Pervasive Autonomy Testbed: CARLA

Puya Fard, Mahmoud Elfar, Salma Elmalaki
University of California, Irvine

***Abstract*—This project aims to advance human-in-the-loop experimentation by developing a custom-built automotive testbed that integrates physical and virtual simulation environments. The testbed features a racing seat, motion platform, steering wheel, pedals, triple-monitor setup, Virtual Reality (VR) headset, and PC station. Leveraging CARLA open source driving simulator, the platform enables comprehensive analysis of driver behavior and interaction under varied scenarios. To enhance data accuracy and insights, the setup incorporates a Zephyr BioHarness 3 biometric belt and a wearable GSR sensor, capturing detailed physiological metrics such as heart rate, respiration, and galvanic skin response. This innovative system serves as a robust tool for understanding human factors in automotive contexts, contributing to the design of safer and more intuitive vehicular systems.**

I. Introduction

This experiment involved the creation of a comprehensive setup designed to simulate a realistic driving environment, enabling human subject testing with both immersive and biometric data capture. The setup included key components such as the Next Level Racing (NLR) GTtrack Motion Platform [3], a triple-monitor stand equipped with three 50-inch flat-screen monitors, a PC station for platform operation, and the Logitech G29 Steering Wheel and Pedals [1] for precise vehicle control. Additionally, the Meta Oculus Quest 2 VR headset [2] was implemented to provide an alternative, immersive method for operating and simulating the driving experience. To support human-in-the-loop experiments, the system incorporated the Zephyr BioHarness 3 biometric belt [4] and a wearable GSR sensor, enabling the monitoring of physiological responses to enhance the realism and effectiveness of the immersive experience.

To ensure safety and functionality, the process began with an environmental preparation phase by clearing clutter and securing the workspace. Unused cables and peripherals were disconnected and stored, the seat assembly was temporarily relocated, and additional space was freed for monitor storage. Once the area was prepared, the next step involved Rebuilding the physical setup, which included reassembling the triple monitor stand to ensure the monitors were positioned at precise angles and distances for an optimal viewing experience. The seat assembly was also reconstructed, with the shifter carefully reattached in the correct orientation.

Fig. 1: Fully assembled testbed in action

The final phase involved rewiring the setup, which required referencing a wiring diagram to accurately connect the steering wheel, pedals, motion platform, and shifter. The monitors were connected to the PC station, ensuring all power and data cables were properly secured. This was followed by organized cable management to enhance safety and provide ease of access. The resulting setup, shown in Figure 1, established a stable and functional environment, fully prepared for conducting detailed human-in-the-loop experiments.

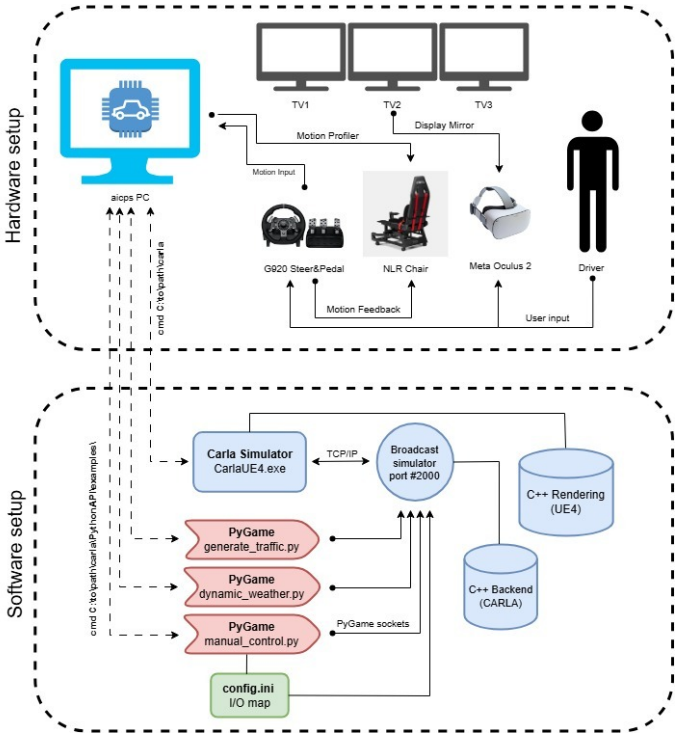

Fig. 2: Dataflow diagram

As shown in the dataflow diagram above Figure 2, the system comprises two primary modules: hardware and software setups. Each module highlights its key components, with the connections and connection types clearly defined. The main connectivity for hardware setup consists of USB and HDMI cables. The main connectivity for software setup consists of TCP/IP and Sockets.

II. Project Objective and Success Criteria

A. Project Objective

The primary objective of this project is to develop an immersive driving simulation testbed designed to replicate realistic driving conditions, enabling the collection of biometric data to study human responses across diverse driving scenarios. The motive of this project is to understand human decision making, reactions, and stress responses under varying driving conditions.

B. Success Criteria

The success of this project was defined by several key criteria: the operation of all physical and VR components, the stability and accuracy of the driving simulation, and the reliable capture of biometric data. Additional criteria included meeting performance benchmarks in hardware synchronization, particularly with GPU and storage demands, and achieving user safety and comfort during prolonged testing sessions.

