is to explore the headset design's impact on the wearer's SA. This study used the *Quest 3* headset, although it is unknown how much the choice of headset currently impacts the SA. Perhaps a lighter headset with the same testing scenarios would yield different results as a consequence to comfortability of the headset itself.

There is a lot of development being done on display technologies, processor efficiency, sensory tracking and batteries. With advancement in these technologies, it is possible that future headsets will be lighter, have longer battery life, have faster screens and better tracking. This could cause future headsets to have a significant impact on SA. Future research may therefore involve how these factors impact SA.

7.6 Source Criticism

The sources in this report are mostly peer-reviewed articles published between 1988 and 2022. The older of these cover more fundamental aspects that have not changed much over time, such as the concepts of SA, SUS and TLX.

Sources covering MR were mostly newer peer reviewed articles. In regards to these, it is important to consider that the field of MR has garnered interest in recent years as Meta have made significant investments into the field. Therefore, much research is currently done in this field, that may further validate or contradict these sources. Given this, it is of essence to keep up to date with current research.

Some sources also point to the current documentation of hardware and software specifications, such as those of Bevy engine's plugins. These may change or be moved over time, although were relevant at the time of implementation.

7.7 Ethics and societal aspects

The integration of MR headsets in professional environments brings forth ethical and societal concerns that should be addressed.

Wearing today's MR headsets for extended periods of time can lead to physical discomfort, such as neck strain caused by the weight of the headsets. Also, the heat emitted during use can subject the user to unreasonable levels of heat. These issues should be addressed by the employer. However, the motivation to this thesis was to evaluate if MR technology for control and visualization of trajectory would be useful when these technological challenges are addressed. Thus, it would be of importance for employers to not start implementing the technology before these challenges have been addressed.

Some user of MR headsets also experience motion sickness. This is caused by the sensory discrepancies between the physical and virtual world. As concluded in *Multi robots interactive control using mixed reality*[26], this could be caused by inaccurate spatial mapping. It could also be caused by high latency, low screen refresh rates, interpupillary distance discrepancy, and low resolution[8]. Therefore, these technical challenges would need to be addressed. Some headsets allow modification of some these aspects. For example, the distance between the screens can be adjusted to address users different interpupillary distances. Some

headsets, such as the Apple Vision Pro, are even modified to the specific user, even providing options to have lens prescriptions to address vision problems.

Many headsets are mainly developed by men due to the male dominance in the technical fields. This has caused many headsets to be, either unintentionally or intentionally, optimized for men. A study by Stanney, Fidopiastis and Foster[29] concluded that women experience significantly more motion sickness than men, with the main cause being that women were unable to achieve a good interpupillary distance fit.

While some users may not experience discomfort in MR, some experience a lot of discomfort. This can cause a shift in what people would be able to work at workplaces using MR as part of everyday tasks. For example, women would more likely need to find jobs at workplaces where they are not required to use MR. Employers may not intend for this to happen when using MR at their workplaces. Their intention may come from other motivations, such as productivity. The question of who is at fault for this issue arises. Is it the employer, MR developers, headset developers? A discussion around this issue at a societal level may be needed to address this issue.

There can also be a displacement of workers when using MR, as employees that find the technology more difficult than others may have to find other jobs. The rise of autonomous and RC vehicles could further displace jobs. A driver would no longer be needed for autonomous vehicles, and RC vehicles could be remotely operated from anywhere. This would allow an employer to hire remote operators where labor is cheaper, thus removing many jobs in the workplace's immediate location.

Further, if accidents occur with a user of MR at a workplace due to MR tools, who is at fault also needs to be discussed. For example, a collision with a worker and RC vehicle could happen if visualizations or virtual controls are not correct. Is it the workers fault for trusting the visualizations, the employers for using tools that can be wrong, or the developer of the MR tool? Perhaps the workplace environment was not lit well enough to allow accurate spatial mapping. Discussions around these issues also need to be had.

Chapter 8

Conclusion

The aim of this thesis was to develop and evaluate the usefulness of a MR system for controlling a RC car with virtual controllers as opposed to joystick controllers. The thesis also aimed to visualize the car's imminent trajectory and determine if such visualization provided benefits in the HMI between the user and the RC car. This type of system was implemented in Bevy and the Rust programming language. After implementation, an evaluation of three cases, *joystick controller*, *virtual controller without visualization* and *virtual controller with visualization* was conducted by using the subjective methods SART, SUS and RAW-TLX. The goal of this was to answer the research questions in chapter 1.3 also seen below.

