

Driving Simulator for Human Drivers Collaborating with Autonomous Trucks

Jarod Tang
1175923, jarod@student.unimelb.edu.au

Xinyu Shen
1171876, shenxs1@student.unimelb.edu.au

Chethaka Jayasekaramudeli
1071277, cjayasekaram@student.unimelb.edu.au

Executive Summary:

Mining is a cornerstone industry in Australia, contributing significantly to the national economy. To reduce labour costs and improve safety in hazardous and remote environments, autonomous mining trucks (AMTs) and other driverless heavy vehicles are being increasingly deployed in mines. As mining companies move toward fully autonomous fleets, many operations still consist of a mix of AMTs and human-driven vehicles (HDVs). This transition introduces new challenges in ensuring safe and efficient interactions between these two types of vehicles. Currently, AMTs treat HDVs as obstacles for safety reasons, often waiting for the HDVs to leave before proceeding. This leads to traffic congestion, reducing overall efficiency. To address these challenges, this project aims to develop a driving simulator designed to capture human driver responses when sharing the road with virtual AMTs. The simulator will be pivotal in enhancing the collaboration between AMTs and HDVs by providing insights into how human drivers interact with AMTs. By incorporating human driver behaviour into the control loop of AMTs, the system will be able to predict human actions, improving team performance and coordination between autonomous and human-driven vehicles, achieving safer and more efficient mining operations. The project consists of five key tasks: 1) Development of ROS2 and Unreal Engine 5 elements to simulate AMTs in realistic mining environments; 2) Creation of a comprehensive driving simulator to measure human driver responses in mining settings. 3) Integration of AMTs with the simulator, enabling seamless interaction between autonomous and human-driven vehicles. 4) Design of Human-in-the-loop experiments with an ethics application to gather real-world data on human driver responses to AMT behaviours in shared spaces. 5) Pilot study to validate the system's effectiveness, refine the models, and ensure operational safety and efficiency.

Copyright © Jarod, Xinyu and Chethaka, 2024.

1 Introduction

From the series of gold rushes during Australia's colonial period in the 1850s to the early 21st century, mining has been a cornerstone of Australia's economy. Today, it remains a well-established sector and a key contributor to the national economy, accounting for around 13.6% of its total GDP in 2023 (International-Trade-Administration [2022]). However, mining has always been an inherently risky occupation, with the industry still having the third-highest fatality rate of any sector, averaging nine worker deaths each year (and Skills-Australia [2024]). Moreover, vehicle interactions have accounted for roughly 20% of mining fatalities over the last decade (Eastwood [2023]). To meet the growing demands of productivity and safety, mining operations are increasingly relying on Autonomous Mining Trucks (AMTs, which are a class of autonomous vehicles (AVs)) to navigate hazardous and often harsh environments. These autonomous vehicles offer a safer alternative to human-driven vehicles, reducing the need for workers to be exposed to dangerous conditions such as extreme weather, dust, and the risk of accidents.

However, as mining sites transition to incorporating more AMTs alongside traditional human-driven vehicles (HDVs), new challenges arise. Human drivers often struggle to collaborate effectively with autonomous systems due to their lack of understanding of AMT behavior. Conversely, human behavior may be difficult for AMTs to predict, which reduces their effectiveness. The lack of collaboration between HDVs and AMTs can lead to issues such as congestion, inefficiencies, and potential safety risks. Understanding human behavior in the presence of autonomous systems, and ensuring collaboration, becomes essential to improving interactions between AMTs and HDVs. With the increasing integration of autonomous mining trucks, ensuring the safe operation of both human-driven vehicles (HDVs) and autonomous mining trucks (AMTs) in a collaborative environment is imperative.

To address these issues, this project aims to develop a high-fidelity driving simulator in Unreal Engine 5, designed to replicate mining scenarios where human drivers share the road with virtual autonomous vehicles. This simulator allows for controlled experiments to collect sensor data on human reactions, enabling researchers to better understand how human drivers behave around autonomous vehicles. By capturing these interactions, the simulator provides critical insights that can inform the design and optimization of future autonomous systems, ensuring smoother collaboration and improving safety and efficiency in mining operations. In addition, to allow for future developments, this project incorporates a ROS2 framework that allows new autonomous models to be modular and flexible.

To assess the effectiveness of the developed simulator, a pilot study is conducted. This study offers the opportunity to recognise and resolve potential issues in the study's design, methodology, and data collection methods. The key objectives of this pilot study are to address the following questions:

1. How do participants' physiological signals respond during the simulation? What do these signals reveal about the physical and cognitive demands placed on drivers?
2. How do human drivers (HDVs) interact with AVs in the simulated environment? What are the patterns of behaviour, decision-making processes, and adaptation over time in these mixed autonomy scenarios?
3. How effective is the simulator in providing a realistic and user-friendly training experience for human drivers interacting with autonomous vehicles (AVs)?

Following the pilot study, the data analysis phase focuses on evaluating the collected physiological signals and behavioural data such as HDV and AV inputs. Statistical and correlation analyses will be applied to identify patterns of interaction between HDVs and AVs, highlighting key behavioural trends and informing the future development of both the simulator and AVs

Copyright © Jarod, Xinyu and Chethaka, 2024.

specific to mining scenarios.

2 Literature Review

Driving simulators have taken up popularity in recent times due to their ease of use in testing autonomous vehicles that would be difficult and expensive to replicate through physical testing. As an open source software, Gazebo has been widely used in creating simulation environments for multi-robot systems (Koenig and Howard [2004]). Gazebo provides tools and packages that can be used to design various types of environments and robot systems while adhering to physics constraints. When combined with the Robot Operating System (ROS), autonomous navigation algorithms can also be implemented accurately through the transfer of information between Gazebo as the virtual environment and ROS (Costa et al. [2016]). Moreover, the ROS framework provides a modular design where new types of autonomous navigation or simultaneous localisation and mapping (SLAM) algorithms can be easily swapped and implemented.

Despite these advantages, Gazebo is very limited in the environment realism and fidelity it is able to offer (Shah et al. [2018]). More comprehensive simulation software such as AirSim, CARLA, Unity and Unreal Engine have the ability to create much more detailed and realistic simulation environments, with significant freedom for customizability from weather conditions to traffic scenarios (Dosovitskiy et al. [2017]) & (Pereira and Rossetti [2012]). Consequently, simulators built using these simulation environments are able to offer a more realistic and immersive simulator experience due to their superiority over the Gazebo environment.

Various autonomous navigation packages such as Autoware and Apollo currently provide capabilities to govern how the vehicles behave within simulation environments (Kato et al. [2015]). Hardware in the loop autonomous driving simulators can be created by combining these existing autonomous navigation packages with a chosen simulation environment. These autonomous navigation frameworks when combined with a detailed simulation environment has the capability to produce practical and realistic driving scenarios that can even be used for machine learning and artificial intelligence training tasks (Brogle et al. [2019]). Some of these current simulators also have the capability to automatically generate various simulation environments while ensuring compatibility with virtual reality equipment (Yao et al. [2018]). The virtual reality capability can be utilised to provide an immersive simulator experience which can be valuable for training human drivers to efficiently and safely behave in these scenarios.

The LGSVL simulator is one such hardware in the loop Unity based simulator and uses modular navigation packages such as Apollo and Autoware for autonomous vehicle design (Rong et al. [2020]). It also provides capability to create additional sensors and develop new controllable objects within its simulation environments. As it is developed using the Unity engine, it is able to provide realistic looking graphics and environmental details which can be used for testing autonomous driving algorithms effectively. CARLA is another such simulation environment that provides capability to create new sensors, environments and actors using its flexible application programming interface (API) allowing to test autonomous navigation algorithms in a cost effective manner within realistic scenarios (Dosovitskiy et al. [2017]).

While each of these existing simulators provides various benefits over each other, testing the accuracy of these simulators is often conducted using human user studies as evident from the RRADS platform (Baltodano et al. [2015]). In the evaluation of the effectiveness of such simulators, the integration of physiological sensors has been invaluable in providing data regarding user engagement and mental workload, particularly when combined with emerging technologies like virtual reality (VR). Metrics such as fixation duration, blink rate, and gaze spread have been identified as reliable indicators of mental workload in driving simulators (Liu et al. [2020]). Surface electromyography (sEMG) has been used to both evaluate driver response to hazardous

Copyright © Jarod, Xinyu and Chethaka, 2024.

situations (Muguro et al. [2019]) and highlight the role of muscle activation in relation to vehicle motion, particularly during high-demand driving tasks such as lane changes (Jung and Mo [2024]). Heart rate (HR) and its variability (HRV) have been widely recognised as key indicators of stress, arousal, and cognitive workload in driving simulators. By monitoring fluctuations in heart rate, researchers can infer the level of mental strain a driver is experiencing during different phases of the simulation (Liu et al. [2020]). Whilst there is much literature on how such usability studies leverage physiological data as a means of understanding user experience, the potential of using physiological signals to control or dynamically adjust the simulation remains largely unexplored in driving simulator research.

This gap points to the need for a more robust simulation framework that can seamlessly integrate physiological feedback into the control systems of simulators, allowing them to adapt based on real-time user data. From research, it is evident that most of the currently existing driving simulators utilise their own frameworks and APIs for customisation purposes (Dosovitskiy et al. [2017]). This could limit their modularity and make it difficult to customise the simulators to fit one's specific goals and intentions for testing. Therefore, this research gains inspiration from both the simulation environment and autonomous navigation based literature and attempts to combine the advantages of currently existing simulators in a modular manner using the Robot Operating System 2 (ROS2) framework. As ROS2 is currently, widely used for robotic applications, it consists of many self contained navigation packages that offers a large amount of modularity and customisation choice (Costa et al. [2016]). Following from literature, virtual reality equipment will also be integrated with the Unreal Engine simulation environment in order to increase immersion. Consequently, rather than exploiting a gap within currently existing solutions, this research aims to combine the advantages of leading simulators such as modularity, environment customisability and VR compatibility.

3 Methodology

3.1 Simulator Development - Virtual environment

A realistic environment is imperative for an immersive mining truck simulator. The environment will be built in UE5, due to its realistic graphics and advanced shaders, which aid in creating a realistic simulator. Initial developments were done in UE 5.1.1, for compatibility reasons with ROS2, the project is later migrated to UE 5.2.1, for virtual reality (VR) compatibility. The default units for UE5 is centimeters, so when unreal units are referred to later in the report, they can be converted into centimeters.

3.1.1 Mining environment development



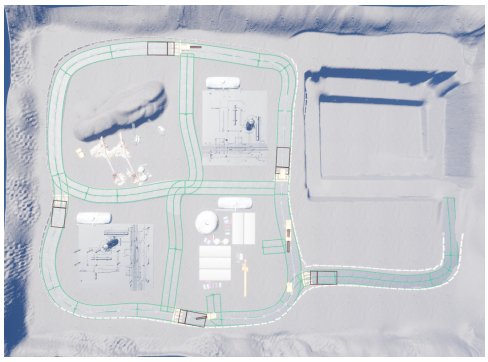
(a) Mine site photo 1 (b) Mine site photo 2 (c) Mine site photo 3

Figure 1: Sino Iron Ore Mine

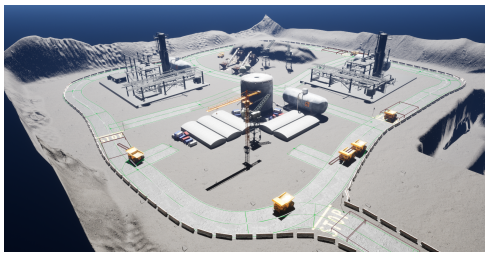
Developing a realistic mining environment is critical for the simulator. This involves designing

Copyright © Jarod, Xinyu and Chethaka, 2024.

the terrain, which can include elements such as roads, gravel paths, shipping containers, road crossings and more. The environment is a crude replica of the Sino Iron Ore Mine located in Western Australia. The size and structure of the mine site was designed with the experiment in mind, where the participants should be able to navigate the entire map in around 5 minutes, and should have to respond to traffic signals.



(a) Simulator Mine site angle 1



(b) Simulator Mine site angle 2

3.2 Simulator Development - HDV configurations

3.2.1 Vehicle Rigging and Importing

The first step to a drivable vehicle, is to rig a vehicle in Blender. This process entails defining the skeletal structure that can be animated and will allow for realistic movement and articulation of the vehicle parts within the simulation. Rigging in Blender involves setting up the bones or control points, which will be used to animate the movements of the vehicle such as steering, suspension, or even the motion of the loading bucket. The possibilities to animations are endless and hence vehicle rigging can be highly tailored to what the project requires, if the project requires the window to be able to roll down, door to open, etc, these parts can all individually be rigged and animated later on in UE5. For the sake of this project, only the 4 wheels and the body of the car need to be rigged, ensuring that the body is the parent node to the 4 wheels. Details to vehicle rigging can be found in Appendix 9.1.1. Once the vehicle is rigged in Blender, the next step involves importing the model as a skeletal mesh into Unreal Engine 5 (UE5). Currently, the textures have yet to be applied, and the truck would have a homogenous grey colour. The process detailing the application of textures is covered in Appendix 9.1.2.

3.2.2 Driving dynamics implementation

The driving dynamics can be implemented using the Chaos Vehicle plug-in: a physics-based simulation feature of UE5 that provides realistic vehicle handling. Implementing driving dynamics involves adjusting parameters such as mass, friction, and motor torque to mimic the real-world behaviour of a mining truck. This step is crucial for creating an immersive simulation experience, allowing the vehicle to interact realistically with varied terrains and under different driving conditions. Many of the aforementioned parameters are hard to come by, also contributed by how the 3D truck model isn't a direct model of any given real-life mining trucks. Hence, parameters are mainly determined by heuristics, as well as trial and error to ensure the driving characteristics of the truck are convincing. The parameters can be found in Appendix 9.1.4.

Copyright © Jarod, Xinyu and Chethaka, 2024.

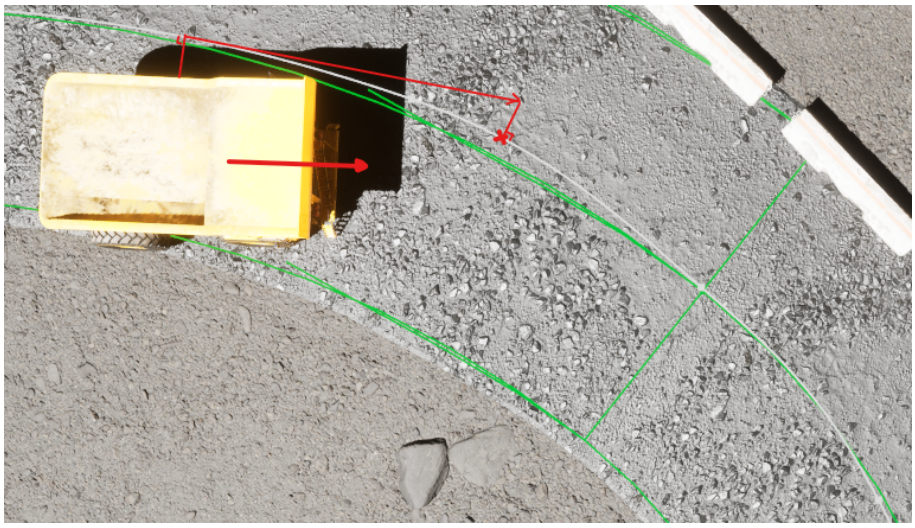


Figure 3: Path Finding algorithm

3.3 Simulator Development - AV configurations

3.3.1 Autonomous vehicles using UE5 capability

Developing a separate AV is beyond the scope of this project, but is needed for the experiments. For testing purposes, the AV algorithm will be entirely contained within UE5. Compared to the ROS2 implementation, running the AV program within UE5 has far lower latency, ensuring a much smoother and immersive experience for the driver. This AV will follow a spline path that loops around the mine site. The AV can also respond to the traffic lights that are placed along the mine site, where they are only allowed to proceed when the light is green. The AVs will also stop when there is an obstacle (AV or HDV) in front of it.