III. High-Level Requirements

The high-level requirements for this project include:

1) **Integrated and Immersive Setup**: The testbed must seamlessly integrate all hardware components, including the Next Level Racing GTtrack chair, triple-monitor stand, Logitech G29 steering wheel, pedals, shifter, and the Meta Oculus Quest 2 VR headset, to provide an immersive and realistic driving experience.
2) **Hardware Synchronization**: Address challenges related to hardware limitations, such as synchronizing monitors via Nvidia Surround and managing system performance with the GTX 1060 Ti GPU. The experimental setup must ensure safety and stability through proper environmental preparation, including secure cable management, equipment alignment, and clutter-free operation.
3) **Accurate Calibration**: All components, including the monitors, steering system, and VR headset, must be calibrated to ensure precise measurements and a realistic simulation. Ensure that the Zephyr BioHarness 3 belt and wearable GSR sensor are fully functional and capable of capturing accurate physiological data during the driving simulations.

IV. Background and Standards

The project initially began in early 2023 as a collaborative effort to create a comprehensive testbed for simulating realistic driving environments with integrated biometric data collection. In the spring of 2024, the project underwent significant modifications led by a group of IoT students under the guidance of Prof. Dr. Salma Elmalaki. During this phase, substantial advancements were made to the software, enabling the simulator to process biometric data, apply filtering techniques, and perform clustering for human and behavioral studies. Recently, the project has been taken over by Puya Fard, who is building on these advancements by working on enhanced VR functionality, improved synchronization among devices and peripherals, realistic driving techniques, and a user-friendly interface for easier navigation.

TABLE I: Standards for Testbed Human Study Using CARLA

Category	Standard
Ethical Standards	IRB Guidelines [**1**], Belmont Report
Simulation and Software	Open-Source Compliance [**2**], IEEE 1733-2011, ISO 26262 [**2**]
Data and Privacy	GDPR, IEEE P7006 [**7**]
Simulation and Testing	NHTSA Guidelines, ISO/TS 15066
Performance and Usability	IEEE 29148 [**8**], ISO 9241-210
Vehicle and Traffic	Traffic Rules, SAE J3016

V. Hardware Requirements

The minimum recommended system requirements for running CARLA include a dedicated GPU with 8 GB VRAM, Intel i9 10th gen or higher, and 32 GB RAM. Carla uses Unreal Engine 4 (UE4) as its graphics engine, which is known to be resource-intensive. For better simulation performance, the recommended specs for UE4 include 64GB+ RAM, a high-end GeForce or RTX A-series GPU.

TABLE II: Hardware Specifications for the Project

Component	Specification
Graphics Card	Nvidia GTX 1060 Ti (6 GB DDR4)
RAM	16 GB DDR4
Primary Storage	512 GB SSD
Secondary Storage	1 TB Hard Drive
Operating System	Windows 10 Home
Processor	Intel i7 (11th Generation)

The recommended requirements are largely provided for running simulations to train machine learning models. For running human subject experiments, however, ensuring a smooth and immersive experience is crucial as performance issues such as lag, stuttering, or low frame rates can lead to discomfort and motion sickness, and may affect the validity of the data collected.

VI. Testbed Setup

This section aims to deliver how to recreate a such testbed from scratch. Beginning with integrates virtual reality (VR), biometric sensors, and a driving simulator to study human behavior in realistic driving scenarios. Using the CARLA simulator and advanced hardware components. This platform combines hardware, software, and data processing pipelines that will be explained in this section.

A. Peripheral Component breakdown

- **Testbed Components**: The testbed incorporates the Next Level Racing GTtrack car gaming chair, a triple monitor stand with three 50-inch flat screen monitors, a virtual reality (VR) Meta Oculus 2 headset [**5**], and a Logitech G29 steering wheel with pedals and a shifter [**4**]. Proper assembly of these components is a critical aspect of the project, as it ensures that the testbed provides a smooth and realistic driving experience. This realism is essential to ensure the accuracy and reliability of the collected data, making it comparable to real-world conditions. Therefore, careful attention is given to correctly assembling each bolt, positioning the monitors at the optimal distance from the driver's sight of view, and calibrating the seat adjustment and motion for precise alignment and functionality.

  Furthermore, connectivity between the hardware components and the operating PC is established through USB-C cables. The flat screen monitors are connected via HDMI cables and synchronized to work seamlessly alongside each other, providing a cohesive and immersive display for the simulator. Finally VR headset is connected via WiFi 2.4GHz.

Fig. 3: NLR Motion platform

Above Figure 3, we can observe how the motion platform setup is designed to be assembled for this application. Beneath the motion chair, the motion motor for the platform is installed, responsible for executing the PhysX rendering required for the simulation displayed on the screens, which, in this case, is powered by CARLA.

Fig. 4: Logitech G29 Steering set

Above Figure 4, we can observe the complete steering setup used for this project. The steering wheel and pedals are utilized to operate the vehicle, while the shifter is not in use, as the car is configured to drive in automatic mode. The pedals and shifter are connected to the steering wheel, which, in turn, is connected to the PC.

- **Biometric Sensor**: The testbed integrates a Zephyr biometric belt [**9**] and a GSR sensor to capture physiological data, including heart rate and skin conductance. Accurate data collection from the Zephyr biometric belt requires proper contact with the driver's skin, as a slightly humid skin surface is necessary for optimal sensor performance. Ensuring the belt is securely and correctly positioned around the driver's waist is a critical step to guarantee precise and reliable data acquisition during experiments.