1. *How can a MR headset be used to visualize the imminent trajectory of an RC car in mixed reality?*
2. *Does visualization of the RC car's trajectory contribute to an improved perceived situation awareness, SA?*
3. *How do virtual controls in a MR environment, as compared to traditional joystick controllers, perform in terms of user experience, situational awareness and efficiency for directing a RC car?*

As discussed in chapter 7.1 and 7.5, there are other ways that an implementation of a system that would fulfill the requirements could be implemented. However, one method, using Bevy and the Rust programming language has been proposed. This provides one answer to question 1, but other options may also be feasible. As for question 3, the evaluation's result showed that there was a notable benefit in using virtual controllers as opposed to joystick controllers. Further, as for question 2, There was some, albeit little benefit to adding visualizations in regards to SA, although SUS and RAW-TLX scores did not indicate any remarkable differences between the scenario *virtual controller without visualization* and *virtual controller with visualization*. It is noted that the evaluation is subjective, and that a higher number of participants would have been useful. The similarity in SUS and RAW-TLX may be caused by participants getting used to how the car behaves when using visualizations, thus making the case without visualizations easier for the participant. This was an expressed feeling of some participants who performed the task with visualizations before toggling them off. This was unfortunately not an explicit question in the survey, and

was thus not recorded. For similar works in the future, it would be beneficial to ask whether if the order of the test impacted their performance.

Appendix A

User study

Participation Survey

Participation information: Participation in this study is voluntary. Any information that can be used to identify individuals will be anonymized to protect the participants' personal integrity. Personal information is used only to avoid surveying the same participant multiple times and to compare previous experiences relevant to the testing. By filling out this form you agree to participate in the study.

The motivation for the study: The motivation behind this study is to evaluate the usefulness of a mixed reality system for controlling a remote control, RC, car.

The testing scenario: The test in conducted three times, with analog controls and no mixed reality headset, with virtual controls without added visualizations and with virtual controls and added visualizations. Participants are expected to perform each different scenario once, each with the same goal. The goal is to navigate the car along the path outlined in Figure 1. There is no set time to achieve this task, although a maximum of five minutes is allowed in order to limit the total time of the test. This should be a great deal of time and is not intended to stress the participant.

