The Development of Self-Driving Vehicle Software via Autonomous Driving Simulators

A Technical Report submitted to the Department of Mechanical and Aerospace Engineering

Presented to the Faculty of the School of Engineering and Applied Science

University of Virginia • Charlottesville, Virginia

In Partial Fulfillment of the Requirements for the Degree

Bachelor of Science, School of Engineering

Anne Forrest Butler

Spring Semester 2022

Capstone Project Team Members

Chet Kleppin

Johnny Grant

Andrew Lin

Mosed Saroor

Casey Welch

On my honor as a University Student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments

Signature anne forrest butler Date 02 May 2022

Anne Forrest Butler

Approved Tomo Furukawa Date 02 May 2022

Tomonari Furukawa, Department of Mechanical and Aerospace Engineering

1 Introduction

1.1 Relevance

There is no doubt that the integration of autonomy in modern technologies is a rampant trend among most sectors of the world's industries today. Autonomous systems have been implemented in manufacturing to ease the burden of menial and potentially dangerous tasks, in the commercial world to disrupt human-to-human interactions in light of a global pandemic, and most relevantly to this topic, in the automotive industry to free our ever-busy hands and minds while commuting in cars. Surely autonomy is prevalent in a plethora of other fields, but it is in these sectors where ethical dilemmas sprout from underneath every effort at its implementation into technology. As human oversight is disconnected from the functions of these technologies, there arises concern in how they will behave in situations which commonly require the complicated nature of human intuition and decision-making. In general, the dilemma our society faces as autonomy spreads throughout the world is how it will be safely integrated into organizations and human daily life. There are many ways to navigate the process of fostering public trust in autonomous systems, but the particular one of interest in this context is simulated studies. Specifically, virtually simulating autonomous automobiles and how they interact with their environment. Simulation software such as this can allow self-driving vehicle manufacturers to gain insight on how the algorithms their autonomous systems utilize hold up in potential real-world scenarios. While manufacturers can alternatively just test their technology in the real-world, this often incurs undesirable risks and consequences, such as harm to vehicle operators, pedestrians, and other third parties. An autonomous driving simulator would give these companies a platform in which they can safely test and develop their systems. Additionally, simulators can also serve to familiarize consumers with operating autonomous vehicles. As

self-driving vehicles become more prevalent throughout the world, it's important that the common user feels comfortable riding in an autonomous car. Experiencing it first in a simulated environment would be an effective way to ease consumers into familiarity with this technology. With these concerns in mind, we aim to develop an autonomous driving simulator as a valuable resource for self-driving vehicle production and familiarization.

1.2 Review of Past Work

Pre-existing research into the development of autonomous driving simulators shows that similar approaches to our problem have been attempted, but none fully address the goals we seek to accomplish. Nonetheless, these serve as valuable examples by which we can guide our own research and development. Each of the referenced studies can be classified into one of two approaches: a primary focus on the user experience and immersivity or a primary focus on external factors and functions, such as simulated traffic or pedestrians. With respect to the first kind of approach, a 2015 study conducted by Baltodano, et. al. documented the development of the Real Roads Autonomous Driving Simulator (RRADS). Their research aimed to gather the overall public attitudes and expectations of how an autonomous vehicle is intended to feel. The data was then used to inform the ongoing development of their simulator, and specifically to maximize the immersivity of their design. Other studies that have sought to address immersivity and user experience include Koo et. al. (2014), which was concerned with user understanding of safety features; Bellem et. al. (2016), which sought to assess whether or not moving-base driving simulators induce a comfortable and realistic sense of motion; and Häuslschmid et. al. (2017), where it was found that indicator displays showing the autonomous algorithm actively processing external data fostered greater user trust. The findings of these studies serve to inform

our own development, as we can gauge what consumers seek in a simulated autonomous experience.

Several other studies align more with the second approach we've established, where there's a greater focus on external functions in the simulated environment. In 2017, You et. al. published a paper that detailed their research into generating realistic virtual environments for autonomous driving simulators. They developed a system which took non-realistic virtual image input and processed it into realistic scene structures. Similar studies include Chao et. al. (2020), which documented research into reconstructing detailed traffic flows using real-time traffic data with traffic simulation models; and Pérez-Gil et. al. (2022), where the use of deep learning algorithms was proposed as a means to create more effective simulated autonomous algorithms.