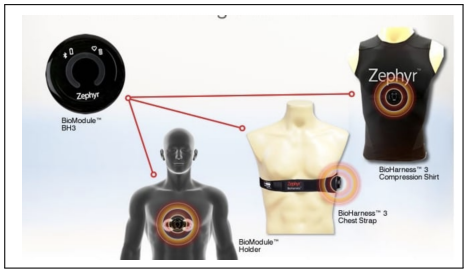

Fig. 5: Zephyr BioHarness 3

The connectivity for the biometric belt, as shown in Figure 5, is established via a Bluetooth connection, with the simulator serving as the medium for streaming data. The data collected from the biometric belt (Figure 5) will be recorded and stored for conducting human behavior studies.

- **VR Integration:** A VR headset is integrated into this project to enhance scalability and provide a more immersive simulation experience for those interested in exploring advanced levels of realism. The Meta Oculus Quest 2 [**5**] is used for this purpose. Detailed connectivity and setup are described in the milestones section of this report. The primary connection between the PC and the VR headset is established via the Meta Quest Link feature, which mirrors the simulator's display onto the headset. Below Figure 6 is the headset being used for this project.

Fig. 6: Meta Oculus 2 VR

This setup allows the driver to operate the simulation using the VR headset while enabling observers to view the driver's perspective on the main flat-screen display, facilitating behavioral analysis.

B. Preparation

In this section, we will explain how to construct the testbed from scratch using the components described in this project. For better understanding, we will follow the flowchart diagram given below in Figure 7.

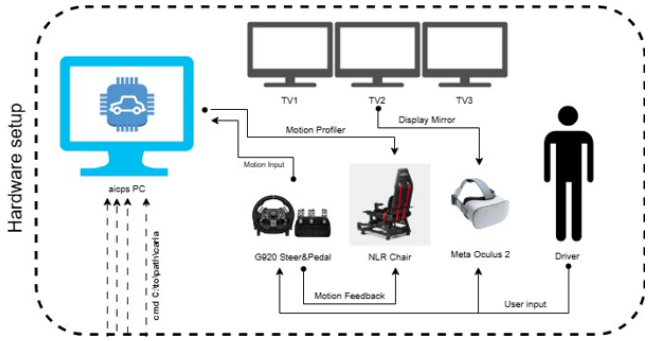

Fig. 7: Physical dataflow diagram

- **Get rid of any clustter and safety hazard.** Before setting up the testbed, it is essential to eliminate any clutter and address potential safety hazards in the workspace. Clear the area of unnecessary items, including unused cables, peripherals, and furniture, to create an organized and spacious environment.
- **Disconnect and store all cables and peripherals** Begin by disconnecting all cables and peripherals that are not required for the testbed setup. Carefully label and store them in an organized manner to avoid confusion and ensure easy access if needed later.
- **Make room for storing the three monitors** Ensure the storage area is clean, flat, and stable to prevent any damage to the screens. Position the monitors in a way that the driver is facing in a perpendicular angle towards the screens.

C. I/O drivers setup

To establish successful communication between the hardware components and the operating PC, it is essential to install the latest versions of the I/O drivers for the steering wheel, pedals, NLR motion platform chair, and Meta Oculus headset. The required drivers are listed below:

- Logitech G Hub [**4**]
- NLR Motion Platform V3 [**6**]
- Meta Quest Link [**5**]
- Nvidia GeForce

Once all the necessary drivers for the I/O devices are downloaded and updated, we can proceed to the connectivity step to ensure proper functionality of all connected devices. The Logitech G Hub software will automatically recognize the G29 steering wheel and pedals connected to the PC, enabling them to transmit input commands. The NLR driver software will allow the selection of an existing driving PhysX profile or the creation of a new one. For this project, a custom motion PhysX profile will be created specifically for CARLA. The Meta Quest Link driver facilitates the connection of the VR headset to the PC, requiring the creation of a Meta account and utilizing Wi-Fi as the connectivity medium to enable display mirroring.

D. PhysX NLR Motion platform profile setup

Once the I/O driver is downloaded and the NLR motion platform is successfully connected to the PC, a new profile can be created specifically for this project and named **Carla**. Below Figure 8 is the configurations used for the PhysX profile to replicate driving experience.

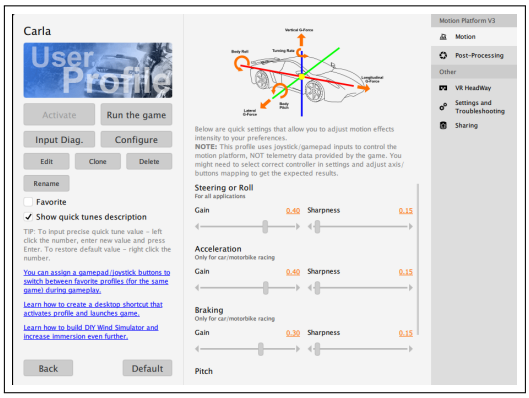

Fig. 8: Motion platform profile

There are three major factors to consider when replicating the motion feedback from the simulator: steering wheel, acceleration, and braking. Each of these categories involves specific parameters, such as gain and sharpness, that require careful adjustment to achieve realistic and responsive feedback. The values for gain and sharpness in each category have been carefully researched and rigorously tested to ensure optimal levels for realistic and accurate motion feedback. We will adhere to the configuration shown in Figure 8.

E. Meta Oculus 2 VR Profile Setup

To connect the VR headset to the host PC, the first step is to set up a new Meta Quest Link account. Next, create a user profile directly on the VR headset and ensure it is connected to the same Wi-Fi network as the host PC used for the project. This step is critical, as the device will not be able to locate the host PC if it is not on the same Wi-Fi network or LAN. The third step is to navigate to **Link Device** in the Meta Oculus app on the PC and follow the steps to add a new VR headset to the profile. As shown below in Figure 9, we will use the Link feature under settings to establish search and connection. Finally, we will click **Launch Quest Link** to activate this feature. Once the **Quest Link** is launched on the VR headset, proceed to select the display mirror feature to reflect the running simulator screen [**5**].