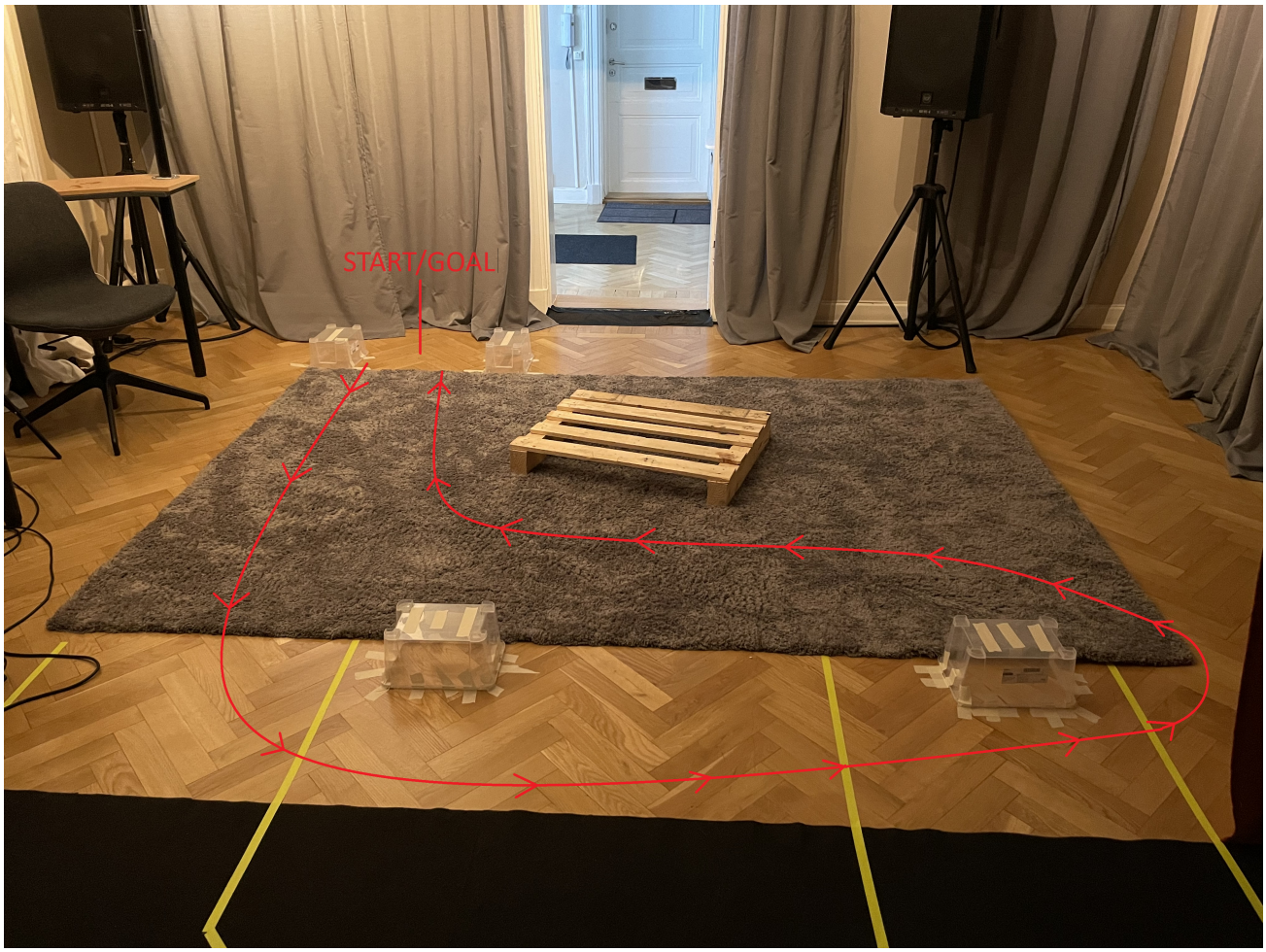


Figure 1: Track with path outlining intended path.

Which scenario was tested?

☐Joystick controls + Without visualization
☐Virtual controls + Without visualization
☐Virtual controls + With visualization

About you

1. First name: _____
2. Last name: _____
3. Age: _____
4. Gender:
   ☐ Man ☐ Woman
5. Have you driven RC cars before?
   ☐ Yes, I feel confident in driving RC cars
   ☐ Yes, but I am not confident in it.
   ☐ No, I have very little or no experience
6. Have you played driving games or used other forms of vehicle simulators before?
   ☐ Yes, quite often
   ☐ Yes, but not often
   ☐ No, I have never used a vehicle simulator or played driving games.
7. Have you used a head-mounted display or head-mounted projector before?
   ☐ Yes, quite often
   ☐ Yes, but not often
   ☐ No, I have never used a head-mounted display or head-mounted projector before

Consent for Participation

Consent: Your participation in this study is voluntary, and by contacting the surveyer at r.hogslatt@gmail.com, you can ask to have your data withdrawn. The data collected during this study will be anonymized to protect your privacy and will only be used for research purposes. **By signing below, you consent to the collection and use of your data as described in this document.**

Signature: _____ **Date:** _____

1 SART Questionnaire

1. Did you feel a sense of control in the situation? Were you able to anticipate where you were headed whilst navigating through the course (low) or not (high)?
low ☐ ☐ ☐ ☐ ☐ ☐ ☐high

2. How much mental effort did you have to put in to control the RC car? Did you have to actively think about how you should control the car (high), or did it feel intuitive (low)?
low ☐ ☐ ☐ ☐ ☐ ☐ ☐high

3. Did the task require most of your attention (high) or only some (low)? Would you have been able to spend attention on anything unrelated to the task at hand?
low ☐ ☐ ☐ ☐ ☐ ☐ ☐high

4. How alert were you during the task? Were you constantly on high alert and feeling ready for activity (high) or was your alertness low (low)?
low ☐ ☐ ☐ ☐ ☐ ☐ ☐high