Spline tracking
First, a spline needs to be defined. One that the truck is designed to follow. For our case, it would be a path that loops the outer perimeter of the mine site.

Next, for the point that is closet to the vehicle, the unit vector of the tangent is multiplied by a fixed scalar, which is 500 unreal units in this case. The idea here is that the truck should track a point that is in front of it. A smaller scaler results in more aggressive tracking, as the vehicle will steer to a point that's closer along the spline, and vice versa. The tangent which is extended by 500 units from the direction of the spline is projected onto the closest point on the spline, and this point is set as the point that the truck is driving towards (Figure 3). The angle between where the truck is facing, and where the truck needs to drive to is extracted, as the steering value. This value is currently in angles, it would be passed through a clamped mapping block that accepts inputs from range -40 to 40, to -1 to 1, with 40 degrees being the max steering angle the truck supports. Values -1 to 1 are what the ChaosVehicle accepts as steering inputs.

The speed control method is as follows, if the plant output is exceeding the reference signal it is shut off entirely, and if it's under the reference it is turned up to max. Given how much momentum the truck has, the constant on and off of the brake inputs doesn't noticeably affect the smoothness of the trucks motion.

Collision avoidance
In order for the truck to dynamically adapt to its surroundings, it needs to be able to start and stop. **Stop** and **Go** commands can be implemented in the AV truck's blueprint. Implementation can be found in Appendix 9.1.3.

In order to detect collisions, in the blueprint for the AV trucks, a collision box in front of the car

Copyright © Jarod, Xinyu and Chethaka, 2024.



Figure 4: Collision box viewport

(see figure 4), such that when there are other vehicles inside the box, whether it is the player controlled truck, or the AV trucks, it will send a stopping signal to itself.

Traffic Light Interaction
The AV trucks should also respond to traffic signals. The traffic lights are configured to cycle between red, amber and green lights. Where the amber lights stay on for 2 seconds, red and green both stay on for 10 seconds. The amber light is there mainly for humans to react to, for AVs, that's not an issue as their response time is virtually instant. Therefore, the truck will be set to "go" when the light is green, and "stop" when it's any other colour.

Two collision boxes are added to the traffic light blueprint, lets call them the stopping collision box (left), and the slowing collision box (right), seen in figure 5. The slowing box is for pre-emptively slowing the truck down. Where the truck is slowed down until the speed is at 300 units/s, ensuring that when the truck needs to stop at the traffic light, it does so in a short enough distance. The stopping box closer to the traffic light is to stop the truck when the light is red or amber.

Below are the conditions for braking.

- If speed of vehicle is more than 1500 units/s
- If the steering inputs are above 0.25, and the speed of the vehicle is more than 700 units/s
- If truck is in slowing collision box, and the speed of the vehicle is more than 300 units/s

The first condition limits the top speed of the truck. The second condition ensures that the truck turns at a reasonable speed. The final condition slows the truck down when near traffic lights.

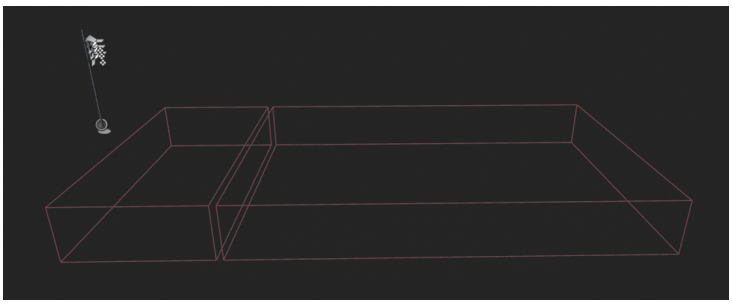


Figure 5: Traffic light in Viewport

There are some niche cases of the simulator not working as intened, which causes several pilot testing results to be invalidated. These cases are discussed in more detail in Appendix 9.1.5.

3.3.2 Autonomous vehicles using ROS2 capability

The autonomous navigation of the mining trucks as well as the steering wheel and the pedal controls highlighted in the previous section were produced solely using Unreal Engine. However,

Copyright © Jarod, Xinyu and Chethaka, 2024.

these designs in Unreal Engine are mostly not modular and is difficult to be expanded into a proper framework as its provided navigation algorithms are difficult to be customised and expanded. Consequently, this study attempts to use Robot Operating System 2 (ROS2) in order to create this capability due to its highly modular nature. The ROS2 Navigation 2 (Nav2) package contains various controllers and path planners that can be easily customised and configured to match the user's testing goals. It can take in information from the vehicle in Unreal Engine and its environment to produce a viable plan to reach its destination. While the simulator completely built on Unreal Engine will be used for the pilot study, this section explores the methodology followed in integrating the ROS2 framework with Unreal Engine 5 for future simulator design.

Development software details
Robot operating system 2 (ROS2) is a flexible framework consisting of packages, libraries and conventions for designing complex robotics systems CITE. One such package is the Nav2 software which provides various tools for creating and deploying autonomous vehicles. Various types of algorithms crucial for simultaneous localisation and mapping (SLAM), autonomous path planning and obstacle avoidance as well as robot movement are already available through this Nav2 package. Therefore, establishing this integration between Unreal Engine 5 and ROS2 will allow for more flexibility for future expansion of the simulator.

The rapyuta simulation plugin from rapyuta robotics was used as a bridge between ROS2 and Unreal Engine 5. For integration Unreal Engine 5.1.1 was used as it was the latest version that was compatible with the rapyuta simulation plugin. ROS2 humble was used as the ROS2 distro for all integration work as it is the latest long term support ROS2 distro available. Long term support was considered to be a priority when selecting the ROS2 distro as this would ensure that the future development of this project can be continued without the need for switching ROS2 distro frameworks. Ubuntu 22.04 was chosen as the linux distribution due to requirements and dependencies extending from the ROS2 humble distro choice.

Communication between ROS2 and Unreal Engine
ROS2 functions based on a node structure where each node corresponds to a smaller part of a program. These nodes interact with each other based on 'topics' that carry information of a certain type. Nodes are capable of either subscribing to a certain topic in order to access the topic data or publish to a topic to its generated data. These nodes are named subscribers and publishers respectively.

The rapyuta simulations plugin introduces a set of functions accessible in Unreal Engine that allows to create nodes, subscribers and publishers using its blueprint tool. These nodes and topics created using the plugin will be available in ROS2 outside of Unreal Engine as well. An implementation of a publisher can be seen in figure 9.2.1. Many other publisher and subscriber implementations follow this basic structure. This functionality for publishing and subscribing to topics provided by the Rapyuta Simulations plugin form the basis for communication between Unreal Engine and ROS2.

Vehicle parent class in Unreal Engine
The vehicle development done in the previous section was conducted using the 'WheeledVehiclePawn' class which allows to configure all of the vehicle details such as its components and movement drive. However, Rapyuta simulation plugins provides a parent vehicle class within Unreal Engine named 'RobotVehicle' that contains ROS2 capability on top of basic vehicle features. Moreover, various components such as lidar sensors, odometry sources and cameras can be configured and their information can be published into ROS2 using this parent class. Therefore, this class was used as the basis on which the autonomous trucks were built with ROS2 capability.

Odometry

Copyright © Jarod, Xinyu and Chethaka, 2024.

The odometry information from the autonomous trucks is one of the main points of topic information required by Nav2. ROS2 has an odometry message type named nav_msgs/Odometry which contains the (x,y,z) cartesian coordinates of the robot as well as its angular velocity information. This information for the autonomous trucks in Unreal Engine are fetched using the world coordinates which tracks all of the objects within the enviornment. An odometry component is provided by the Rapyuta simulation plugins where once attached to a vehicle with parent class 'RobotVehicle' automatically publishes its odometry information in the required message type. This odometry data can then be access by the Nav2 package in ROS2 and used for path planning and localisation purposes.

2D LiDAR scans

The autonomous trucks are equipped with a 2D LiDAR sensor close to the floor to detect barriers placed on either side of the road. This sensor allows for obstacle avoidance as well as localisation within the environment by taking point cloud scans of the environment. One such LiDAR sensor in use can be seen from figure 6 where the red lines represent points detected by the laser. The Rapyuta simulations plugin provides a 2D LiDAR component with ROS2 capability. Once this sensor is attached on to an autonomous truck using the 'RobotVehicle' parent class, it will be capable of automatically publishing the LiDAR sensor information as a sensor_msgs/LaserScan.msg that can then be used by the Nav2 package. The range of the LiDAR sensor was set to 50m as this was sufficient for accurate obstacle avoidance by the truck due to its slow moving nature. This value was chosen to be conservative as currently existing popular LiDAR sensors such as the Velodyne VLP-16 is capable of having a range of 100m with a recommended range of 80m (Mapix-technologies [2020]). Therefore, with any LiDAR sensor range greater than 50m, the obstacle avoidance performance is likley to increase as the obstacles can be identified when they are further away from the truck allowing for more informed navigation and path planning.

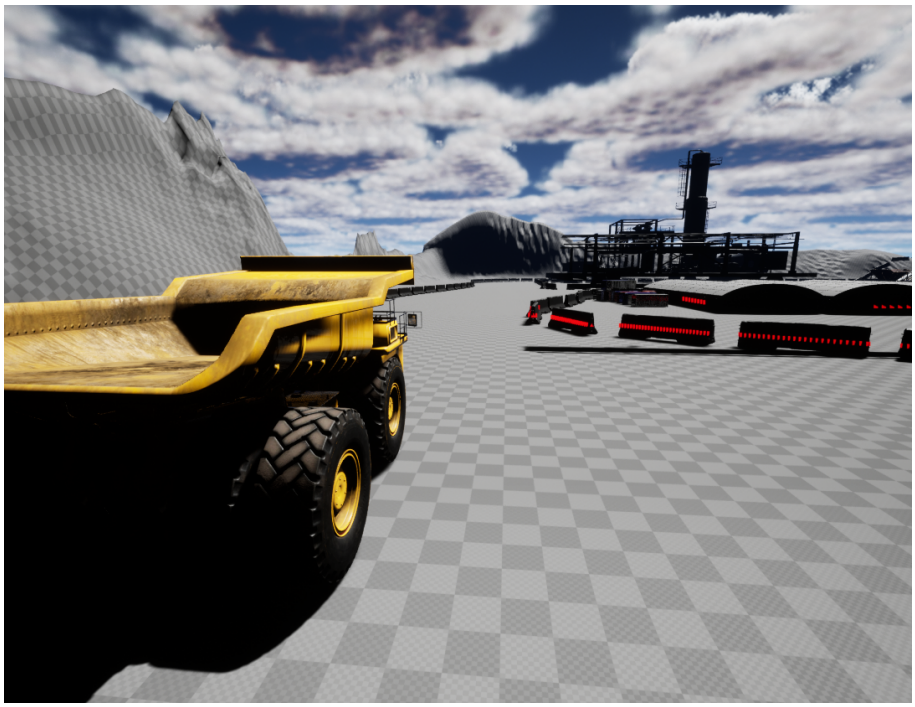


Figure 6: LiDAR sensor in use within the environment

Command velocity

Once the Nav2 package receives the odometry and the laser scan information of the autonomous truck in Unreal Engine through ROS2, it can use its controllers and planners to design a path

Copyright © Jarod, Xinyu and Chethaka, 2024.

for the truck to follow in order to reach its destination. At each time step, the Nav2 package will output a velocity command for the truck to follow according to the path generated. The 'RobotVehicle' class provides capability to automatically subscribe to this command velocity topic which is usually named 'cmd_vel' in ROS2 applications. This cmd_vel message will be of type geometry_msgs/Twist.msg and will contain linear and angular velocity commands in the (x,y,z) directions. This message can then be broken down in unreal engine and provided to a movement component to allow the autonomous truck to move in unreal engine.

Environment mapping and Nav2 configuration
Nav2 relies on an accurate map of the environment in which the robot is operating in for path planning purposes. This map can be created by mapping the environment using a LiDAR sensor attached to a robot or by scaling an image of the map appropriately. The resulting map must be of type .pgm and will contain an associated .yaml file with information about the distance per pixel (resolution) of the map along with the origin pixel location in map coordinates. This information will allow Nav2 to align the map in the same coordinates as the robot in order to ensure accurate navigation as well as laser scan and odometry readings.

In this project, the map was created by using a birds eye image of the minesite. The map from section 3.2 was updated with barriers around all of the obstacles in order to make the obstacle detection tasks easier with more clearly detected obstacles. The image was converted to binary form and scaled by comparing the distance between certain pixels to real distance in the coordinate frame in Unreal Engine according to Addison-Flair [2021]. The scaling was done so that each pixel has a resolution of 0.05 metres in the world frame in Unreal Engine. The origin location in the associated .yaml file was changed to match the origin in the Unreal Engine coordinate system. This ensures that the topic information between Nav2 that uses this generated map and Unreal Engine are consistent and in the same coordinate frame. An occupied threshold of 0.65 and a free threshold of 0.196 were specified in the .yaml file where pixels with intensity above the occupied threshold are considered as obstacles. The created map is seen from figure 7. This resulting map can then be used with the Nav2 software for localisation and path planning tasks for autonomous navigation.

Nav2 uses a .yaml parameters file that can specify various configurations such as the planner and controller used, robot information as well as localisation algorithms. For the truck model used, the robot footprint parameter was changed to be a rectangle of length and width of 12m and 9m respectively. This parameter corresponds to the birds eye cross section of the truck and is used as robot dimensions for path planning purposes. Further detailed changes made to Nav2 configuration can be found in the appendix 9.2.2.

Design of the autonomous truck
In the previous section where the truck implementation was purely in Unreal Engine, the 'wheeledvehiclepawn' parent class was used which is commonly used for vehicles. This class consists of a movement component named 'chaosvehiclemovement' component which most vehicular components in Unreal Engine uses including the mining truck model. This movement component allows for easy control of the truck model as seen from the implementation in previous section. In this section, for ROS2 compatibility the 'RobotVehicle' parent class is crucial as it is needed for proper functioning of the LiDAR and odometry components as well as for topic publishing and subscriptions. However, this parent class provided by rapyuta simulation plugins only contained movement components appropriate for differential drive of two wheeled robots as the plugin was predominantly developed to support these robot types. This leads to a conflict as both parent classes contains capability that is required by the autonomous truck but is not commonly available.

In order to share the capability of both parent classes, a two truck approach was used where two trucks were created from each parent class. The truck with the ROS2 capable parent

Copyright © Jarod, Xinyu and Chethaka, 2024.