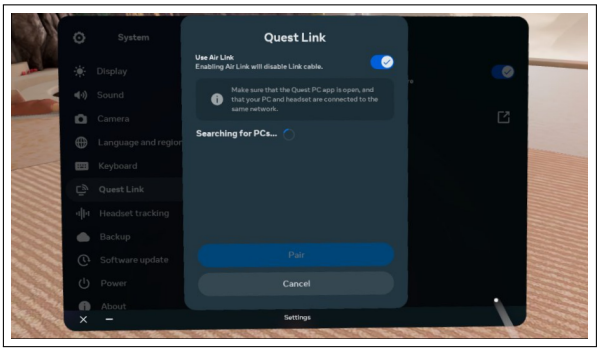

Fig. 9: Meta Quest Link

F. G29 Steering Wheel Profile Setup

Connecting the G29 steering wheel and pedals is straightforward and can be done by launching the downloaded Logitech G Hub application on the host PC [**4**]. Upon opening the application, you will be prompted to create a profile using an email or Facebook account. The application will then automatically scan for devices connected to the PC and establish a successful connection.

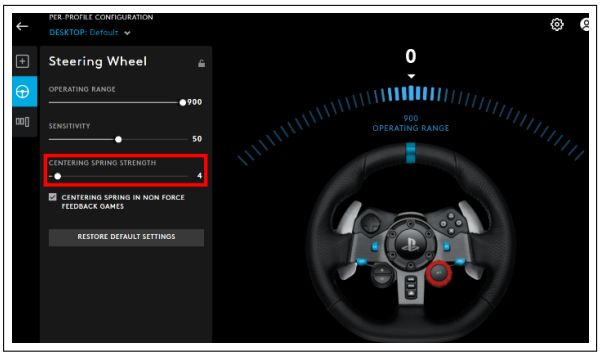

Fig. 10: Logitech G Hub profile

As shown in Figure 10, we will adjust the settings within the steering wheel profile to make it suitable for our application. To avoid the steering wheel being overly sensitive during operation, we will fine-tune the settings accordingly.

G. Challenges Encountered in the Hardware Setup

During the hardware setup process, several challenges were encountered:

- **Cable Management**: Organizing and securing the numerous cables for the monitors, steering wheel, pedals, and motion platform required careful planning to prevent tangling and ensure safe operation. It is important to consider the seat entrance and ensure it is free from cable clutter to allow the driver to sit and stand up comfortably.
- **Motion Platform Setup**: Positioning and securely mounting the motion platform under the racing chair required precise alignment and calibration to deliver accurate motion feedback during simulations. Gathering feedback from test users was crucial for refining the setup and making improvements to the PhysX motion feedback of the chair.
- **VR Integration**: The initial setup of the VR headset required configuring the Wi-Fi network and resolving connection issues with the host PC to enable display mirroring and Quest Link features. Successful connectivity could not be established without a shared LAN or Wi-Fi network, making this configuration step critical.

Addressing these challenges was crucial for achieving a functional and reliable hardware setup capable of supporting the project's objectives.

H. Software-level project breakdown

CARLA is a sophisticated software that requires a dedicated GPU, sufficient RAM, and ample storage to ensure smooth operation without crashes or performance issues. This project utilizes an Nvidia GTX 1060 Ti with 6 GB of VRAM, 16 GB of DDR4 RAM, and 512 GB of SSD storage to ensure optimal performance. Even with these specifications, it cannot run the latest version of CARLA, version 0.9.15. To ensure smooth operation, this project utilizes CARLA version 0.9.13, released in 2022. This version includes all the necessary features and functionalities required for the project, making it a suitable choice without any limitations. Furthermore, this project utilizes Visual Studio Code (VS Code) as the primary development environment and employs Microsoft's C++ rendering engine for visual rendering.

I. Setting up CARLA

Since this project utilizes CARLA version 0.9.13, the steps required to install this version will be described below.

- **Before you begin:** The following requirements should be fulfilled. The software requires a dedicated GPU with at least 6 GB of VRAM, although an 8 GB GPU is recommended, especially for tasks involving machine learning. Additionally, CARLA requires approximately 20 GB of disk space to accommodate its installation and functionality, ensuring sufficient storage capacity is crucial for a smooth setup.
  Python is the primary scripting language used in CARLA. The platform supports both Python 2.7 and Python 3 on Linux, while only Python 3 is supported on Windows. Additionally, certain installation methods for the CARLA client library require pip or pip3 (depending on the Python version) with a minimum version of 20.3. You can verify the installed pip version using a simple command.

Python 3 Commands

```
 pip3 install --upgrade pip
# and
pip3 install --user pygame numpy
```

- **CARLA Installation**: CARLA can be installed by visiting the official CARLA repository at https://github.com/carla-simulator/carla/blob/master/Docs/download.md. The package is provided as a compressed file named `CARLA_0.9.13`. To begin, download and extract the release file. This file includes a precompiled version of the simulator, the Python API module, and several example scripts to help you get started.