Based on our preliminary research and the input of our customers' needs and specifications, we've established these primary objectives:

- Construct an immersive, realistic user environment. The cockpit should resemble the actual interior of a car to the greatest extent possible.
- Develop the simulator to be fully operable in a manual functioning mode. This will allow the user to swap between autonomous and manual mode, which is a vital function of actual autonomous vehicles.
- Generate a realistic virtual environment. The user should be able to drive reasonable distances and interact with other moving entities in a realistic city-like landscape.
- Create a bounded, encompassing display.

- Mitigate latency between user input and haptic/visual response to be negligibly perceptible.

These objectives are intended to guide our development toward our end goal and satisfy our customers' expectations.

In order to accomplish our objectives, we've broken down our process into three main stages:

1) Design and research
    a) Customer input
    b) Concept generation and selection
    c) Research into previous studies
    d) CAD of necessary components
    e) Gathering background on software and other tools
    f) Ordering parts and components
2) Development
    a) Installation and familiarization with necessary software
    b) Acquiring and installing components for chassis and frame
    c) Constructing bounded display
    d) Generating code for vital functions in simulation software
3) Testing
    a) Benchmark testing of software CPU/GPU usage
    b) Latency testing for input and response
    c) Long-term assessment of structural integrity

d) Assessment of user immersivity and general experience

2 Essential Knowledge

There are three past teams from Virginia Tech who have worked on developing an autonomous driving simulator prior to our team. In 2018, Virginia Tech's capstone team was focused on developing their driver's space, an enclosure to surround the MOOG platform, and a display system. For their driver's space, the team chose to utilize a gaming seat as their driver's seat and to incorporate an actual dashboard into their design along with their steering wheel. For their enclosure, the team chose to construct a frame out of aluminum bars and then use polycarbonate boards connected with aluminum sheets to create walls around the platform. For their projection system, the group chose to utilize one curved monitor and one projector placed at the back of the enclosure. The team chose to use a similar software setup to the original 2017 team, who utilized OpenDS as their software for rendering graphics simulation, ROS as their merging platform between OpenDS and Gazebo, and Gazebo as their robust physics simulation.

In 2019, Virginia Tech's capstone team decided to focus on creating an HD mapping software for their simulator, improving the driver space, and enhancing the software algorithm's efficiency. The team chose to utilize the same interior, enclosure, and projection system as the 2018 team. The primary issues the team faced regarding the driver space were unnecessary vibrations created by the projector and a lack of ability to adjust the position of the driver seat. The team also chose to utilize the same software setup as the 2018 team, however they found that their graphics were consistently lagging and that maintaining the multi-platform communication

was difficult. The 2019 team recommended for future improvements to the system, to utilize a software system called DSpace in order to reduce system lag and improve graphic rendering.

Both the 2018 and 2019 teams chose to utilize OpenDS and Gazebo due to the built in physics engine within OpenDS and the ROS compatibility included within Gazebo. However both softwares' graphics are outdated and our team experienced issues finding source code to enhance the software's environments. The projector system utilized by both teams were more immersive than the original monitor system utilized by the 2017 team. However, by attaching the projector to the outside of the enclosure, the teams experienced issues with the projector causing vibrations that could be felt within the driver space.

Our team's goal is to develop a fully operating autonomous driving simulator that at a minimum, matches the capabilities of commercially available simulators. We are going to focus on achieving this goal by making our simulator more immersive than the previous efforts made by past teams. Our immersiveness is going to be developed via the usage of a newer software system entitled CARLA, the utilization of a realistic vehicle interior, and the creation of a CAVE projection system.