5. How well could you divide your attention between the RC car and its surroundings? Were you focused on both controlling the RC car and its environment simultaneously (high), or did you have to stop or slow down to look for upcoming obstacles (low)?
low ☐ ☐ ☐ ☐ ☐ ☐ ☐high

6. Did you feel a sense of control of the RC car (high) or not (low)? If the obstacle course would have changed suddenly, would you have felt comfortable in your ability to adapt to the situation?
low ☐ ☐ ☐ ☐ ☐ ☐ ☐high

7. Did you get an understanding of the behavior and trajectory of the RC car (high) or not (low)?
low ☐ ☐ ☐ ☐ ☐ ☐ ☐high

8. Were you able to anticipate the RC car's movements (high) or not (low)?
low ☐ ☐ ☐ ☐ ☐ ☐ ☐high

9. Did you feel like you understood the provided controls well (high) or not (low)? Note: "Controls" refer to either the analog or virtual controls varied during the experiments.
low ☐ ☐ ☐ ☐ ☐ ☐ ☐high

10. Were you able to integrate different sources of information well(high) such as both virtual and physical visual cues during the task, or did these types of information not contribute to your understanding (low)?
low ☐ ☐ ☐ ☐ ☐ ☐ ☐high

2 System Usability Scale

The System Usability Scale, SUS, provides measures the usability of the system. It consists of a 10 item questionnaire with five response options per questions, ranging from *Strongly agree* to *Strongly disagree*. Please circle the option that best represents your experience.

1. I think that I would like to use this system frequently.

1 - Strongly disagree
2
3
4
5 - Strongly agree

2. I found the system unnecessarily complex.

1 - Strongly disagree
2
3
4
5 - Strongly agree

3. I thought the system was easy to use.

1 - Strongly disagree
2
3
4
5 - Strongly agree

4. I think that I would need the support of a technical person to be able to use this system.

1 - Strongly disagree
2
3
4
5 - Strongly agree

5. I found the various functions in this system were well integrated.

1 - Strongly disagree
2
3
4
5 - Strongly agree

6. I thought there was too much inconsistency in this system.

1 - Strongly disagree
2
3
4
5 - Strongly agree

7. I would imagine that most people would learn to use this system very quickly.

1 - Strongly disagree
2
3
4
5 - Strongly agree

8. I found the system very cumbersome to use.

1 - Strongly disagree
2
3
4
5 - Strongly agree

9. I felt very confident using the system.

1 - Strongly disagree
2
3
4
5 - Strongly agree

10. I needed to learn a lot of things before I could get going with this system.

1 - Strongly disagree
2
3
4
5 - Strongly agree

3 Task Load Index

The Task Load Index, TLX, is a widely used tool for assessing the perceived workload of tasks, considering various dimensions that contribute to workload. Rate the following dimensions of the task you just performed from low to high.

1. **Mental Demand:** How much mental and perceptual activity was required? Was the task easy or demanding, simple or complex?

   Low 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 High

2. **Physical Demand:** How much physical activity was required? Was the task easy or demanding, slack or strenuous?

   Low 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 High

3. **Temporal Demand:** How much time pressure did you feel due to the rate or pace at which the tasks or task elements occurred? Was the pace slow and leisurely or rapid and frantic?

   Low 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 High

4. **Performance:** How successful do you think you were in accomplishing the goals of the task set by the experimenter (or yourself)? How satisfied were you with your performance in accomplishing these goals?

   Low 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 High

5. **Effort:** How hard did you have to work to accomplish your level of performance?

   Low 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 High

6. **Frustration Level:** How insecure, discouraged, irritated, stressed, and annoyed versus secure, gratified, content, relaxed, and complacent did you feel during the task?

   Low 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 High

Please circle the number that best reflects your experience for each dimension.