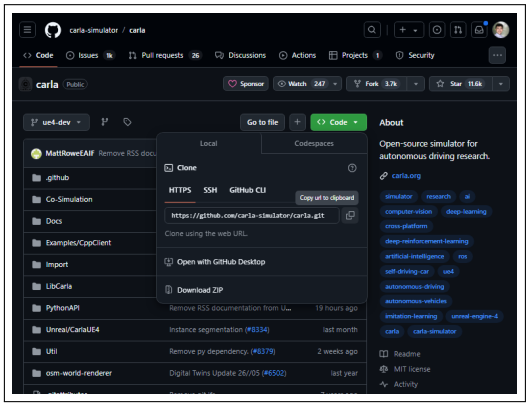

Fig. 11: CARLA Repository

Once downloaded, ensure that a well-structured folder is created to store the downloaded files, keeping all documents and resources organized for easier access and management. For a detailed procedure on setting up the CARLA environment and simulation, refer to the official documentation at CARLA Quickstart Guide.

J. Setting up VS Code workspace

Visual Studio Code (VS Code) will serve as the environment for running Python scripts and managing the CARLA simulation. To get started, simply download the latest version of VS Code from https://code.visualstudio.com/download, and create a new or log into the existing user profile. Furthermore, create a "workspace library" and organize the project by moving all files related to `CARLA_0.9.13` into this folder. This helps maintain a clean and structured environment for managing the simulation and associated scripts. Once the Python extensions are installed, you can start running the CARLA simulator and utilizing the Python APIs.

K. Challenges Encountered in the Software Setup

During the Software setup process, several challenges were encountered:

- **CARLA Installation:** Selecting the appropriate version of CARLA for the given PC specifications is a significant challenge in this project. While the setup may initially appear functional, even minor changes to the Python scripts or hardware components can lead to crashes and runtime issues. Therefore, determining the version of CARLA that best aligns with the PC's specifications is a critical first step for ensuring stability and smooth operation.

VII. Running Simulation

Now that all the necessary components are installed and set up, we can proceed with the steps required to begin the simulation and driving process.

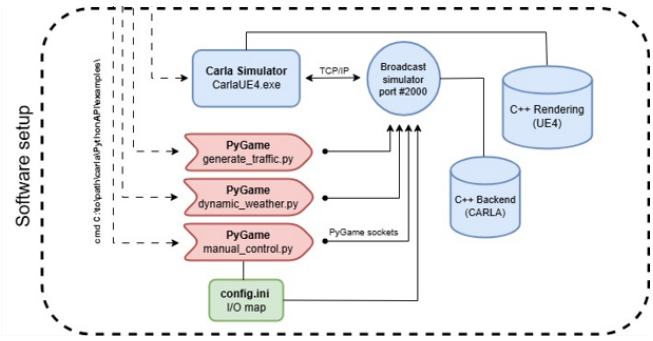

Fig. 12: Software dataflow diagram

For reference and ease of understanding, Figure 12 illustrates the data flow between the various software components used in the simulation and setup. Each process will be explained in detail in the following sections.

A. Launching CARLA Unreal Engine

Launching the CARLA Unreal Engine is a crucial step in initializing the simulation environment. First, navigate to the directory where CARLA is installed and locate the `CarlaUE4.exe` file (for Windows) or the equivalent executable file (for Linux). Double-click the executable to start the Unreal Engine-based simulation. Upon launching, the engine will load the default environment and assets necessary for the simulation. It is recommended to verify that the GPU and CPU are not overloaded during this process to ensure smooth performance.

In certain scenarios, it may be necessary to run `CarlaUE4` with different configuration settings, such as reducing the resolution, to ensure smoother performance or accommodate hardware limitations. To modify these settings, navigate to the `Config` folder within the CARLA installation directory and locate the `DefaultEngine.ini` file. Open this file using a text editor, such as Visual Studio Code, and adjust the resolution parameters under the `[SystemSettings]` section. For instance, you can reduce the resolution by modifying the values for `r.ScreenPercentage` or setting a custom resolution using `r.SetRes`.

Alternatively, if you are launching CARLA from the command line, you can specify configuration settings directly by appending arguments to the executable. For example, use the following command to set a "low" resolution:

Carla Commands

```
CarlaUE4.exe -quality=Low
```

These adjustments can help optimize the simulation for lower-spec hardware or reduce the processing load during complex scenarios. After making changes, restart the simulator to apply the new settings. Always ensure that any modifications align with the project's requirements to maintain an effective simulation environment. Below Figure 13 is CarlaUE4.exe during runtime.

Fig. 13: CarlaUE4.exe

B. Launching Python APIs

To utilize CARLA's Python APIs, begin by ensuring that the Python environment is properly configured with all necessary dependencies installed. Navigate to the CARLA installation directory and locate the `PythonAPI` folder, which contains a variety of scripts and examples to interact with the simulator.

First, verify that the Python version matches CARLA's compatibility requirements (e.g., Python 3.7 or 3.8 is often recommended). Install the required dependencies using the following command:

Carla Commands

```
pip install -r PythonAPI/examples/requirements
```

Once the dependencies are installed, you can start running example scripts or creating your custom Python programs. To execute an example script, navigate to the `PythonAPI/examples` directory and use the following command:

Carla Commands

```
python spawn_npc.py
```

To interact with the CARLA simulator using Python APIs, several scripts are provided to create and customize simulation scenarios. One of these is `generate_traffic.py`, which is used to populate the simulation with dynamic non-player characters (NPCs) such as vehicles and pedestrians.

Another useful script is `manual_control_steeringwheel.py`, which allows the user to control a vehicle in the simulation using a steering wheel and pedal setup. Launching this script with the command `python manual_control_steeringwheel.py` enables real-time vehicle control, providing a hands-on experience for the driver.