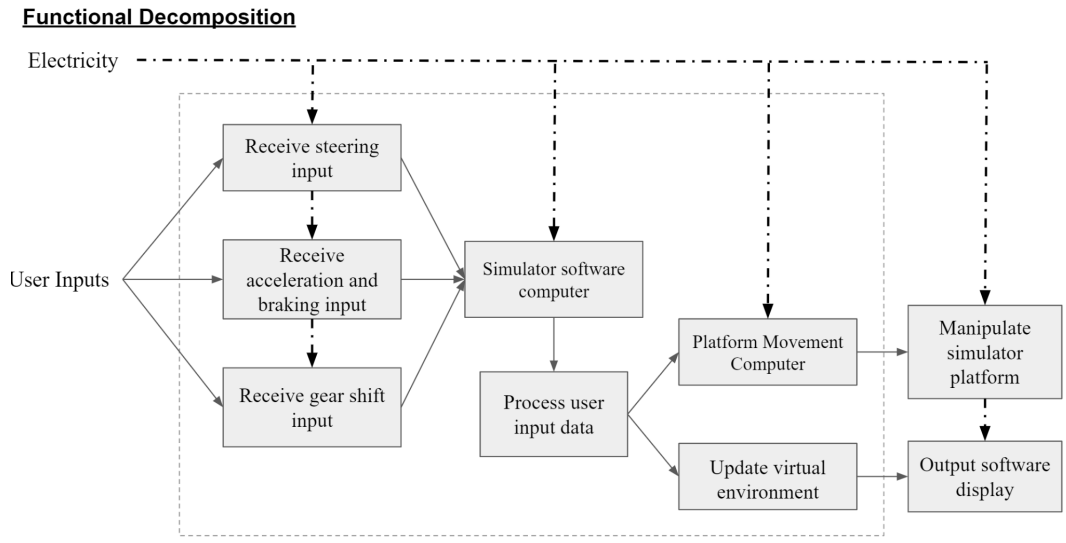
3 Design Process

3.1 Customer Needs and Target Specifications

The customers have all expressed that they want their experience within the driving simulator to be more immersive through various enhancements. Based on the five specific needs that they outlined, we created five target specifications. First and foremost, the customers have expressed that they want less delay in the steering and pedal responses. In order to accomplish this, we developed the target specification to mitigate the latency between the steering input and

the visual response to ten milliseconds. Next, the customers said they wanted to feel like they were in a real car. In order to accomplish this, we developed the target specification of creating the driver space using the interior of an actual car. The customers also said they wanted the graphics to be more steady as the simulator moved. In order to make our display system more stable, we created the target specification of fixing our projector system to the ceiling rather than the simulator's enclosure. This would ensure the projection was not affected by the simulator's movements. The customers also stated that they wanted a spacious driving environment that still included all necessary driving features. We decided that the target specification of creating a driver space using the interior of an actual car would guarantee us the necessary space and set up to accomplish this objective. Lastly the customers expressed that they wanted the driving space to revolve around autonomous designs. They want the driving space to have an aesthetic appeal and functionality that models current autonomous vehicle trends. In order to achieve this objective, our team developed the target specification of integrating a visual aid like an ipad into the interior of the car.

3.2 Concept generation



With these target specifications in mind, our team was able to develop three solutions using the functional decomposition illustrated above. We broke down our system into various sub-functions and identified which functions had various potential solutions. The sub-functions we identified with various potential solutions were our graphics display, our visual environment software, and our gear shift inputs. In all of the solutions we identified a wheel as our steering input, actuators for platform movement, ROS as our communication protocol, Gazebo as our software to actuate movement, a 240V outlet as our power source, and a button to initiate an emergency stop. In solution one, we would utilize automatic gear shifts, DSpace for our environment software, and monitors for our display system. In solution 2, we would utilize a manual gear shifter, virtual reality for our display system, and DSpace for our environment software. In solution three, we would utilize an automatic gear shifter, a projector for our display system, and openDS for our environment software. A chart of all three solutions and their sub-functions and their various subfunctions is illustrated below.

Sub-Functions	Solution 1	Solution 2	Solution 3
Steering Input	Wheel	Wheel	Wheel
Gear shift Inputs	Automatic Stick	Manual Stick and Clutch	Automatic Stick
Output software display	Monitors	VR	Projectors
Platform movement	Actuators	Actuators	Actuators
Communication Protocol	ROS	ROS	ROS
Software to visualize environment	OpenDS	Dspace	OpenDS