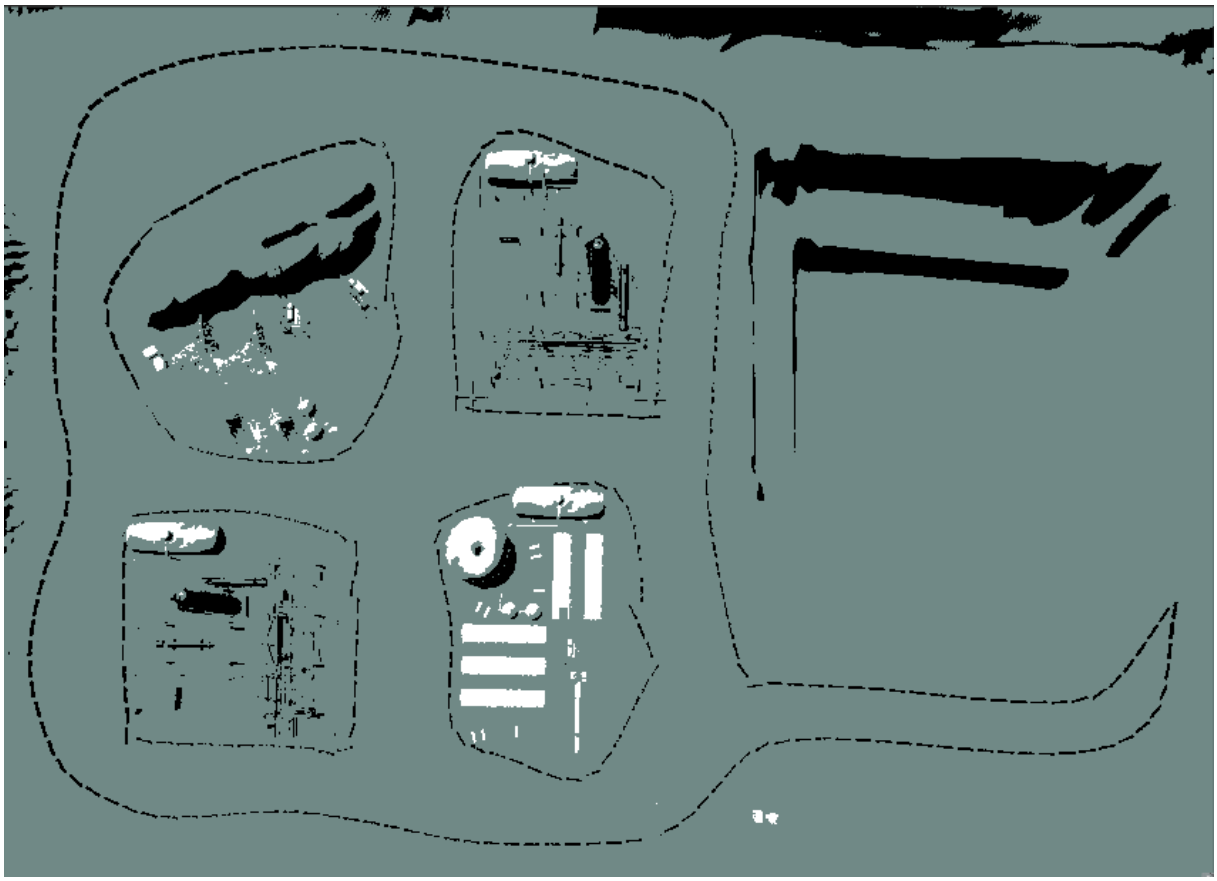


Figure 7: Produced map of the mining environment for Nav2 usage

class was used to attach the 2D LiDAR sensor and odometry component. This truck was responsible for broadcasting the laserscan and odometry messages to ROS2 for use by the Nav2 package. However, this truck was not given a skeletal mesh or a movement component and therefore appeared to be invisible in the Unreal Engine environment. The second truck made with the 'wheeledvehiclepawn' class was given the 'chaosvehiclemovement' component and was subscribed to the 'cmd_vel' topic that the Nav2 package uses to publish command velocities for path following. These two trucks were then combined together using the Unreal Engine command 'Attach Actor to Actor' which causes them to overlap and follow each other perfectly within the Unreal Engine environment. This then leads to the two truck combination where the ROS2 capable parent class scans the environment and publishes information for the Nav2 package to use through ROS2. The second truck then moves along in the Unreal Engine environment using command velocities received from the Nav2 package leading to the ROS2 capable invisible truck to perfectly overlap and follow. Further detailed information about this approach can be found in the appendix 9.2.3.

This approach was able to provide autonomous capability to the trucks within Unreal Engine using ROS2 and the Nav2 framework. The two truck approach manages to combine the movement component requirements of the truck model with the ROS2 capability needed for autonomous navigation. Therefore, this procedure managed to successfully integrate ROS2 with Unreal Engine in providing autonomous capability to the mining truck.

Copyright © Jarod, Xinyu and Chethaka, 2024.

3.4 Simulator Development - Hardware

3.4.1 Steering wheel and pedals

Users can deliver their driving inputs with the Logitech G920 steering wheel and pedal set (Figure 8). In UE5, the truck will be expecting a float input of -1 to 1 for the steering input. This is what the steering wheel will deliver. The neutral position will correspond to a float value of 0, fully rotating the wheel to the right sets the steering input to 1, and fully left sets the steering input to -1.

Since the HDV is modelled to have an automatic gearbox, the left clutch pedal is not needed. The middle pedal is set to the brake value, and the right pedal is set to the throttle value. Both of them are at 0 in their neutral position and increases to 1 when fully depressed.

For data collection purposes, the steering, throttle and brake values can all be extracted from the HDV blueprint in UE5.



Figure 8: Logitech G920

3.4.2 Chassis

The Next Level Racing GTLite Pro is used to mount the G920 steering wheel and pedal set, to ensure emersion of the simulator, comfort of the test participants.



Figure 9: Next Level Racing GTLite Pro

3.4.3 Simulator Setup

To create an immersive experience, the Vive Elite XR headset was connected via usb-c. The Delsys Trigno Avanti Platform is connected via a usb-b port. The Logitech G920 Driving Force Steering Wheels and pedal set is mounted onto the Next Level Racing GTLite Pro Chassis and secured using nuts and bolts. The simulation itself was run on a Windows 10 PC with Intel(R) Core i7-8700k processor and GeForce GTX 1080 Ti graphics card. The final simulator setup is depicted below by Figure 10.

Copyright © Jarod, Xinyu and Chethaka, 2024.



Figure 10: Simulator Setup

3.5 Pilot Study

In this section, the pilot study will be presented. A pilot study is an essential preliminary step in validating the feasibility and effectiveness of a larger research project. Moreover, the overall goal of this study is to evaluate the driving simulator's ability to provide a realistic and user-friendly training experience for human drivers interacting with autonomous vehicles (AVs) in a mining environment. This study offers the opportunity to recognise and resolve potential issues in the study's design, methodology, and data collection methods.

3.5.1 Physiological Sensor Selection

To infer information regarding the physical and cognitive demands placed on the driver during the simulation, four physiological responses were recorded: muscle activation, heartrate, gaze direction and head posture. Moreover, these were measured using the following sensors: (Surface Electromyography) sEMG, heartrate monitor and a VR headset's inbuilt eye tracker and inertial sensors. These sensors were chosen not only for their availability, but also because they have been widely used in previous usability studies for driving simulators, providing a solid rationale for their use in this context. Additionally, some measurements, while not directly verified by past studies, were included due to the convenience of having sensors capable of extracting this data simultaneously. For example, Unreal Engine 5.2 can extract both gaze direction and camera orientation. This approach allows for the exploration of potentially meaningful outcomes, even if these metrics are being used experimentally in this study. Moreover, as this is a pilot study, and the first iteration of such, the choice and placement of sensors is flexible for future iterations.

Surface EMG: Surface EMG is a non-invasive technique which is used to measure and record the electrical activity produced by skeletal muscles. Using the Delsys Trigno Avanti Platform depicted in Figure 11, high-quality sEMG data can be extracted. For data collection, the sEMG sensor depicted in Figure 11b must be directly placed on the skin. More specifically, the sensor should be placed such that the arrow is placed parallel to the muscle fibers underneath the sensor. Moreover, sensor should also be placed in the center of the muscle belly away from tendons and the edge of the muscle. The measured data is then transmitted to the base station via a WiFi signal. Using a Transmission Control Protocol/Internet Protocol (TCPIP) connection, this data is then streamed into a CSV file for further analysis.

The sEMG sensors were strategically placed on three key muscle groups: biceps brachii long head, triceps brachii long head, and sternocleidomastoid (SCM). This placement is based on the findings from studies such as Cruz et al. (2021) and Jung and Mo (2024), both of which

Copyright © Jarod, Xinyu and Chethaka, 2024.



(a) Base station (b) sEMG sensor

Figure 11: Delsys Trigno Avanti Platform

demonstrate that these muscle groups correlate well with vehicle movement and driver responses under real driving conditions. The biceps and triceps provide data on gross arm movement, particularly during steering and manoeuvring, where significant muscle activation occurs to control vehicle direction in dynamic driving tasks. Additionally, neck muscles are highly relevant for monitoring the physiological stress or fatigue experienced by the driver. In high-stress driving situations, such as collision avoidance or sudden braking, the body involuntarily prepares for impact through neurophysiological responses. These responses include muscle toning, particularly in the neck, where muscles work to stabilise the head and spine to protect vital structures during a potential collision (Muguro et al. [2019]). Sensor placement in this region can offer insights into the driver's cognitive and emotional responses during high-stakes scenarios, such as emergency braking or sudden steering corrections. For all three muscles, the sensors are to be placed in the middle part of each muscle, as according to the literature, sEMG recordings were found to be the most consistent and optimal. The sensor placement and corresponding muscles are shown in Figure 12 and Table 1.

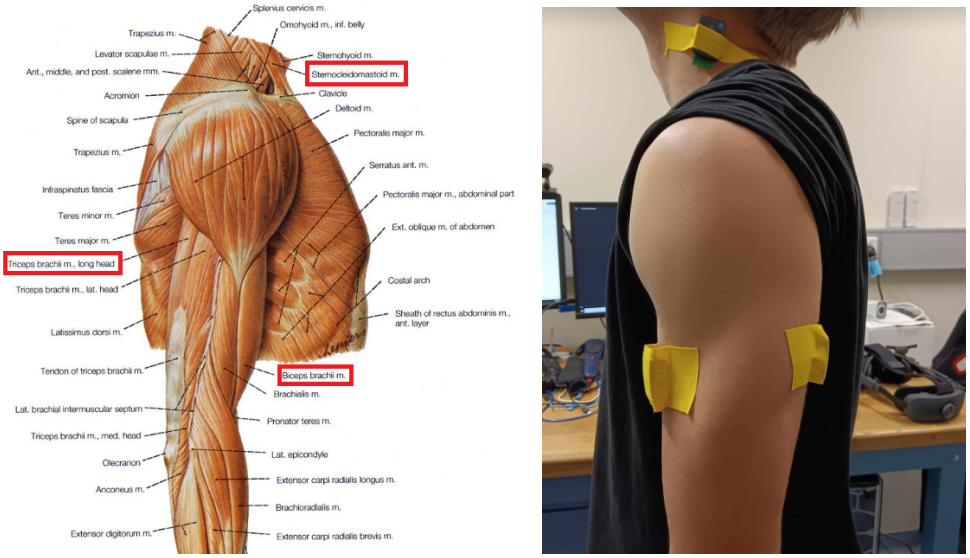
Sensor #1	Sensor #2	Sensor #3
Left sternocleidomastoid	Left triceps brachii long head	Left biceps brachii
Sensor #4	Sensor #5	Sensor #6
Right sternocleidomastoid	Right triceps brachii long head	Right biceps brachii

Table 1: Sensor placement

Heartrate Sensor: Heart rate is monitored during the simulation to measure the driver's arousal and stress levels. This data is streamed in real-time into the researcher's phone via Bluetooth and later can be extracted from the Polar Flow website as a CSV file. Moreover, it has been suggested that heart rate can be used as an objective measure to assess emotional or physiological responses to virtual environments (Liu et al. [2020]). Using the *Polar H9* sensor, heart rate is continuously tracked to gauge the participant's physiological responses to varying driving conditions. Elevated heart rate during critical or high-stress scenarios, such as sudden braking or obstacle avoidance, can indicate heightened arousal or stress. In contrast, lower heart rates may reflect lower cognitive load or more routine driving tasks. The sensor is placed directly onto the participant's skin, just below their sternum, as depicted by Figure 13.

Eye Tracker: Using the *Vive Elite XR* virtual reality (VR) headset, gaze direction data can be captured and extracted via the *OpenXR* Unreal Engine plugin. Within the domain of driving distraction research, eye-tracking technology has served as an essential tool in assessing the attention allocation and visual behaviour of drivers (Liang et al. [2024]). Moreover, gaze direction data can provide qualitative information on how drivers monitor the AV's movements, anticipate

Copyright © Jarod, Xinyu and Chethaka, 2024.



(a) (b)

Figure 12: Sensor Placement on Muscles



Figure 13: Heart Rate Sensor Location

Copyright © Jarod, Xinyu and Chethaka, 2024.

its actions, and respond to potential hazards. This is particularly important in mixed-autonomy scenarios, where successful collaboration between human drivers and AVs depends on continuous visual engagement and situational awareness. Eye-tracking data can thus provide critical insights into the cognitive processes at play during these interactions, helping to identify areas where driver focus may need improvement for safer and more effective human-AV collaboration.

Head Posture: While there is no direct literature to support using either head orientation or head posture data for driving studies, it is a valuable and convenient metric in this context. The Vive Elite XR headset, along with Unreal Engine, enables the tracking of head movements easily, making it practical to include this data. Observing head posture could reveal interesting correlations between a driver's physical reactions and AV behaviour, especially during sudden manoeuvrers or collisions. This flexibility to explore head movement data, even experimentally, may provide unexpected insights into driver-AV interaction dynamics.

3.5.2 User and AV Inputs

The collection of both HDV and AV vehicle data, including travel path, steering input, pedal use, and velocity, is essential for understanding the mechanical aspects of driver performance and human-AV interactions in mixed-autonomy environments. These metrics, alongside physiological data, offer a full-spectrum analysis of how drivers respond to AV behaviour.

Monitoring path and steering input assesses the driver's control, particularly in convoy driving and minesite navigation. Steering adjustments and pedal inputs provide insights into decision-making and reactions to AV actions or obstacles. The velocity profile further reflects how vehicle dynamics are managed, highlighting driving smoothness and efficiency. Abrupt changes in these metrics can reveal challenges in adapting to AV behaviour or operating the vehicle, guiding potential improvements in AV programming and driver training.

Moreover, the idea behind collecting this data is that not only would it serve as crucial information in assessing driver performance, but also would play an influential role in informing future AV development. For example, by identifying how human drivers adjust to AV actions (or fail to adjust), engineers can refine AV algorithms to better accommodate human behaviours. Studies such as Afsaneh Koohestani et al. (2018) have demonstrated how vehicle metrics like braking and speed changes provide vital information on how well drivers adapt to complex driving conditions in simulation (Liang et al. [2024]). This kind of analysis can also inform the development of AVs that are more responsive to human behaviour, improving safety and efficiency in mixed-autonomy systems.