Lastly, the `dynamic_weather.py` script can be used to adjust environmental conditions within the simulation. Executing `python dynamic_weather.py` introduces dynamic changes in weather, such as rain, fog, or varying light conditions, adding realism and variability to the simulation scenarios. Each of these scripts offers unique functionalities, making them essential tools for testing and refining simulation experiments.

Ensure that the CARLA simulator is running before executing any Python scripts. The Python APIs communicate with the simulator through a TCP connection, so both must be active for proper functionality. You can modify these scripts or create new ones to implement custom scenarios, sensor configurations, or data collection pipelines, tailoring the simulation to your project requirements.

Carla Commands

```
python generate_traffic.py
python manual_control_steeringwheel.py
python dynamic_wheather.py
```

C. Launching Zephyr Bioharness 3

To connect to the Zephyr BioHarness 3, the required files must be downloaded from the GitHub repository available at https://github.com/Saurya0401/CBSvC.git. After downloading, ensure that all contents are unzipped in the root folder alongside CARLA. Then, navigate to `src/zephyr` and execute `run.cmd`. Ensure that the BioHarness belt is turned on and that the blue light is flashing on and off, indicating that it is broadcasting its Bluetooth address. The `run.cmd` script will use the Bluetooth connection and the MAC address of the belt to establish a direct connection.

Below Figure 14 is the command to connect zephyr BioHarness 3, before running any other PythonAPI scripts. The MAC, and Bluetooth adress needs to be set correctly inside the script in order to establish a successful connection.

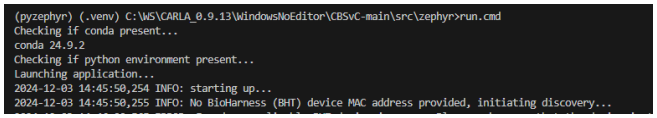

Fig. 14: Connecting to Zephyr BioHarness 3

In the same downloaded GitHub folder, `CBSvC`, there is a PythonAPI script designed to collect data from the Zephyr BioHarness 3 and display it on the driving interface for the CARLA simulation. To execute the script, navigate to the `src/zephyr` directory and run the following command:

Execution Command

```
cd /Carla-root-folder/CBSvC
python -m src.driving.zephyr_stream
#Stream data from the connected belt

cd /Carla-root-folder/CBSvC
python -m src.scenarios.scenario
<scenario_name>
# Run the specified simulation
scenario

python -m src.driving.manual_control
--name <name>
# Enable the belt, run the simulation
scenario, and start the driving
session.
```

This script integrates biometric data into the CARLA simulation environment, enabling real-time feedback and monitoring during the driving session.

D. Python script adjustments for this project

Now that we have covered the basics of running Python scripts to interact with the `CarlaUE4` simulator, we can move on to discussing the adjustments and improvements required to modify the provided scripts and make them suitable for this project's specific requirements. This project, built upon the given PythonAPI scripts by Carla github repository, and utilizes their templates with certain modifications to enhance user experience.

```
834     argparser.add_argument(
835         '--res',
836         metavar='WIDTHxHEIGHT',
837         default='1280x720',
838         help='window resolution (default: 1280x720)')
```

Fig. 15: Connecting to Zephyr BioHarness 3

Above Figure 15 we are adjusting the screen size in PythonAPI Script `manual_control_steeringwheel.py` in terms of WIDTH AND HEIGHT to fit our display. These parameters are dependent on the display connected to the operating PC. For this project the parameters are set to 1280x720. This resolution will provide a single full screen driving simulation view for the driver. However in order to change this to a three screen version, it has to be 3840x720. Furthermore, enabling Nvidia surround will enhance the driving experience as all three screens sync together better for improved driving experience. Below Figure 16 is the option to enable surround display feature. It is only feasible if the computer graphics and specs are optimal.

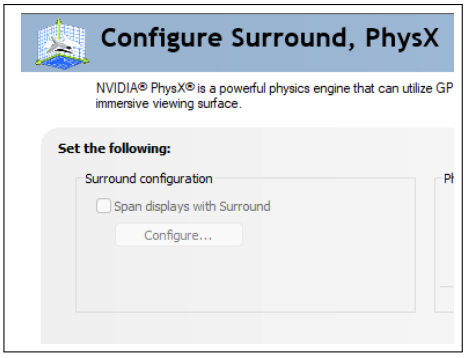

Fig. 16: Nvidia Surround Display

Another important modification that has been made to manual driving Python script is that enabling driver view to be inside of the car, and make sure the car driven is always the same car. The car model is set to tesla model 3. Below is the modification done in the Python script to achieve it.

```
839     argparser.add_argument(
840         '--filter',
841         metavar='PATTERN',
842         default='vehicle.tesla.model3',
843         help='actor filter (default: "vehicle.*")')
```

Fig. 17: Tesla Model 3

```
678         self._camera_transforms = [
679             carla.Transform(carla.Location(x=-5.5, z=2.8), carla.Rotation(pitch=-15)),
680             carla.Transform(carla.Location(x=-8.10, y=-8.4, z=1.2), carla.Rotation())]
681         self.transform_index = 1
682         self.sensors = [
```

Fig. 18: Inside the Car POV

```
331         K1 = 0.55  # 0.55
332         steerCmd = K1 * math.tan(1.1 * jsInputs[self._steer_idx])
333
334         K2 = 1.3  # 1.6
335         throttleCmd = K2 + (2.05 * math.log10(
336             -0.7 * jsInputs[self._throttle_idx] + 1.4) - 1.2) / 0.92
337         if throttleCmd <= 0:
338             throttleCmd = 0
339         elif throttleCmd > 1:
340             throttleCmd = 1
341
342         brakeCmd = 1.6 + (2.05 * math.log10(
343             -0.7 * jsInputs[self._brake_idx] + 1.4) - 1.2) / 0.92
344         if brakeCmd <= 0:
345             brakeCmd = 0
346         elif brakeCmd > 1:
347             brakeCmd = 1
```

Fig. 19: Tuning Steering and Acceleration

Above we further tuned the acceleration and steering sensitivity of the simulating car to enhance more realistic experience for the driver. The values needed to change are k1 and k2 that is used for the gain in control systems equation.