Appendix B

Raw images of implementation

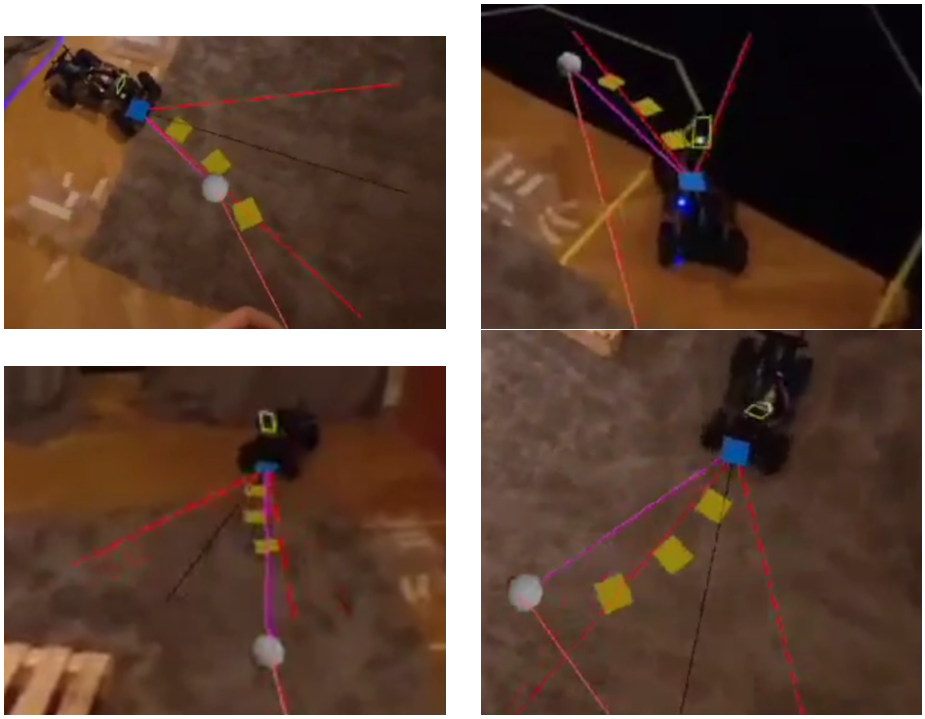


Figure B.1: Non-upscaled result of implemented visualizations.

Bibliography

[1] App manifest overview, howpublished = https://developer.android.com/guide/topics/manifest/manifest-intro, note = Accessed: 2024-05-04.

[2] Ar vs vr vs mr vs xr – what is the difference?, howpublished = https://www.plugxr.com/augmented-reality/ar-vr-mr-xr/, note = Accessed: 2024-05-04.

[3] bevy_mod_raycast, howpublished = https://github.com/aevyrie/bevy_mod_raycast, note = Accessed: 2024-05-04.

[4] bevy_mod_raycast, howpublished = https://github.com/awtterpip/bevy_oxr, note = Accessed: 2024-05-04.

[5] F statistic / f value: Simple definition and interpretation, howpublished = https://www.statisticshowto.com/probability-and-statistics/f-statistic-value-test/, note = Accessed: 2024-06-08.

[6] How to interpret the f-value and p-value in anova, howpublished = https://www.statology.org/anova-f-value-p-value/, note = Accessed: 2024-06-08.

[7] Mixed reality with passthrough, howpublished = https://developer.oculus.com/blog/mixed-reality-with-passthrough/, note = Accessed: 2024-05-04.

[8] Motion sickness in vr, howpublished = https://varjo.com/learning-hub/motion-sickness/, note = Accessed: 2024-05-04.

[9] Openxr, howpublished = https://www.khronos.org/openxr/, note = Accessed: 2024-05-04.

[10] System usability scale: What it is, calculation + usage, howpublished = https://www.questionpro.com/blog/system-usability-scale/#system_usability_scale_sus_survey_questionnaire_with_examples, note = Accessed: 2024-05-04.

[11] Xr_fb_passthrough, howpublished = https://registry.khronos.org/openxr/specs/1.0/html/xrspec.html#xr_fb_passthrough, note = Accessed: 2024-05-04.

[12] John Brooke. *SUS – a quick and dirty usability scale*. 1996.

[13] Josef Buchner, Katja Buntins, and Michael Kerres. The impact of augmented reality on cognitive load andperformance: A systematic review. *Journal of computer assised learning*, 2021.

[14] Carlo Caponecchiaa, Wu Yi Zheng, and Michael A. Regan. Selecting trainee pilots: Predictive validity of the wombat situational awareness pilot selection test. *Applied ergonomics*, 2018.