3.5.3 Test Subject Recruitment

To ensure statistically meaningful results, a group of eight healthy adult participants were selected, consisting of three females and five males. A lower age limit of 18 was set as it reflects the minimum age at which an individual can obtain a driver's license in Victoria, while an upper limit of 65 corresponds to the average retirement age in Australia as of 2022, recorded at 64.8 years (Australian Bureau of Statistics, 2023). This age range is critical, as the physically demanding nature of mining roles may pose challenges for individuals beyond the typical retirement age. By including participants up to 65 years old, the study ensures that the selected group can still meet the physical requirements of the profession, while also minimising health and performance risks in older participants. Those with a history of epilepsy will be excluded to mitigate potential health complications, a precaution that aligns with licensing conditions. Table 2 depicts the demographic information of the participants.

Copyright © Jarod, Xinyu and Chethaka, 2024.

Demographic Question	Mean	Std. Deviation	Min	Max
Age (Years)	22.88	1.25	21	25
Height (cm)	172.38	8.61	158	185
How long have you had your driver's licence? (Years)	3.38	1.59	1.5	6.5
When is the last time you drove? (Days)	169.13	318.4	0	955

Table 2: Demographic Information

3.5.4 Task Design

For the purposes of this pilot study, a single task was crafted to test critical aspects of human driver (HDV) and autonomous vehicle (AV) interactions within the mining environment. The task also serves as a platform to address the three questions from Section 3.5.

In this task, the participant is instructed to drive in a convoy of mining trucks as closely as possible whilst following the road rules to the best of their ability. After navigating through the convoy along a pre-determined route shown before the experiment, the participant diverges from the convoy at a specified point and parks the truck, completing the task. To simulate a common mixed-autonomy challenge; rear-end collisions, five traffic lights are placed throughout the route to create multiple opportunities for interactions between the HDV and AV. The rationale behind this chosen scenario becomes clear when considering the following example. A typical scenario might occur at a traffic light, where a human driver expects the AV to accelerate as soon as the light turns green. However, unlike human drivers, AVs typically are programmed to optimise fuel efficiency rather than respond with immediate acceleration. This discrepancy in driving behaviour can lead to rear-end collisions, as human drivers may anticipate a faster response from the AV, reacting as they would to a human driver. The delay between the expectation and the AV's actual acceleration can catch the human driver off guard, resulting in a collision. Moreover, this type of interaction would be more prevalent and occur at a higher probability in scenarios where the HDV is following closely behind the AV.

In addition to assessing HDV-AV interactions, the task fully immerses the participant in the mining environment, providing insights into how well the simulator replicates real-world conditions. From navigating uneven terrain to reacting to environmental hazards, the task tests the simulator's fidelity and the participant's adaptability. It also evaluates the sense of 'presence' within the virtual environment, a crucial factor for eliciting realistic behaviours and physiological responses, making the training more effective.

3.5.5 Experiment Protocol

Prior to the experiment itself, participants will be asked to read and sign a consent form to indicate that the participant is willing to participate in this experiment. Upon arrival at the lab, the participants are introduced to the task, the devices that they will be wearing and the simulator itself and its subsequent input devices. Participants will be asked to report the comfort and fit of the measuring devices and simulator setup to facilitate the calibration process to determine whether the devices were worn properly. Before the actual experiment, participants underwent a 5-minute free virtual driving run so that they could become familiar with the simulator setup and controls. After completing the free run and the instructed tasks, the participants will be asked to fill in a Microsoft forms questionnaire to provide the researchers information about their age, gender, prior gaming and VR experience, driving experience, cybersickness experience and perceived performance in the simulator.

Copyright © Jarod, Xinyu and Chethaka, 2024.

3.5.6 Questionnaire

The post-experiment questionnaire used in this study includes a demographic survey, a simulation perception rating, and the NASA Task Load Index (NASA-TLX). Participants' perception and experience of the driving simulator is rated on a 10-point Likert scale, providing insight into their subjective experience of the simulation's realism, usability, and overall effectiveness. This helps to gauge the participant's comfort level and engagement with the simulator, offering valuable feedback that can be used to refine the user interface and environmental details.

Additionally, the NASA-TLX is employed to assess cognitive and physical load from six perspectives: mental demand, physical demand, temporal demand, performance, effort, and frustration. Each of these factors is rated on a 10-point Likert scale, allowing for a nuanced understanding of how the participants experience workload during the tasks. This comprehensive assessment of task load is critical for identifying areas where the simulator may need adjustments, such as reducing unnecessary cognitive overload or physical strain. By understanding how the tasks are perceived by participants, the feedback guides future development efforts, ensuring that the simulator remains both realistic and user-friendly, while effectively training drivers for complex environments.

4 Data Analysis

In this section, the methods and results of data processing and analysis are presented. The method of data processing consists of three key stages: 1) The extraction and timestamping of raw data into CSV files. 2) Data processing and refinement. 3) Data synchronisation. Lastly, several data analysis methods are used to infer meaningful information and relationships between the data. The data pathway of this data is depicted below in Figure 14. For the sake of brevity, only stages two and three will be outlined in this section. Information pertaining to stage one can be found Appendix 9.3.

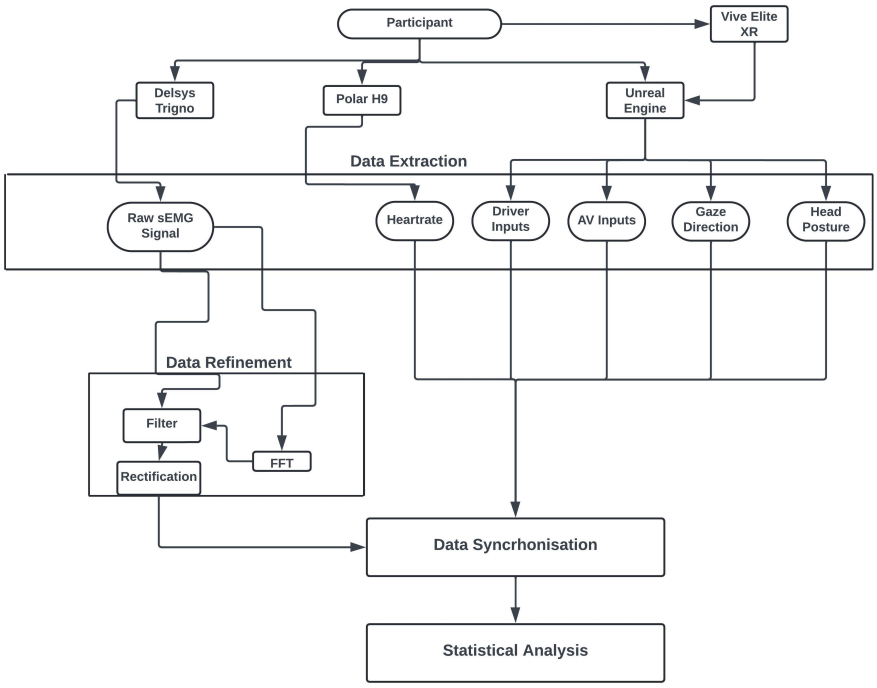


Figure 14: Data Pathway

Copyright © Jarod, Xinyu and Chethaka, 2024.

4.1 Data Refinement

For proper data analysis, it is best practice to work with data that have not been corrupted by external disturbances, as such disturbances can alter how the data is perceived and interpreted. This is especially important for both the sEMG and accelerometer signals, which can be heavily susceptible to noise. To ensure that both the sEMG and accelerometer signal are suitable for data analysis, several signal processing steps must be applied to the raw signal data. Note that since HDV, AV, posture and eye data are all derived directly from Unreal Engine (UE), it is inherently clean and free from external distortions as noise and other disturbances are not present within the simulation and hence, does not require any filtering. Additionally, heartrate sensors such as the *Polar H9* being used include algorithms that remove noise and artifacts (e.g., from motion) before outputting the heart rate data, meaning that the data received is already cleaned.

4.1.1 sEMG Signal Processing

The purpose of this section is to simply outline the techniques and rationale behind the data refinement stage. As this is the case, for the sake of brevity, the results pertaining to an individual anonymised participant will be used to show the effect of each technique. Whilst the data presented in section may exhibit some unique characteristics specific to this anonymised participant, the same approach is to be used for all data collected. This stage will consist of three parts: 1) Deriving the appropriate cutoff frequencies by performing a Fast Fourier Transform. 2) The application of an appropriate filter. 3) Signal rectification. To design an effective filter, the cutoff frequencies must be determined. This is accomplished by performing a Fast Fourier Transform (FFT) on the raw signal depicted by Figure 15.

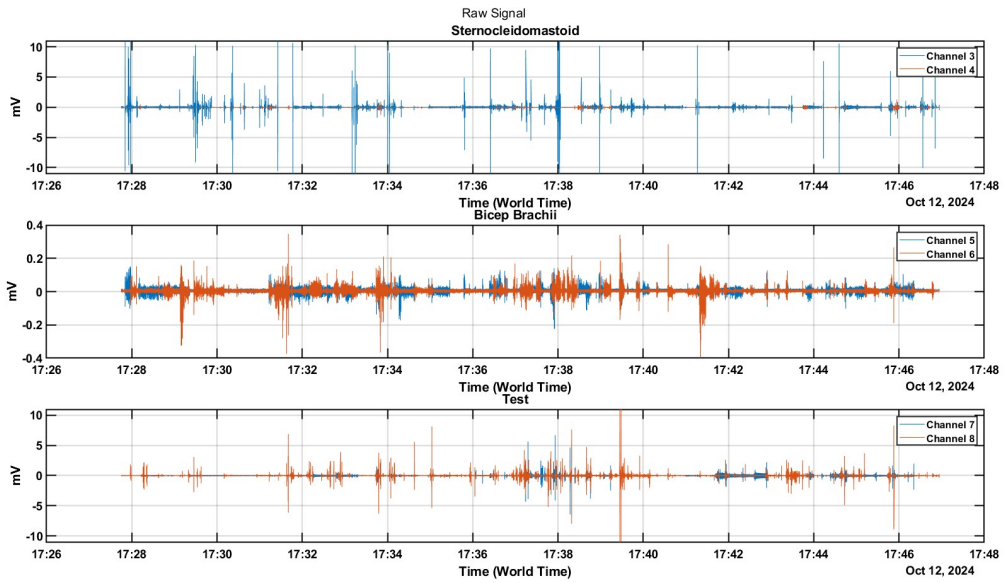


Figure 15: Raw sEMG Signal

For context, FFT is a mathematical operation that converts a time-domain signal into the frequency domain, breaking down what could be a complicated time signal into its frequency constituents (builtin, 2024), enabling the analysis of the signal's characteristics in terms of the energy present at certain frequencies. The FFT generates a plot of frequencies versus intensities, providing a visual representation of the frequency components present in the signal. By analysing this plot, appropriate cutoff frequencies can be selected to filter out undesirable noise. Using the

Copyright © Jarod, Xinyu and Chethaka, 2024.

fft routine in *MATLAB*, a plot can be generated for analysis and is depicted below by Figure 16.

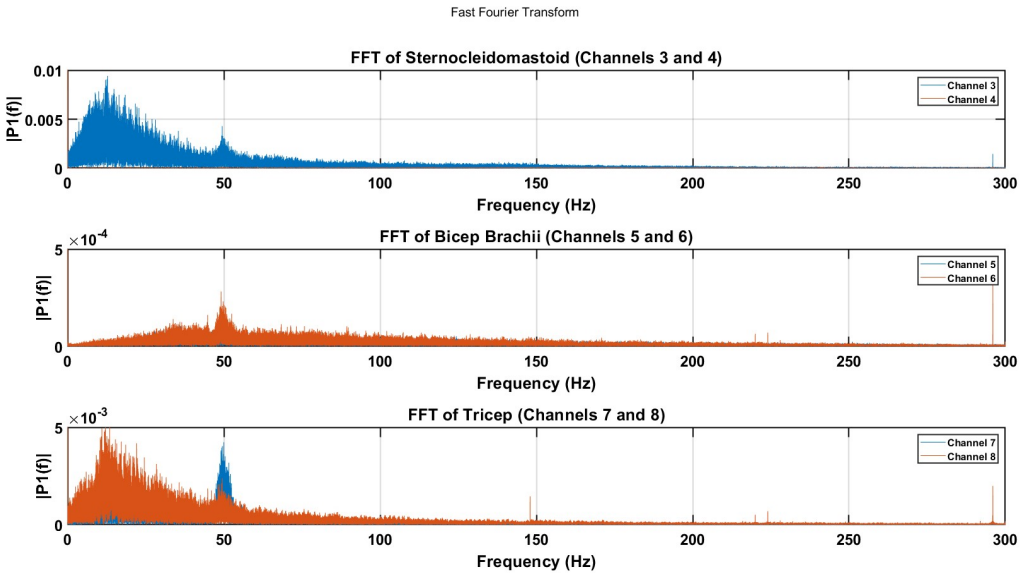


Figure 16: FFT of Different Muscles

Observing Figure 16, several trends can be ascertained. The biceps brachii signals show most of the energy is concentrated between 0 and 100 Hz, with some activity extending to 150 Hz. There are also notable spikes around 300 Hz, 350 Hz, and 400 Hz, which may correspond to artifacts or higher frequency noise. The low-frequency components below 30 Hz depict a broad rise in amplitude, potentially associated with movement artifacts such as unwanted movements of the electrodes when placed on the surface of the skin. The triceps data exhibits comparable characteristics, with significant energy below 100 Hz and a peak around 50 Hz. However, there is a significant low-frequency rise below 30Hz. Since muscle activity primarily occurs at 20-300 Hz, it would be reasonable to also attribute these signals to signal artifacts. Therefore, for both of the discussed muscle groups, a bandpass filter with cutoff frequencies of 30 Hz to 150 Hz is appropriate for isolating the desired muscle signal.

Upon examining the signal specific to the sternocleidomastoid (SCM) muscle, there are prominent two peaks exhibited at roughly 10 Hz and 50 Hz. While these low-frequency peaks may be indicative of signal artifacts, it is important to consider the influence of electrocardiogram (ECG) pulses due to the proximity of the SCM to the carotid artery. The carotid artery generates strong ECG pulses, which can increase the root mean square (RMS) value of the EMG signal, thus distorting the data (Muguro et al. [2019]). A commonly used approach to mitigate ECG interference is the application of a high-pass filter with a cutoff frequency of 30 Hz. Based on this understanding, a lower cutoff frequency of 30 Hz has been selected to effectively remove low-frequency artifacts, including ECG contamination. Additionally, to address potential high-frequency noise, an upper cutoff frequency of 150 Hz has been applied, ensuring that only the relevant muscle activity is captured for further analysis.

Once the appropriate cutoff frequencies have been selected, a Butterworth bandpass filter is applied due to its maximally flat frequency response, which preserves the signal's integrity without introducing ripples. This makes it ideal for physiological data, where maintaining the natural characteristics of the signal is critical. The trade-off between lower and higher-order filters comes down to the sharpness of the cutoff. A lower-order filter offers a smoother transition

Copyright © Jarod, Xinyu and Chethaka, 2024.

and less signal distortion, while a higher-order filter provides a steeper cutoff but may introduce phase distortion. For this study, a 4th order Butterworth filter strikes the right balance—sharp enough to remove noise at the 30 Hz and 150 Hz cutoffs, but smooth enough to retain the key features of the EMG signal for accurate analysis. The filtered signals are depicted by Figure 17.