VIII. Automation and User Guide

This section will provide information in regards to how to run everything all together step by step. In addition, there will be automation shortcut applications for windows that will enable to execute the commands without users having to enter. First step is so make sure everything is turned on and ready.

- **Start Zephyr background process**
  - Open VS Code workspace
  - execute conda activate carlaenv
  - cd root folder /CBSvC/src/zephyr
  - Execute ./run.cmd
  - Wait for biometrics to stabilize
  - This step must be completed before starting manual control
- **Check Biometric reading**
  - Run new terminal in the same VS Code workspace
  - cd root folder /Carla_0.9.13/CBSvC
  - Execute python -m src.driving.zephyr_stream
  - Check console output and wait until biometrics are accurate and stable
- **Initialize scenario and traffic**
  - Run new terminal in the same VS Code workspace
  - cd root folder /Carla_0.9.13/CBSvC
  - (Optional) Execute python -m src.scenarios.scenario –help to get info on scenario options
  - Execute python -m src.scenarios.scenario `<scenario_name>`
  - NOTE: Don't add –aggression or –congestion in step 5 as these arguments will override scenario specific settings
- **Initialize Manual Control**
  - Run new terminal in the same VS Code workspace
  - cd root folder /Carla_0.9.13/CBSvC
  - Execute python -m src.driving.manual_control –name `<name>`
  - NOTE: Using the same name multiple times will override previous files.
  - Run each scenario for 10 mins Keep an eye out on the biometrics, sometimes it stops working.

Now there is an easier way to run all this step by a single click. By creating a `.bat` file. The complete `.bat` files will be available to review in Appendix section of this report.

IX. Challenges and Future Improvements

During the Hardware and software setup process, several challenges were encountered:

- **CARLA Installation:** Selecting the appropriate version of CARLA for the given PC specifications was a significant challenge in this project. While the setup initially appeared functional, even minor changes to the Python scripts or hardware components often resulted in crashes and runtime issues. Ensuring compatibility between CARLA and the system's hardware/software configuration required extensive testing and fine-tuning to achieve stability and smooth operation.
- **Cable Management:** Organizing and securing the numerous cables for monitors, the steering wheel, pedals, and the motion platform required careful planning to prevent tangling and ensure safe operation. Additionally, maintaining a clear and accessible seat entrance was crucial to allow drivers to comfortably sit down and exit without obstruction caused by cable clutter.
- **Motion Platform Setup:** Positioning and securely mounting the motion platform under the racing chair required precise alignment and calibration to deliver accurate motion feedback during simulations. Feedback from test users proved invaluable for identifying areas of improvement in the setup and for refining the PhysX motion feedback of the chair.
- **VR Integration:** Setting up the VR headset involved configuring the Wi-Fi network and resolving connection issues with the host PC to enable display mirroring and Quest Link functionality. Successful connectivity required a shared LAN or Wi-Fi network, and ensuring this configuration was critical to achieving seamless VR integration with the simulation system.
- **Three-Screen View Integration:** Integrating a three-screen view posed a significant challenge due to the high GPU rendering requirements. The additional computational load caused frequent crashes and degraded performance during simulations. Optimizing the GPU settings and scaling down graphical details were necessary steps to ensure smoother operation, but further hardware upgrades may be required to fully support the demanding rendering needs of a multi-screen setup.

X. Conclusion

In this report, we have presented a comprehensive overview of the methods, tools, and outcomes related to our project. The integration of reality testbed setup, including the use of CARLA for simulation and analysis, has demonstrated significant potential in achieving our objectives. Key findings highlight the effectiveness of our approach in optimizing performance and ensuring robust functionality within the specified parameters. Furthermore, the modular design and scalability of our implementation provide a solid foundation for future enhancements, such as incorporating additional parameters or extending the framework to more complex scenarios.

To facilitate reproducibility and further exploration, the complete implementation, including code and documentation, is available in the following GitHub repository: CARLA_2024 Project.

XI. Contributor Background

A. Puya Fard

Puya Fard is a Master of Science candidate in Computer Engineering at the University of California, Irvine, where he conducts research under the supervision of Prof. Salma Elmalaki in the Pervasive Autonomy Lab. His current research focuses on building a human-aware automotive testbed using CARLA and a virtual reality environment. His areas of interest include embedded systems, machine learning, and IoT. He has a strong academic background with a Bachelor of Science in Computer Engineering from California State University, Fresno, and professional experience in web design, teaching, and autonomous systems development.

B. Dr. Mahmoud Elfar

Dr. Mahmoud Elfar is a postdoctoral scholar at the University of California, Irvine, working with Prof. Yasser Shoukry in the Resilient Cyber-Physical Systems Lab (RCPSL). He earned his Ph.D. in Computer Engineering from Duke University and has extensive research experience in cyber-physical systems, human-robot interaction, and reinforcement learning. Prior to academia, Dr. Elfar worked in industry as an embedded software and R&D engineer, where he developed innovative tools and systems for companies such as Valeo and Schneider Electric.