[15] Mark Colley, Oliver Speidel, Jan Strohbeck, Jan Ole Rixen, Jan Henry Belz, and Enrico Rukzio. Effects of uncertain trajectory prediction visualization in highly automated vehicles on trust, situation awareness, and cognitive load. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.*, 2024.

[16] Jurado-Rodríguez David, Muñoz-Salinas Rafael, Garrido-Jurado Sergio, and Medina-Carnicer Rafael. Design, detection, and tracking of customized fiducial markers. *IEEE Access*, 2021.

[17] G. Jones Debra and R. Endsley Mica. Examining the validity of real-time probes as a metricof situation awareness. *SA Technologies, Inc*, 2000.

[18] Mica R. Endsley, Stephen J. Selcon, Thomas D. Hardiman, and Darryl G. Croft. A comparative analysis of sagat and sart for evaluations of situation awareness. *The Human Factors and Ergonomics Society*, 1998.

[19] Mario Gianni, Gonnelli Gonnelli, Arnab Sinha, Matteo Menna, and Fiora Pirri. An augmented reality approach for trajectory planning and control of tracked vehicles in rescue environments. *IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR)*, 2013.

[20] Sandra G. Hart and Lowell E. Staveland. *Development of NASA-TLX (Task Load Index): Results of Empirical and Theoretical Research.* 1988.

[21] Sunwook Kim, Maury A. Nussbaum, and Joseph L. Gabbard. Influences of augmented reality head-worn display type and user interface design on performance and usability in simulated warehouse order picking. *Applied Ergonomics*, 2019.

[22] Matthew L. Bolton, Elliot Biltekpoff, and Laura Humphrey. The level of measurement of subjective situation awareness and its dimensions in the situation awareness rating technique (sart). *IEEE TRANSACTIONS ON HUMAN-MACHINE SYSTEMS*, 2022.

[23] Endsley Mica R. Toward a theory of situation awareness in dynamic systems. *Human Factors The Journal of the Human Factors and Ergonomics Society*, 1995.

[24] Paul Milgram and Fumio Kishino. Taxonomy of mixed reality visual displays. *IEICE TRANSACTIONS on Information and Systems*, 1994.

[25] Choi Minji, Ahn Seungjun, and Seo JoonOh. Vr-based investigation of forklift operator situation awareness for preventing collision accidents. *Accident Analysis and Prevention*, 2020.

[26] M Ostanin, R Yagfarov, D Devitt, A Akhmetzyanow, and A Klimchik. Multi robots interactive control using mixed reality. *INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 2021.

[27] Thammathip Piumsomboon, Arindam Dey, Barrett Ens, Gun Lee, and Mark Billinghurst. The effects of sharing awareness cues in collaborative mixed reality. *Frontiers in robotics and AI*, 2019.

[28] Gautam Puneet, Sanjay Agrawal Prajwal, Sahai Shubham, Sunil Kelkar Sachin, and Reddy D Mallikarjuna. Designing variable ackerman steering geometry for formula student race car. *International Journal of Analytical Experimental and Finite Element Analysis (IJAEFEA)*, 2021.

[29] Kay Stanney, Cali Fidopiastis, and Linda Foster. Virtual reality is sexist: But it does not have to be. *Frontiers in Psychology*, 2022.

[30] John Sweller. Cognitive science. *INTERNATIONAL JOURNAL OF PRODUCTION RESEARCH*, 1988.

[31] R.M Taylor. Situational awareness rating technique (sart): The development of a tool for aircrew systems design. *NATO AGARD*, 1989.

[32] Sen Wang, Hong Zhao, and Xiaomei Zhu. Effects of human-machine interaction on employee's learning: A contingent perspective. *Frontiers in Psychology*, 2022.

[33] Wei Wang, Xuefeng Hong, Sina Dang, Ning Xu, and Jue Qu. 3d space layout design of holographic command cabininformation display in mixed reality environmentbased on hololens 2. *Brain sciences*, 2022.

[34] Li Xiaohui, Sun Zhenping, Chen Qingyang, and Liu Daxue. An adaptive preview path tracker for off-road autonomous driving. *IEEE International Conference on Control and Automation (ICCA)*, 2013.