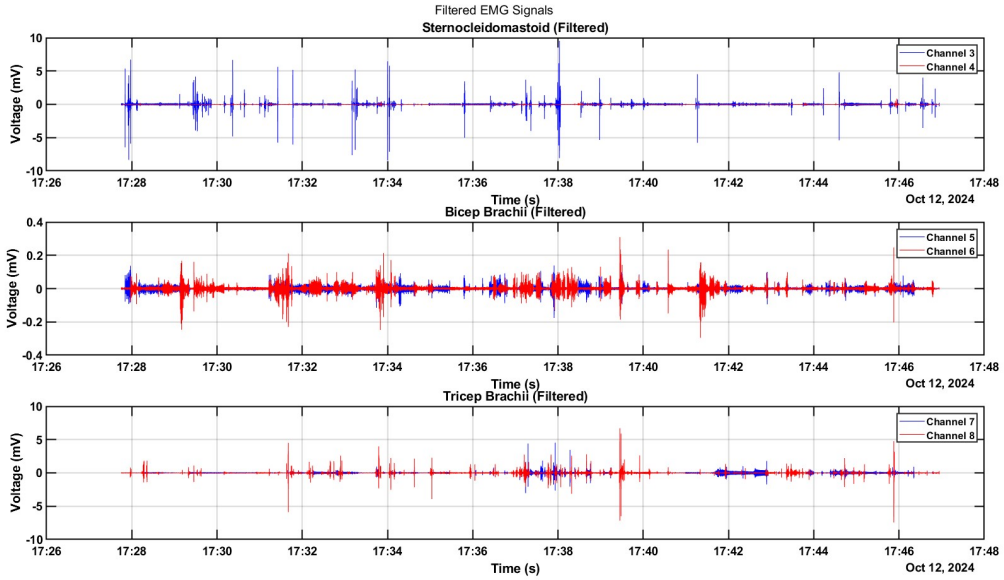


Figure 17: Filtered EMG Signals

Full-wave rectification is performed next to address the presence of both positive and negative components in the filtered EMG signal. If these components were averaged without rectification, they would cancel each other out, yielding a value close to zero, which is statistically insignificant. To ensure that the signal's power and amplitude are accurately represented, the signal is rectified by taking its absolute value. This ensures that both positive and negative portions of the signal contribute equally to the analysis, allowing for more accurate calculations, such as mean power and signal amplitude, which are essential for interpreting muscle activity.

4.2 Data Synchronisation

Data synchronisation is a critical step in the data analysis process to ensure that all sensor data, whether physiological or motion-related, is accurately aligned over time. Even though each sensor collects data independently, they must be synchronised to allow for accurate comparison and analysis. This is particularly important in high-frequency data collection, where even slight delays can lead to misaligned datasets, ultimately resulting in inaccurate interpretations of driver behaviour (Liu et al. [2020]). To mitigate issues such as latency and drift, all sensors are connected to the same computer, ensuring that data streams remain aligned. Additionally, timestamps are recorded using world time, making it easier to align different datasets during post-processing.

To facilitate comparisons across all datasets, they must share a common sampling rate. This can be accomplished either through upsampling or downsampling. Upsampling is the process by which the sampling rates of other datasets by adding interpolated data points inbetween the original measurements, enabling lower frequency data to match the higher frequency data whilst retaining potentially valuable high-frequency information. Conversely, downsampling reduces the sampling rate of higher frequency data by removing certain data points, making previously high-frequency datasets compatible with lower frequency datasets. The main tradeoff between

Copyright © Jarod, Xinyu and Chethaka, 2024.

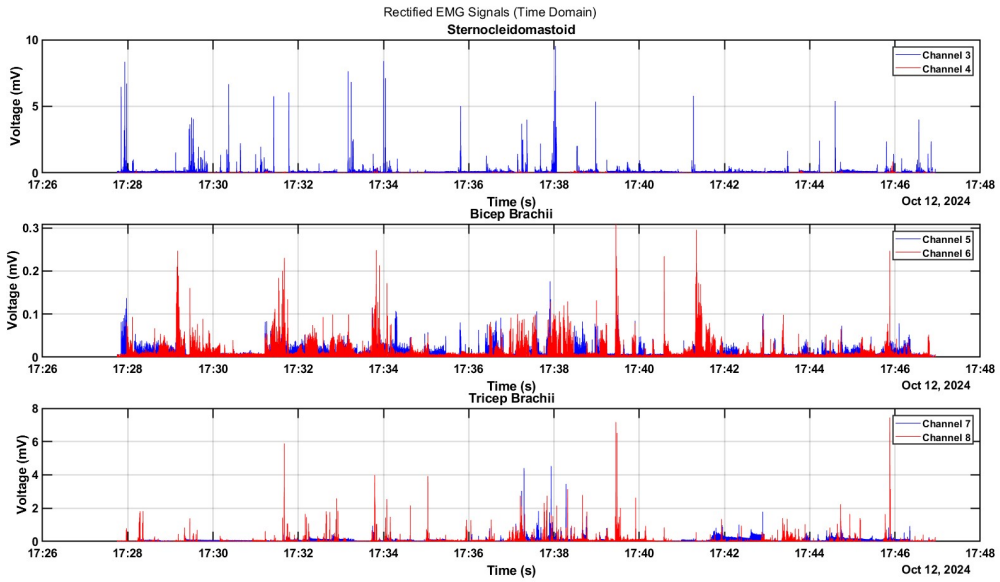


Figure 18: Rectified EMG Signals (Time Domain)

the two methods is the balance between computational complexity and the potential loss of high-frequency information due to a reduction in temporal resolution. Given that the computational resources used in this study are sufficiently powerful, upsampling was chosen as the preferred method to retain as much information as possible.

In this study, the EMG data, sampled at 2 kHz, serves as the reference timebase for synchronisation due to its high resolution. Using this reference, new time vectors are generated for each dataset, allowing the lower-frequency data to be aligned with the EMG signals. Through the use of MATLAB's linear interpolation function, *interp1*, new data points are computed between the existing measurements, ensuring that all datasets are synchronised and ready for further analysis. An example of this refined data has been included in Appendix 9.4.

4.3 Statistical Analysis Methods

To evaluate the processed data and infer meaningful relationships between driver physiological signals, driver inputs, and AV inputs, a combination of Principal Component Analysis (PCA), Correlation Analysis and Time-Series Cross-Correlation Analysis was employed. The idea behind this is that each method provides a different perspective on the interactions between the datasets, enabling a comprehensive and robust evaluation.

4.3.1 Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that reduces large datasets into their principal components that retain most of the original identity of said dataset. More specifically, PCA enables the identification of key components that account for the most variance. Each dataset (e.g., steering input, throttle, brake usage) is standardised using z-scores and combined, before applying the MATLAB routine *pca* to derive the contribution of each feature to the principal components.

Copyright © Jarod, Xinyu and Chethaka, 2024.

4.3.2 Correlation Analysis

Correlation analysis is a statistical method used to measure the strength and direction of linear relationships between two variables. In the context of this study, it allows for an initial understanding of the direct associations between different physiological readings and HDV/AV inputs. This provides a foundational perspective on how physiological responses and inputs correlate under different driving conditions, without accounting for the temporal nature of the data. Using the inbuilt MATLAB routine *corrcoef*, the correlation matrix can be generated. The correlation matrix quantifies the strength and direction of these relationships, with values ranging from -1 to 1. A correlation of 1 implies a perfect positive relationship, -1 implies a perfect negative relationship, and 0 indicates no linear relationship. The interpretation of each value derived at this stage can be found in Table 5 within Appendix 9.5.

4.3.3 Time-Series Cross-Correlation Analysis

One pitfall of correlation analysis is that it is unable to consider the temporal order of data. Time-series cross-correlation enables such dynamic analysis, allowing for the evaluation of relationships such as whether changes in driver inputs precede or follow changes in physiological responses, with special attention paid towards the maximum absolute correlation between pairs of variables to highlight the strongest possible relationships between signals, which can reveal meaningful insights about moments of high interaction or driver reaction. For this analysis, a time window of 0.75 seconds was used to evaluate lagged correlations between each pair of variables as this was found to be the average response time of drivers according to literature (Hawley [2017]). As this study serves as an exploratory, early-stage validation of the processes used, the precise timing of these correlations is less critical than understanding whether strong correlations occur at all. As such, the decision not to retain time points associated with these maximum correlations allows for a simplified analysis. Additionally, the absolute value of these correlation values was taken as a quick way to focus on the most significant relationships. Given that the main goal at this stage is to identify patterns of interaction rather than build precise temporal models, this method is appropriate for identifying which variables merit closer examination in future studies.

5 Results

5.1 Statistical Analysis Results

As per Section 4.3, PCA identifies the most influential features in the dataset by ranking variables based on their contributions to overall variance. Observing Table 3, the top three ranked features were the muscle activation of the left SCM with a loading of 0.0918, the head posture in the Z-axis with a loading of 0.0881 and the gaze direction in the Y axis with a loading of 0.0847. These results suggest that neck muscle activity and head orientation were dominant factors, likely reflecting the drivers' physical reactions to the driving environment and interactions with AVs.

The correlation matrix depicted by Figure 19 highlights the direct relationships between variables. Most variables demonstrate weak immediate correlations with one another, except for a few expected relationships, such as the moderate correlation between throttle and pedal inputs and between head posture and gaze direction. These patterns suggest that the data from individual signals does not immediately reveal strong relationships, which may imply that immediate interactions are not the key factor in this study. The weak correlations between physiological data and vehicle control inputs suggest that the interactions between driver behaviour and AV actions are not instantaneous but may unfold over time, which points to the need for further temporal analysis.

Copyright © Jarod, Xinyu and Chethaka, 2024.

Observing the time-series cross-correlation matrix depicted by Figure 20 provides more insightful trends, especially regarding physiological responses and AV control inputs. The left and right SCM muscles exhibit moderate correlations with AV throttle and velocity inputs. Specifically, the left SCM has correlation values of 0.5747 and 0.5426 for throttle and velocity, respectively, while the right SCM shows correlation values of 0.4126 and 0.3912. The head posture data, particularly along the X-axis, demonstrates moderate to strong correlations with AV throttle and velocity inputs, and weaker correlations with AV steering and braking. Additionally, a moderate cross-correlation was found between the driver's throttle and velocity inputs and those of the AV, suggesting that the driver's actions adapt in response to the AV's behaviour. Interestingly, heart rate data also follows a similar pattern, showing moderate to strong correlations with AV velocity and throttle, and weaker correlations with AV braking and steering inputs. Both the left and right biceps muscles show moderate correlations with AV velocity and throttle inputs, with the left bicep reaching a correlation value of 0.4626 with throttle and 0.5055 with velocity. Similarly, the right bicep exhibits comparable correlation values of 0.3882 and 0.3875 for throttle and velocity, respectively. This suggests that there is a relationship between arm muscle activation and the physical responses required to handle the driving environment in response to AV manoeuvres.

The tricep data follows a similar pattern, with the right tricep showing moderate correlations of 0.5114 and 0.4228 with throttle and velocity inputs, respectively. However, the left tricep shows slightly weaker correlations overall. The presence of these correlations across multiple muscle groups supports the idea that driver responses, particularly those involving the arms, are closely tied to AV actions, especially when changes in throttle and velocity occur. This implies that drivers may engage their arms more actively when they perceive the need to respond to AV behaviour, especially during acceleration or deceleration.

Rank	**Feature**	**Loading**
1	Left SCM	0.0918
2	Posture Z	0.0881
3	Gaze Direction Y	0.0847
4	Left Bicep	0.0824
5	Gaze Direction Z	0.0735
6	AV Brake	0.0721
7	Posture X	0.0716
8	AV Steering	0.0713
9	Right Tricep	0.0702
10	Right Bicep	0.0698
11	Driver Brake	0.0687
12	Velocity Driver	0.0685
13	AV Velocity	0.0684
14	Heartrate	0.0677
15	Gaze Direction X	0.0673
16	Posture Y	0.0671
17	AV Throttle	0.0668
18	Right SCM	0.0583
19	Driver Steering	0.0544
20	Left Tricep	0.0533
21	Driver Throttle	0.0460

Table 3: Ranked Features by PCA Loadings

Synthesising these findings, multiple patterns emerge. Firstly, PCA highlights that neck EMG

Copyright © Jarod, Xinyu and Chethaka, 2024.

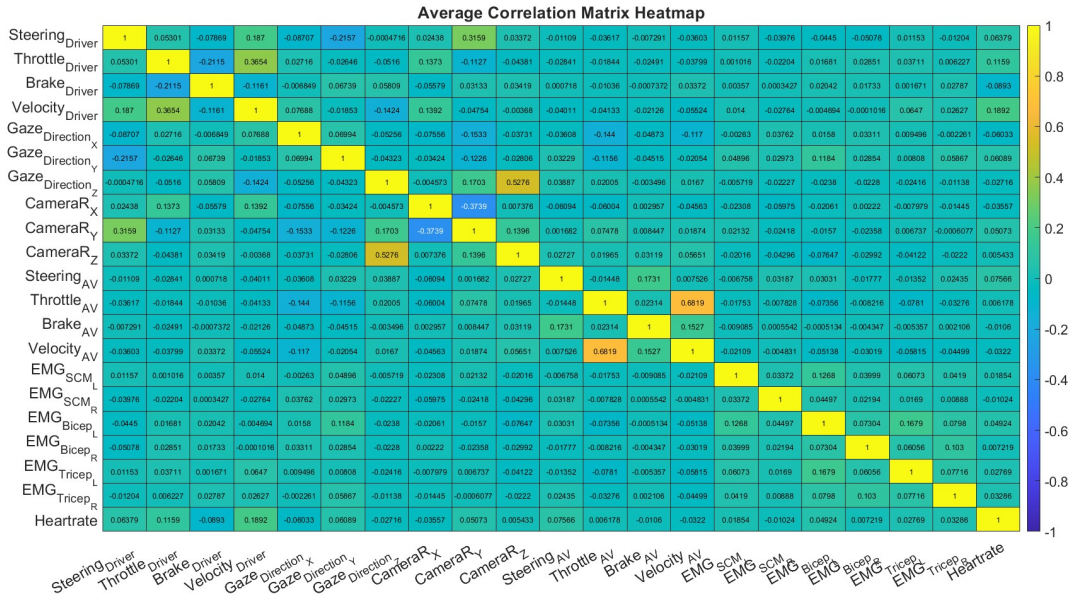


Figure 19: Correlation Matrix

data and head posture are among the most dominant features, which aligns with the cross-correlation analysis that shows time-lagged relationships between these variables and AV control inputs, such as throttle, brake, and velocity. This suggests that the driver's physical state, muscle activation and head orientation adjust in response to the AV's actions, reinforcing the importance of these physiological markers in understanding driver-AV interaction dynamics. The comparison between the weak direct correlations in the raw data and the more meaningful correlations found in the cross-correlation analysis underscores the importance of temporal factors. Driver responses may not be immediate, but rather occur as part of a more complex, time-delayed interaction with the AV system.

Moreover, the presence of gaze direction as a top feature in PCA, alongside its correlations with posture data, indicates that visual monitoring plays a significant role in understanding driver engagement. However, the weaker correlations between gaze data and direct control inputs (e.g., throttle and brake) suggest that while drivers are visually engaged with their surroundings, there may be less immediate influence of visual attention on control decisions in this specific context. This could highlight an area for further investigation in future studies, where visual attention and decision-making in mixed autonomy environments are analysed in greater depth.

5.2 Post Experiment Questionnaire

Responses to the task were mostly positive, with many agreeing that vehicle handling felt accurate, and that the simulation helped them engage with the virtual environment. Some participants also felt the experience improved their understanding of working with autonomous vehicles. Additionally data from the questionnaire can be found in Appendix 9.6.