C. Prof. Salma Elmalaki

Prof. Salma Elmalaki is an assistant professor in the Department of Electrical Engineering and Computer Science at the University of California, Irvine. She directs the Pervasive Autonomy Lab, focusing on integrating human factors into cyber-physical systems. Prof. Elmalaki earned her Ph.D. from UCLA and has received numerous accolades, including the NSF CAREER Award and the Early Career Teaching Excellence Award. Her research spans embedded systems, human modeling, and machine learning, with a goal of enhancing performance, reliability, and fairness in autonomous systems.

Appendix A
Source Code for Default Driving -Low-Res CARLA

```
@echo off
echo Starting CARLAUE4 in low quality...
cd C:\WS\CARLA_0.9.13\WindowsNoEditor
start CarlaUE4.exe -quality-level=Low

echo Waiting for 30 seconds before starting
    the next script...
timeout /t 30 >nul

echo Starting traffic generation script in a
    new window...
cd C:\WS\CARLA_0.9.13\WindowsNoEditor\PythonAPI
   \examples
start cmd /k C:\Python\Python37\python.exe
    generate_traffic.py

echo Waiting for 10 seconds before starting
    the manual control script...
timeout /t 10 >nul

echo Starting manual control with steering
    wheel in a new window...
start cmd /k C:\Python\Python37\python.exe
    manual_control_steeringwheel.py

echo Waiting for 10 seconds to ensure smooth
    completion...
timeout /t 10 >nul

echo All tasks completed!
pause
```

Appendix B
Source Code for Default Driving CARLA

```
@echo off
echo Starting CARLAUE4 in mid quality...
cd C:\WS\CARLA_0.9.13\WindowsNoEditor
start CarlaUE4.exe

echo Waiting for 30 seconds before starting
    the next script...
timeout /t 30 >nul

echo Starting traffic generation script in a
    new window...
cd C:\WS\CARLA_0.9.13\WindowsNoEditor\PythonAPI
   \examples
start cmd /k C:\Python\Python37\python.exe
    generate_traffic.py

echo Waiting for 20 seconds before starting
    the manual control script...
timeout /t 20 >nul

echo Starting manual control with steering
    wheel in a new window...
start cmd /k C:\Python\Python37\python.exe
    manual_control_steeringwheel.py

echo Waiting for 20 seconds to ensure smooth
    completion...
timeout /t 20 >nul

echo All tasks completed!
pause
```

Appendix C
Source Code for Driving with Biometrics CARLA

```
@echo off
echo Starting CARLAUE4 in mid quality...
cd C:\WS\CARLA_0.9.13\WindowsNoEditor
start CarlaUE4.exe

echo Waiting for 30 seconds before connecting
    to Zephyr biometric belt...
timeout /t 30 >nul

echo Connecting to Zephyr biometric belt...
cd C:\WS\CARLA_0.9.13\WindowsNoEditor\
   CBSvC-main\src\zephyr
run.cmd

echo Waiting for 30 seconds before starting
    the traffic generation script...
timeout /t 30 >nul

echo Starting traffic generation script in a
    new window...
cd C:\WS\CARLA_0.9.13\WindowsNoEditor\PythonAPI
   \examples
start cmd /k python generate_traffic.py

echo Waiting for 10 seconds before starting
    the manual control script...
timeout /t 10 >nul

echo Starting manual control with TestRun
    parameters in a new window...
cd
    C:\WS\CARLA_0.9.13\WindowsNoEditor\CBSvC-main
start cmd /k python -m
    src.driving.manual_control --name "TestRun"

echo All tasks completed!
pause
```

References

[1] Guidance for Institutions and IRBs. U.S. Department of Health and Human Services (HHS). https://www.hhs.gov/ohrp/regulations-and-policy/requests-for-comments/guidance-for-institutions-and-irbs/index.html. Accessed: 2023-12-05.

[2] How Does ISO 26262 Address Open Source Software? SRES.AI. https://sres.ai/functional-safety/how-does-iso-26262-address-open-source-software/. Accessed: 2023-12-05.

[3] ISO 21448:2019 - Road Vehicles – Safety of the Intended Functionality. International Organization for Standardization (ISO). https://www.iso.org/standard/62996.html. Accessed: 2023-12-05.

[4] Logitech G Driving Force Racing Wheel. https://www.logitechg.com/en-us/products/driving/driving-force-racing-wheel.html. Accessed: 2023-12-05.

[5] Meta Quest 2. https://www.meta.com/quest/products/quest-2/. Accessed: 2023-12-05.

[6] Next Level Racing Motion Platform V3. https://nextlevelracing.com/products/next-level-racing-motion-platform-v3/. Accessed: 2023-12-05.

[7] P7006 - Standard for Personal Data Artificial Intelligence (AI) Agent. IEEE. https://www.standict.eu/standards-repository/ieee-p7006-standard-personal-data-artificial-intelligence-ai-agent. Accessed: 2023-12-05.

[8] IEEE 29148:2018 - Systems and Software Engineering – Requirements Engineering. IEEE Standards Association. https://standards.ieee.org/ieee/29148/6937/. Accessed: 2023-12-05.

[9] Zephyr BioHarness BTLE Echo Module Gen 3. https://www.habdirect.com/product/zephyr-bioharness-btle-echo-module-gen-3/. Accessed: 2023-12-05.