The results from the post-simulation survey depicted in Table 4 indicate that participants generally found the simulation comfortable and enjoyable, with high scores for comfort (7.5), likability (7.25), and perceived performance (6.88). These subjective assessments align with the statistical analysis, where PCA identified driver posture and gaze as key contributors to performance, suggesting that participants were engaging effectively with the task. However, the lower ratings for physical demand (2.75) and mental effort (4.50) suggest that the simulation, while immersive, may not have been as challenging or physically taxing as a real-world scenario. This could

Copyright © Jarod, Xinyu and Chethaka, 2024.

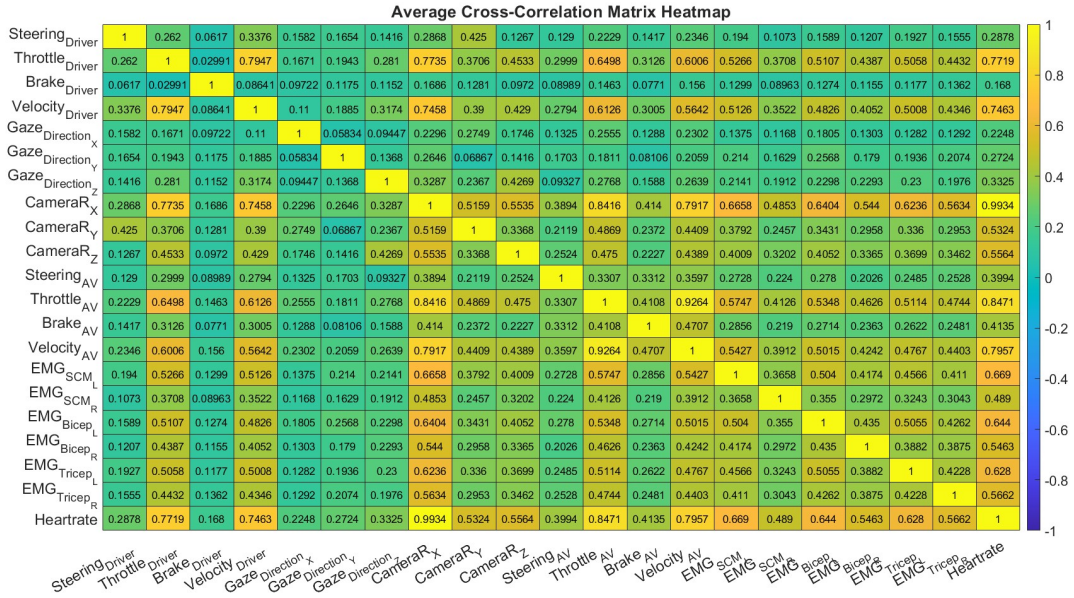


Figure 20: Time-Series Cross-Correlation Matrix

Survey Question	**Average Rating**	**Std Dev**	**Min**	**Max**
Natural feel of simulation	6.75	1.20	4	8
Pleasant driving experience	6.13	1.36	4	8
Typicality of driving controls	6.38	1.65	3	9
Like or dislike of simulation	7.25	1.71	4	10
Comfort in simulation	7.50	1.12	6	9
Mental demand of simulation	4.50	1.50	3	7
Physical demand of simulation	2.75	1.64	1	6
Time pressure in simulation	3.00	1.80	1	7
Performance in simulation	6.88	0.93	5	8
Effort required in simulation	4.50	1.32	3	7
Frustration during simulation	5.13	1.83	2	8

Table 4: Post-Simulation Survey Responses

Copyright © Jarod, Xinyu and Chethaka, 2024.

aid in explaining the weak correlations found in the time series and correlation matrices for variables like HDV brake and AV brake, as participants' relaxed engagement may have led to fewer critical interactions during the simulation. This mismatch between perceived ease and system complexity could highlight areas for refining the task difficulty to increase physiological engagement.

Additionally, the demographic data from Table 2 showed participants were generally young, with an average age of 22.88 years and 3.75 years of driving experience. The time since participants last drove averaged 169.13, which may have influenced both their performance and perception of the simulation. This may further explain why, despite the comfort and ease reported, participants rated physical and mental demands lower. These combined results suggest that while participants were engaged, the relatively low challenge level of the task could account for the weaker correlations in brake-related variables, highlighting the need to refine the task to better match real-world scenarios and increase both physical and cognitive engagement.

6 Discussion

Observing these results seems to provide a promising basis for building future pilot studies. However, other factors beyond simply just these quantitative results must be considered, keeping in mind the well-known caveat that 'correlation does not imply causation.'

A key focus of this analysis was neck muscle (SCM) activation, which was used to infer drivers' hazard responses. While this approach is valid for detecting muscle toning under stress or alertness, the signals recorded could have been influenced by other factors. As detailed in Section 3.5.1, SCM muscle activation can also result from non-hazard-related movements, such as turning or tilting the neck. Many participants were observed interacting with the virtual environment during the simulation, which could have introduced noise and distorted the intended interpretation of the EMG signals. Hence, not all neck-related EMG activity should be attributed to threat responses alone. Moreover, the analysis employed a lower frequency cutoff of 30 Hz to eliminate ECG artifacts (Section 4.1), but there are inherent limitations with this method. According to the work by Muguro et al. (2019), this approach risks attenuating signals below 30 Hz, potentially altering the spectral characteristics of the EMG data. As a solution, they suggest using median filtering on rectified and filtered signals to remove ECG peaks without compromising the signal's integrity. This could be an avenue for further refinement in future iterations of the analysis.

While gaze direction data exhibited a moderate correlation with both AV and user inputs, the Vive Elite XR headset offers additional capabilities such as tracking fixation points, pupil diameter, and blink rates. Unfortunately, the current iteration of the Vive Elite XR plugin does not yet support these advanced features. Moreover, several eye-tracking metrics are widely accepted as indicators of cognitive workload in usability research. These include variations in pupil size, blink duration, horizontal gaze dispersion, blink frequency, gaze duration, saccade count, and the horizontal standard deviation of eyeball rotation. However, there remains a gap in the literature regarding how mental workload evolves over time in response to varying distractions, particularly in environments such as driving simulators (Liang et al. [2024]). Future studies could explore the effects of visual distractions on driving performance and cognitive load to better understand these dynamics.

Additionally, variability in driving behaviour among participants was observed. Despite being instructed to follow the AV convoy as closely as possible, some drivers opted to maintain a significant distance from the nearest AV truck. This behaviour could indicate unfamiliarity with the driving setup, a lack of trust in autonomous vehicles, or even a diminished sense of immersion in the simulation environment. Interestingly, participants who maintained larger gaps

Copyright © Jarod, Xinyu and Chethaka, 2024.

tended to use the brake pedal less frequently, suggesting a weaker relationship between HDV and AV braking inputs. This pattern warrants further investigation, as it may reflect deeper issues related to driver confidence or engagement with the simulation.

Natural instincts and initial reactions to the simulator may have also influenced the data. Many participants, encountering the simulation for the first time, likely explored the virtual environment out of curiosity, potentially distorting the recorded responses. Furthermore, individual driving habits and possible misunderstandings of the tasks may have contributed to the variability in the results. It is also important to highlight that the participant pool did not reflect the typical demographic of mining operations. In mining, the median age is around 42 years, with only 12% of the workforce being female (and Skills-Australia [2024]). In contrast, the participants in this study were younger and lacked the truck-driving experience typically seen in the mining sector. Moreover, participants were only screened for a car license, not for truck-driving proficiency, which may have led to some discomfort in operating the simulated truck. Future studies should aim to include a larger and more representative sample of participants, with backgrounds more relevant to the mining industry.

Several limitations with the VR setup and available computational resources were also encountered during the study. During the initial integration of the VR headset with Unreal Engine, the system frequently froze or crashed due to insufficient computational capacity. As a result, the streamed visual quality had to be reduced to accommodate the prototype. This reduction in quality may have negatively impacted the immersion experience for participants, potentially contributing to some of the observed behavioural discrepancies. Moreover, one participant reported cybersickness, which could have been exacerbated by the lower visual quality and frame rate reduction. Future iterations of the simulator should aim to optimize system performance and consider incorporating audio, which could enhance immersion and reduce cybersickness. Currently, participants can still hear the outside environment during the simulation, which detracts from the immersive experience.

In addition to the physiological and gaze data collected during the experiment, truck path data for both the HDV and AV were recorded and saved, along with live footage of each participant's runs. Unfortunately, due to time constraints, this valuable dataset was not analysed within the scope of this study. However, incorporating these datasets in future analyses could provide critical insights into how closely participants adhered to the expected driving behaviour and how their interactions with the AV affected their driving paths. The footage, in particular, could reveal nuanced behavioural patterns that may not be fully captured through sensor data alone.

Furthermore, exploring the integration of Inertial Measurement Units (IMUs) could unlock new dimensions of analysis by tracking participant body movements and postural changes during the simulation. While initial efforts were made to collect accelerometer data using the Delsys system, technical difficulties prevented its successful extraction. However, IMUs present a promising avenue for future studies. By capturing fine-grained data on body orientation and movement, IMUs could complement existing physiological and gaze datasets, enabling a deeper understanding of driver responses to environmental stimuli and specific tasks within the simulator. This additional layer of data would provide a more holistic view of participant behaviour, offering valuable insights into both the physical and cognitive dimensions of driver interaction with the simulation.

7 Conclusions and Recommendations

The pilot study successfully addressed the key objectives of the project, demonstrating that the physiological signals captured during the simulation reveal significant insights into the physical and cognitive demands placed on drivers. Physiological signals such as SCM muscle activations,

Copyright © Jarod, Xinyu and Chethaka, 2024.

gaze direction and head posture demonstrated moderate correlation to AMT inputs such as throttle and velocity. Additionally, the performed analysis underscored the importance of temporal factors and demonstrated that driver reactions are not instantaneous but instead unfold as part of a more intricate, delayed interaction with the AMTs. Integration between ROS2 and Unreal Engine was also established using the Rapyuta Simulations Plug-in. Despite some performance challenges, the simulator provided a realistic and engaging experience, validated by both participant feedback and observed behaviours.

Future work should focus on implementing alternative forms of autonomous vehicle control to enhance the simulator's flexibility and performance. Incorporating Ackermann steering control within the ROS2 framework would better simulate the dynamics of real-world vehicles, especially in mining environments with complex terrain. Additionally, exploring advanced control strategies such as reinforcement learning or model predictive control could further optimize AMT performance. More in-depth testing, with a wider range of environmental conditions and human drivers, will also be essential for capturing diverse behaviours and ensuring the system's robustness in real-world operations.

8 Acknowledgements

Our deepest gratitude goes first and foremost to our supervisors Prof. Ying Tan, Dr Jonathan Eden, Mingliang Liu and Murphy Xu for their continued guidance and instruction in our project. Thanks to their significant instructions and patience, we were able to complete our project to a high standard and achieve the Mechatronics Discipline Merit award at Endeavour.

We would also like to extend our thanks to all the participants of our experiments. Our project has collected more than enough data for data analysis with their help. Their feedback has also invaluable to this project.

References

International-Trade-Administration. Australia - mining, 07 2022. URL `https://www.trade.gov/country-commercial-guides/australia-mining`.

Jobs and Skills-Australia. Miners, 1 2024. URL `https://www.jobsandskills.gov.au/data/occupation-and-industry-profiles/occupations/712212-miners`.

Alexandra Eastwood. An affordable autonomous solution, 08 2023. URL `https://www.australianmining.com.au/an-affordable-autonomous-solution/`.

N. Koenig and A. Howard. Design and use paradigms for gazebo, an open-source multi-robot simulator. In *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, volume 3, pages 2149–2154 vol.3, 2004. doi: 10.1109/IROS.2004.1389727.

Valter Costa, Rosaldo J.F. Rossetti, and Armando Sousa. Autonomous driving simulator for educational purposes. In *2016 11th Iberian Conference on Information Systems and Technologies (CISTI)*, pages 1–5, 2016. doi: 10.1109/CISTI.2016.7521461.

Shital Shah, Debadeepta Dey, Chris Lovett, and Ashish Kapoor. *AirSim: High-Fidelity Visual and Physical Simulation for Autonomous Vehicles*, pages 621–635. 01 2018. ISBN 978-3-319-67360-8. doi: 10.1007/978-3-319-67361-5_40.

Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

Copyright © Jarod, Xinyu and Chethaka, 2024.

José L. F. Pereira and Rosaldo J. F. Rossetti. An integrated architecture for autonomous vehicles simulation. In *ACM Symposium on Applied Computing*, 2012. URL https://api.semanticscholar.org/CorpusID:14645986.

shinpei Kato, Eijiro Takeuchi, Yoshio Ishiguro, Yoshiki Ninomiya, and Kazuya Takeda. An open approach to autonomous vehicles. *IEEE Micro*, vol.48, 11 2015. doi: 10.1109/MM.2015.133.

Craig Brogle, Chao Zhang, Kai Li Lim, and Thomas Braunl. Hardware-in-the-loop autonomous driving simulation without real-time constraints. In *IEEE Transactions on Intelligent Vehicles*, volume 4 of *IEEE Transactions on Intelligent Vehicles*, pages 375–384, United States, September 2019. IEEE, Institute of Electrical and Electronics Engineers. doi: 10.1109/TIV.2019.2919457. 2018 IEEE Intelligent Vehicles Symposium, IV 2018 ; Conference date: 26-09-2018 Through 30-09-2018.

Shouwen Yao, Jiahao Zhang, Ziran Hu, Yu Wang, and Xilin Zhou. Autonomous-driving vehicle test technology based on virtual reality. *The Journal of Engineering*, 2018(16): 1768–1771, 2018. doi: https://doi.org/10.1049/joe.2018.8303. URL https://ietresearch.onlinelibrary.wiley.com/doi/abs/10.1049/joe.2018.8303.

Guodong Rong, Byung Hyun Shin, Hadi Tabatabaee, Qiang Lu, Steve Lemke, Mārtiņš Možeiko, Eric Boise, Geehoon Uhm, Mark Gerow, Shalin Mehta, Eugene Agafonov, Tae Hyung Kim, Eric Sterner, Keunhae Ushiroda, Michael Reyes, Dmitry Zelenkovsky, and Seonman Kim. Lgsvl simulator: A high fidelity simulator for autonomous driving. In *2020 IEEE 23rd International Conference on Intelligent Transportation Systems (ITSC)*, pages 1–6, 2020. doi: 10.1109/ITSC45102.2020.9294422.

Sonia Baltodano, Srinath Sibi, Nikolas Martelaro, Nikhil Gowda, and Wendy Ju. The rrads platform: a real road autonomous driving simulator. 09 2015. doi: 10.1145/2799250.2799288.

Ying-Hsang Liu, Moritz Spiller, Jinshuai Ma, Tom Gedeon, Md Zakir Hossain, Atiqul Islam, and Ralf Bierig. User engagement with driving simulators: An analysis of physiological signals. *Lecture notes in computer science*, pages 130–149, 01 2020. doi: 10.1007/978-3-030-59987-4_10.

Joseph Kamau Muguro, Minoru Sasaki, and Kojiro Matsushita. Evaluating hazard response behavior of a driver using physiological signals and car-handling indicators in a simulated driving environment. *Journal of Transportation Technologies*, 09:439–449, 2019. doi: 10.4236/jtts.2019.94027.

Myung-Chul Jung and Seung-Min Mo. Correlation between muscular activity and vehicle motion during double lane change driving. *Sensors*, 24:5982, 09 2024. doi: 10.3390/s24185982. URL https://pubmed.ncbi.nlm.nih.gov/39338727/.

Mapix-technologies. Velodyne vlp-16 — highly resilient 3d lidar sensor with no visible rotating parts, 2020. URL https://www.mapix.com/lidar-scanner-sensors/velodyne/velodyne-puck-vlp16/.

Addison-Flair. How to create a map for ros from a floor plan or blueprint – automatic addison, 06 2021. URL https://automaticaddison.com/how-to-create-a-map-for-ros-from-a-floor-plan-or-blueprint/.

Zhoushuo Liang, Yahui Wang, Cheng Qian, Yueyang Wang, Chengyi Zhao, Hongjiang Du, Jie Deng, Xiaoqiong Li, and Yue He. A driving simulator study to examine the impact of visual distraction duration from in-vehicle displays: Driving performance, detection response, and mental workload. *Electronics*, 13:2718–2718, 07 2024. doi: 10.3390/electronics13142718. URL https://www.mdpi.com/2079-9292/13/14/2718.

Copyright © Jarod, Xinyu and Chethaka, 2024.

Dustin Hawley. When driving what is the average reaction time?, 2017. URL https://www.jdpower.com/cars/shopping-guides/when-driving-what-is-the-average-reaction-time.

9 Appendices

9.1 Appendix: UE5 Developments

9.1.1 Vehicle Rigging

The body of the vehicle is merged into a single object using the shortcut **Ctrl + J** in blender, with the 4 wheels as its child bones. Interestingly, the back wheels of the truck model was defined as a single object, which had to be split into the left and right wheels. It is also important to ensure that all of the local axes of each bone has the x-axis pointing forward and z-axis pointing upwards, failing to do so will cause the car to be driving in the wrong axis.

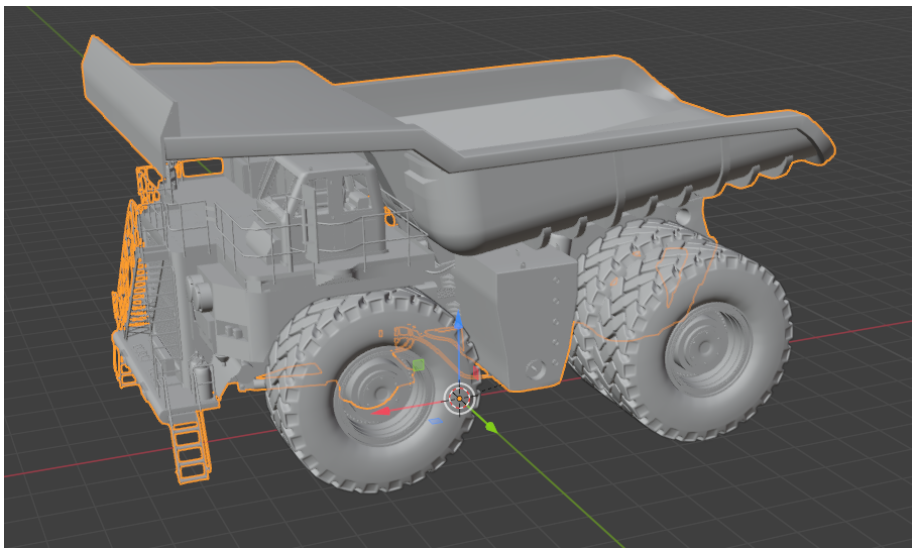


Figure 21: Rigged truck in Blender

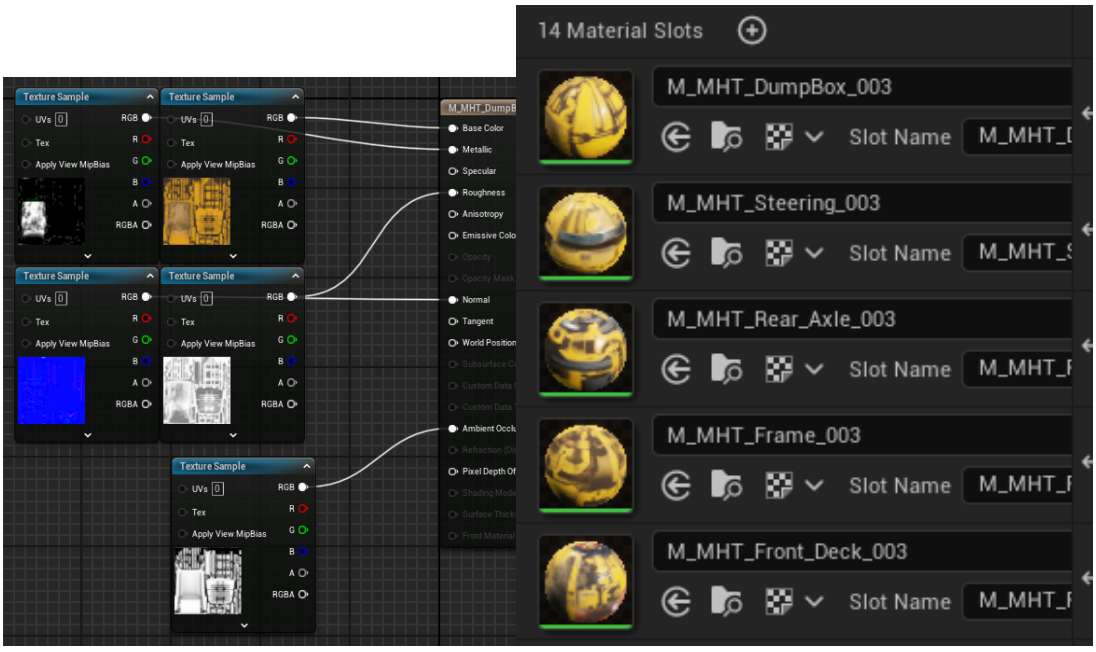
9.1.2 Texture Application

Currently, the vehicle has a homogenous grey colour, as none of the textures have been applied. This is very easy to fix, as the textures were provided with the truck model, and they simply need to be individually applied onto the mesh of the truck to complete its look. Each part of the truck has several PNG files that matches the name. These png files are the base colour layer, metallic texture, reflective, ambient occulsion. A material can be made for each of these truck parts, and applied accordingly.

9.1.3 Vehicle stop and go commands

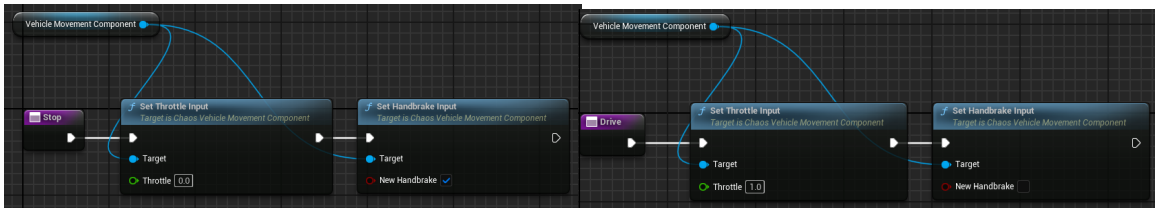
If stop, set throttle to 0 and handbrake to 1. If go, set throttle to 1, and handbrake to 0. The reason why handbrake is used instead of brake: ChaosVehicle uses reverse as brake, so by braking, throttle is set to -1, which sends an input to the truck to start reversing. When driving, this is very intuitive for the driver to use, but not desirable for designing a definitive function that stops the vehicle.

Copyright © Jarod, Xinyu and Chethaka, 2024.



(a) Creating individual materials (b) Applying materials

Figure 22: Texturing process



(a) Function to make AMT stop (b) Function to make AMT go

Figure 23: Stop and Go functions

9.1.4 Vehicle Driving Parameters

It is a matter of tuning the parameters until the truck starts to feel right to drive. For how heavy the truck is, it needs to have enough torque to go up hills, yet the engine rpm can't be too high so that it accelerates too fast for a truck on flat ground. The truck also would need to be able to stop within a reasonable distance, so the tyre grip and braking threshold for both the front and rear tyres are cranked up until desired braking distance is achieved. In the initial testing, when the truck is going at full speed, and was turning at max angle, the truck would topple over. This issue is resolved by lowering the centre of mass of the body in the physics asset by 500 units in the z-direction.

Copyright © Jarod, Xinyu and Chethaka, 2024.

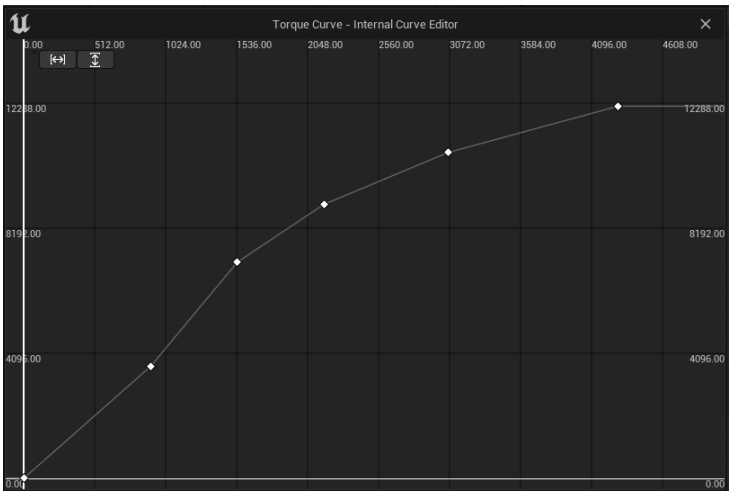


Figure 24: Torque Curve

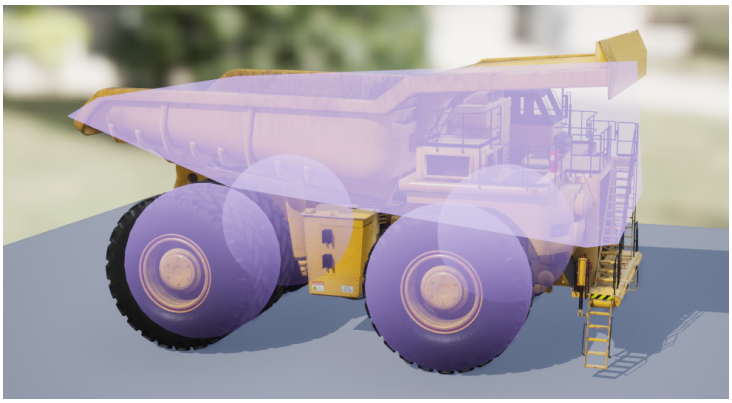


Figure 25: Collision Body

Parameter	Value
Wheel slip threshold	30000
Wheel skip threshold	30000
Front wheel max steer angle	40°
Front wheel max brake torque	20000
Front wheel max handbrake torque	30000
Rear wheel max brake torque	17000
Rear wheel max handbrake torque	30000
Mass	16000
Max Torque	15000
Max RPM	3000
Engine Idle RPM	800

In the physics asset of the truck model, the collision meshes need to be generated for the wheels and the body. The collision body can be simply generated with spheres for the wheels. For the body however, since it exhibits an irregular shape, it would be good to use a single convex hull for the collision body to adhere to the complex contours of the truck.

9.1.5 AV debugging

There is a gap between the end of the stopping collision box, and the traffic light. This is to account for the length of the truck. As seen in figure 26, more than half of the truck's body

Copyright © Jarod, Xinyu and Chethaka, 2024.

has already left the collision box, but since the back part is still inside it, the AV truck will still receive stopping signals. Due to this implementation, the truck is able to stop without exceeding the stop line, and be able to drive off if it has entirely left the collision box, regardless of what colour the traffic light is.

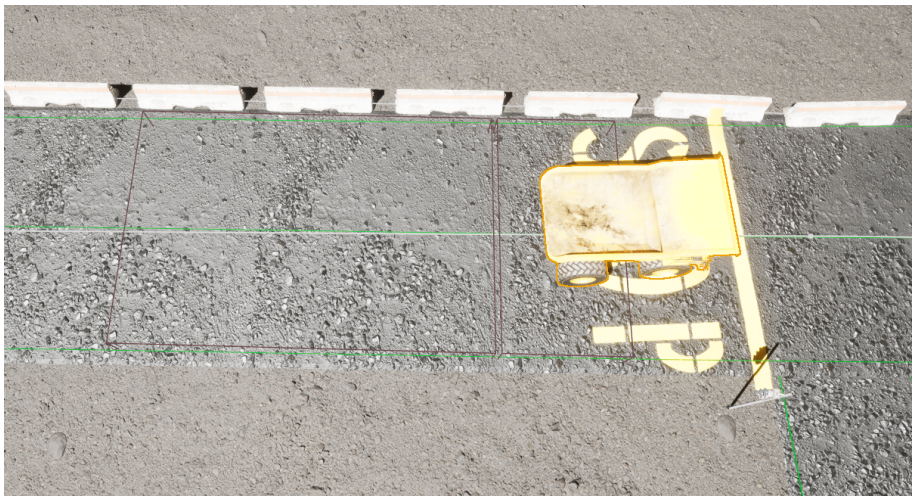


Figure 26: Traffic light interaction with AV

The bug that had invalidated many test results caused some of the trucks to sporadically stop indefinitely at traffic lights. In order for the truck to respond to the traffic light, it needs to be inside the stopping collision box. Occasionally, the trucks will approach the traffic lights with too much speed, when the light is red. This will sometimes cause the trucks to entierly overshoot the stopping collision box, whilst still being commanded to stop (red light). This causes the trucks to be stuck in the ”stopping” stage indefinitely. A fix to this is to add a slowing region before the traffic light that limits the speed of the trucks to 300 units/s, and also adding a new collision box after the traffic light, where if the truck enters it, it will be forced to drive off. This has fix the issue of trucks randomly stopping at traffic lights.

9.2 Appendix: Design of Autonomous vehicle using ROS2

9.2.1 Publisher example using ROS2

Many publishers and subscribers follow the procedure where the first ’Init’ block attached to the ’ROS2Node’ initialises the node and makes it available within ROS2. A publisher is then added to the created node using ’Add Publisher’ function and the message type for publication is casted as a ’ROS2StrMsg’ meant for the publication of string topics. The message ’Test publication’ is then added to the topic for publication using the ’Set Msg’ function and finally published using the ’Publish’ command block. The created node and the published topic messages will then be accessible within ROS2 outside of Unreal Engine.

9.2.2 Nav2 configuration

For Nav2, the default parameters file only contains planners and controllers for differential drive robots which are able to turn in place due to their two wheeled design. Consequently, it was attempted to change the controller server to Model Predictive Integral Path controller (MPPI) and the planner to Smac Hybrid planner which support ackerman driving for four wheeled vehicles. The robot minimum turning radius was also changed to 3 metres to be a conservative estimate for the mining truck. However, this approach lead to multiple compatibility issues with the ROS2 messages being received by the av2 framework. Therefore, for the final prototype of

Copyright © Jarod, Xinyu and Chethaka, 2024.

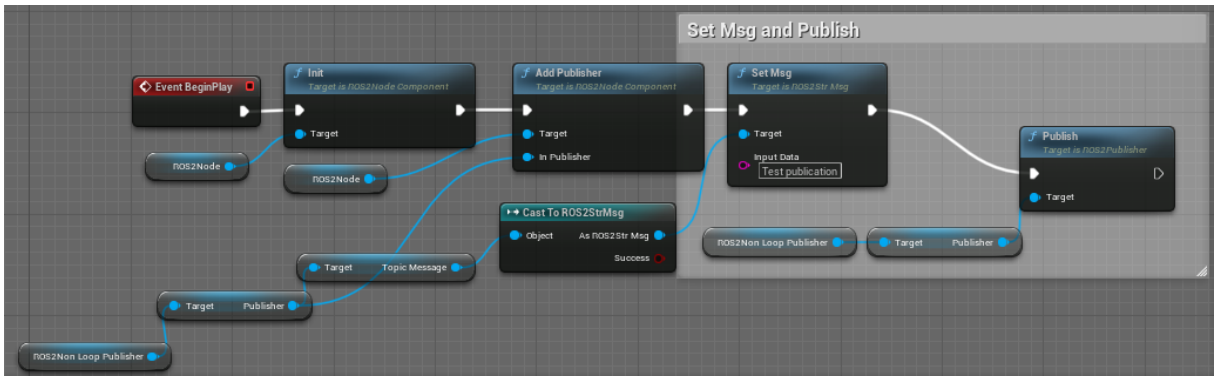


Figure 27: Publisher example in Unreal Engine

this project, differential drive planners and controllers were used. The main downside to this is that the planner and controller expect the truck to be able to turn in place following a differential drive robot. Consequently, it can lead to deteriorated movement of the autonomous truck with sudden stops of the truck as it is incapable of turning in place when the controller requests it. However, the controller and the planner tend to overcome this over time and managed to bring the truck to its destination successfully.

9.2.3 Autonomous vehicle design

In order to share the capability of both parent classes, an approach was used with two truck models using each of the classes. A truck model named 'control' was designed using the 'wheeled-vehiclepawn' parent class and the 'chaoswheeledmovement' component. This class contains the physical appearance of the truck designed in previous section using structural components and a skeletal mesh. However, this control truck do not have any ROS2 capability as it is not using the appropriate parent class. Consequently, another truck was designed using the 'RobotVehicle' parent class to allow for ROS2 capability and was named 'robot'. The LiDAR sensor as well as the odometry component were attached to this vehicle so that it can publish the LiDAR scans as well as the current odometry information. The movement component was left empty for this vehicle and it was not given a skeletal mesh so it would appear invisible within Unreal Engine. Then the two trucks using different parent classes were attached to each other's position using the Unreal Engine command 'Attach Actor to Actor'. This would attach the invisible 'robot' truck to the visible 'control' truck causing them to perfectly overlap and whenever the second truck moves, the attached invisible truck would follow matching the position exactly.

This then leads to the invisible truck to gather and publish all of the LiDAR scan data as well as the odometry data through its components and ROS2 capability. This information will be common to both the 'control' and 'robot' trucks as they are perfectly overlapped. The Nav2 package will then analyse the data and produce a path as per 9.2.2 and publish command velocity (cmd_vel) at each time step. A subscriber is created in the 'control' truck to subscribe to this cmd_vel topic using a similar method as per 27. The 'control' truck then breaks down the velocity information and provides these details to its movement component as per . Part of the blueprint for breaking command velocity messages is shown in figure 28. The 'Break ROS TWIST' seperates the velocity commands in the linear and angular directions. As seen from the 'Break Vector' blocks, the linear velocity in the x direction was multiplied by two and used as the throttle input. The multiplication was used as the initial throttle values were too low and did not manage to provide enough torque to cause the truck to start moving. The angular velocity in the z direction was used as the steering input with a multiplication by -1 to align the data to the frame used by the movement component. This procedure will then cause the 'control' truck to move according to the cmd_vel received using its 'chaoswheeledmovement' component. The

Copyright © Jarod, Xinyu and Chethaka, 2024.

'robot' truck will then perfectly follow along while overlapping the 'control' truck continuously publishing new laser scan and odometry messages using its ROS2 capability.

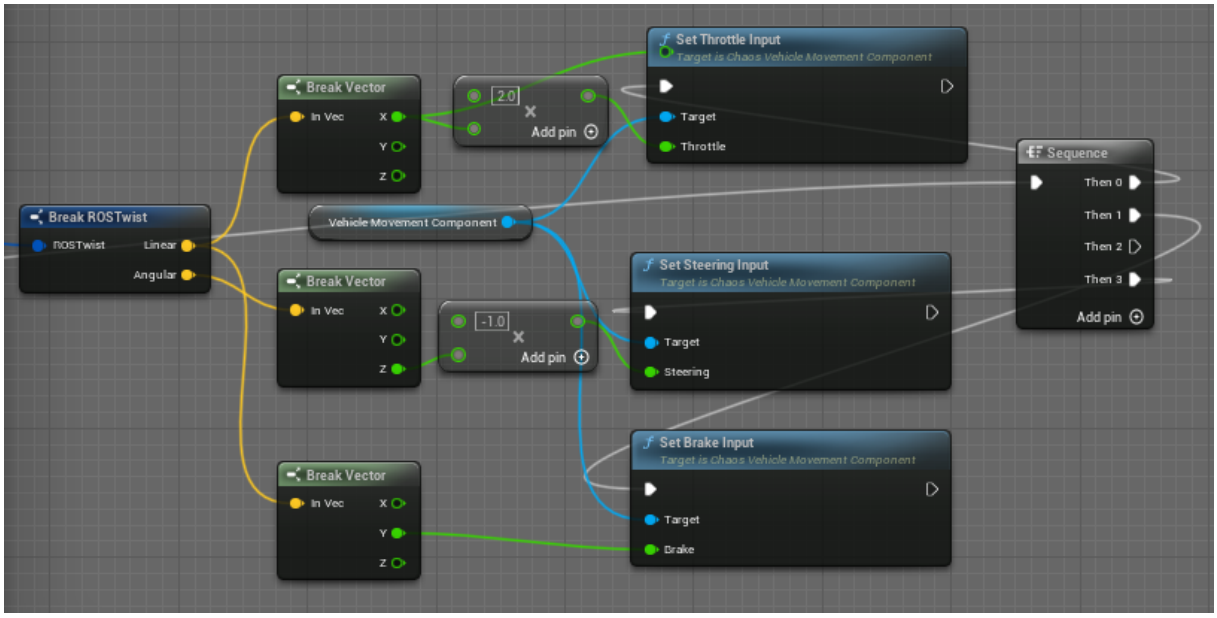


Figure 28: Using Command velocity topic information for movement

9.3 Appendix: Data Extraction

9.3.1 Extraction of EMG Signals

In the data collection process for the EMG signals, a buffer is established to continuously stream and store incoming data from the Delsys Trigno wireless system. This is achieved through a TCP/IP connection, where the program communicates with the Delsys SDK to handle data transmission from both EMG and accelerometer sensors.

The data collection begins by defining the number of sensors and creating handles for the real-time data plots. The TCP/IP connection is set up for both EMG and accelerometer data streams, where the program receives data in packets over designated ports. For EMG data, packets are transmitted through port 50041, while accelerometer data is received via port 50042. A third connection is maintained on port 50040 to communicate commands, such as starting and stopping data streaming, with the Delsys SDK.

Data is streamed into the system in chunks based on the sampling rate. The program reads a fixed number of bytes, dependent on the rate determined during setup. These data chunks are then processed in callback functions, specifically localReadAndPlotMultiplexedEMG for EMG and localReadAnPlotMultiplexedACC for accelerometer data. These functions handle the retrieval of data from the input buffer, cast the incoming bytes into single-precision floating-point format, and append the data into globally defined arrays.

Simultaneously, real-time plotting of this data is carried out in the updatePlots function. This function takes the buffered EMG and accelerometer data, extracts individual sensor channels, and updates the live plots. At each update interval, the function retrieves new data from the buffer and updates the respective plots in the figure.

Lastly, the updatePlots function also writes the collected EMG and accelerometer data to CSV files for further analysis. Each dataset is accompanied by timestamps generated using the posixtime function, ensuring that the data is properly indexed with respect to time. The use of the 'writetable' function allows for efficient data storage, appending the new data to the CSV

Copyright © Jarod, Xinyu and Chethaka, 2024.

files.

9.3.2 Extraction of Data Using Unreal Engine

In the simulation environment, driver data and other metrics are logged using custom C++ functions integrated with Unreal Engine's systems. Specifically, the SaveArrayText function within the UDriverData class facilitates the efficient saving of collected data to a CSV file, enabling real-time recording of driver input data. The function takes a directory path, file name, an array of text (representing data rows), and a flag to either overwrite or append the file. By leveraging Unreal Engine's FFileHelper::SaveStringToFile() method, this function dynamically creates or updates log files in a designated "Logs" folder within the project directory.

9.4 Appendix: Synchronised Dataset Plots

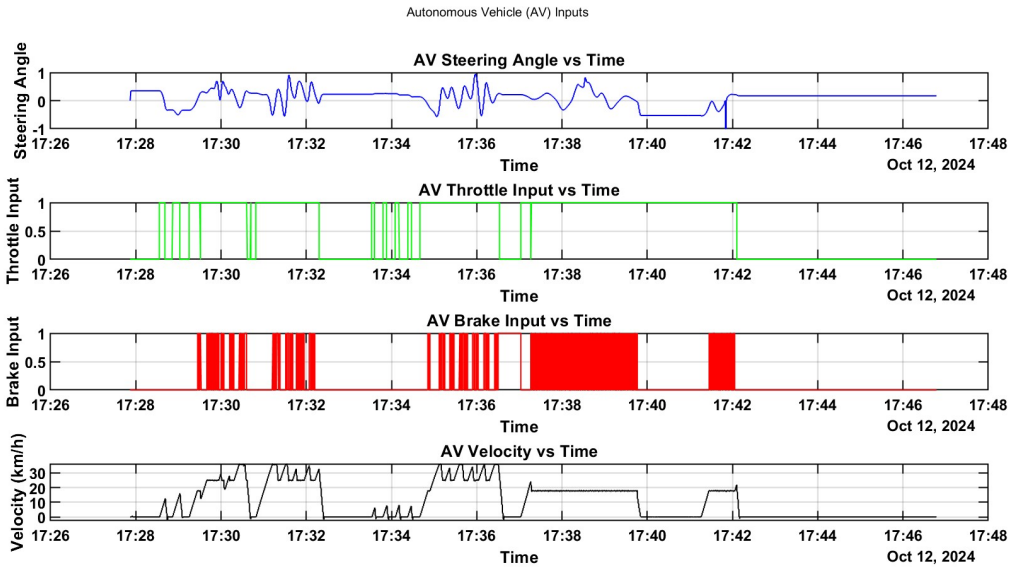


Figure 29: Synchronised Autonomous Vehicle Inputs

9.5 Appendix: Correlation Interpretation

Size of Correlation	Interpretation
0.90 to 1.00 (–0.90 to –1.00)	Very high positive (negative) correlation
0.70 to 0.90 (–0.70 to –0.90)	High positive (negative) correlation
0.50 to 0.70 (–0.50 to –0.70)	Moderate positive (negative) correlation
0.30 to 0.50 (–0.30 to –0.50)	Low positive (negative) correlation
0.00 to 0.30 (0.00 to –0.30)	Negligible correlation

Table 5: Correlation Size and Interpretation

9.6 Appendix: Questionnaire Results

Copyright © Jarod, Xinyu and Chethaka, 2024.

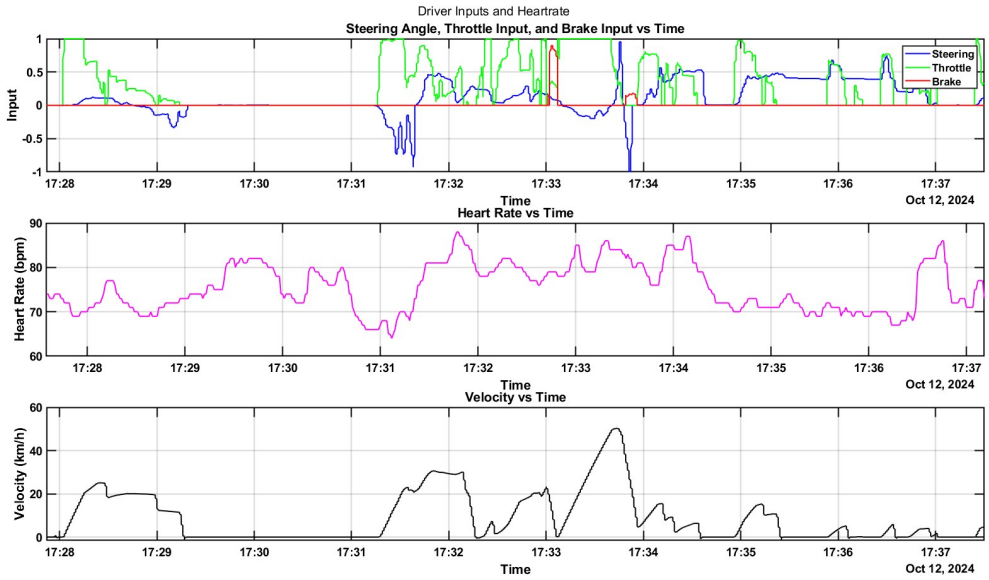


Figure 30: Synchronised User Inputs

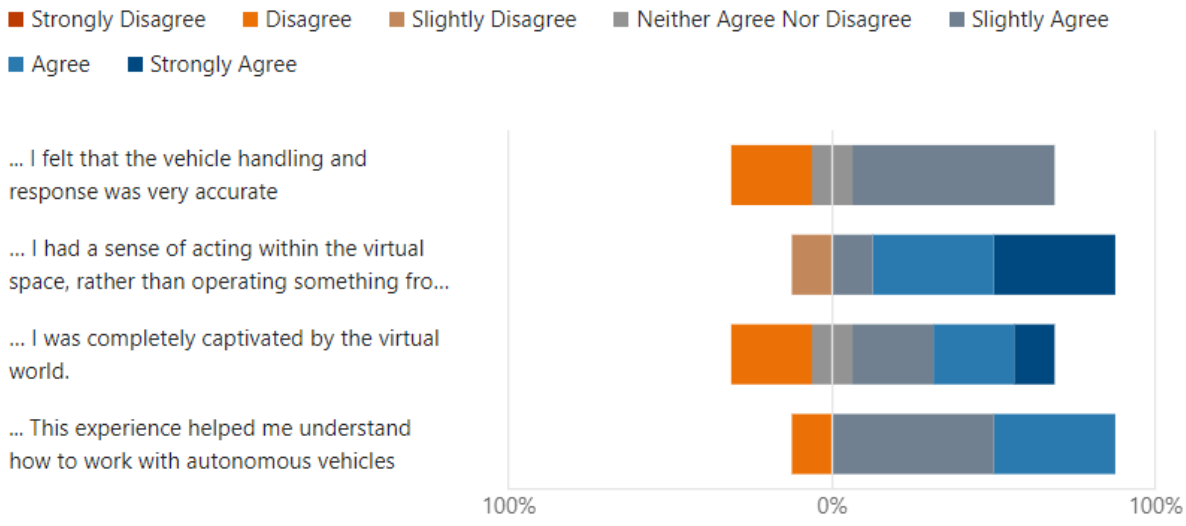


Figure 31: Part 2 of Questionnaire

Copyright © Jarod, Xinyu and Chethaka, 2024.