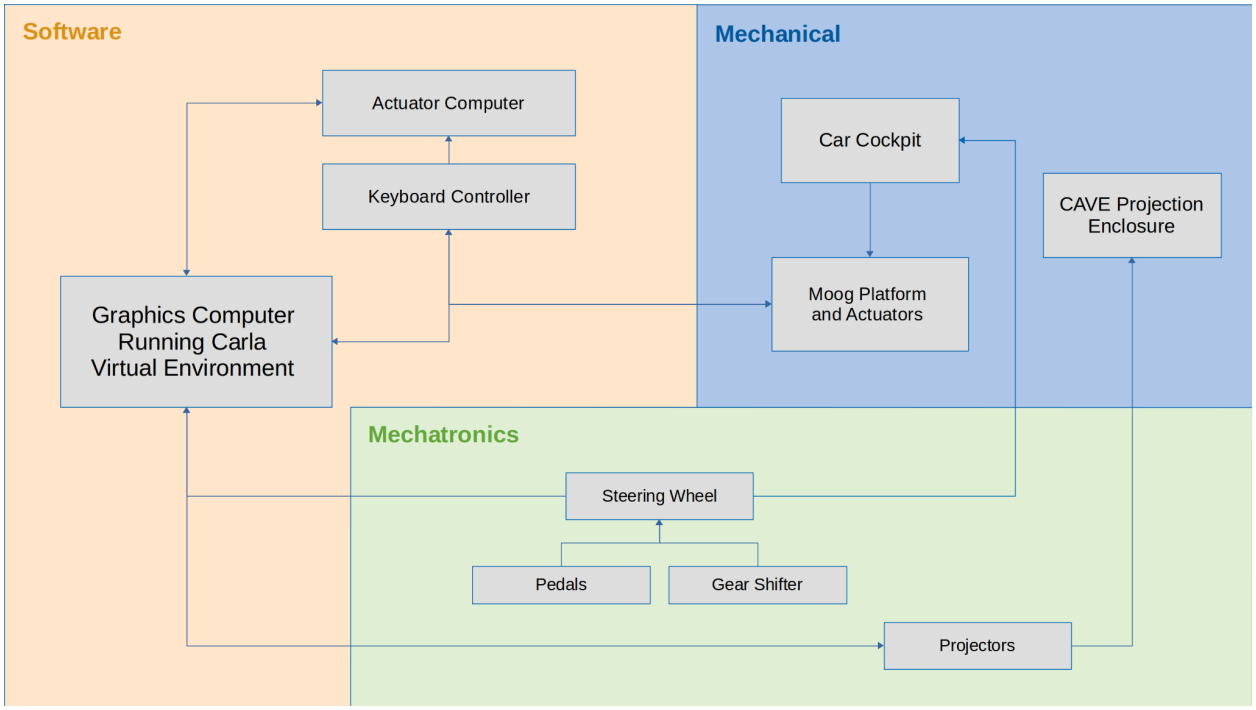
Software to actuate motion	Gazebo	Gazebo	Gazebo
Power Source	240 V Outlet	240 V Outlet	240 V Outlet
Emergency Stop	Button	Button	Button
Programming Language	Python/ Java	Python/ Java	Python/ Java

3.3 Concept selection

We ended up selecting solution three due to its score on the Screening and Scoring and its ability to meet all target specifications and customer needs. Solution three had potential as well, however we ended up deciding that the use of projectors rather than monitors would more effectively achieve the customer's desire to feel like they were in a real car. Furthermore, with the usage of projectors our team realized we could develop a CAVE projection system, which would allow for the simulator to have 270 degrees of display and really enhance the immersiveness of the experience.

However, as the project progressed, we altered from our original concept and decided to try out an updated environment software called CARLA. CARLA was more effectively able to fulfill the customer need for a low latency between steering input and visual response. Furthermore, through CARLA, the latency between user inputs and the motion platforms' response could be lowered to a maximum delay of three ms.

4 Final Design



The final design of the system is categorized into three sectors: mechanical, mechatronics, and software. Each category overlaps and interfaces with each other category. The arrows in the diagram above show the simple relationships between each component and the following subsequent sections will elaborate on the interfacing systems.

Beginning with the mechanical portion of the project, the final design utilizes a cockpit enclosure of a 2009 Subaru Forester mounted to a MOOG motion base platform seen in the figure on the right. The Subaru Forester was selected due to its simple dashboard display which provided us the space necessary to install our own steering wheel,

pedals, and gear shifter while still adhering to the space constraints we were given.

The mechatronic components of our design include the user controls and display devices. This area of our design essentially involves interfacing the hardware with our simulation software. The cockpit is centered within our CAVE projection system, which incorporates three short-throw projectors to create an immersive view of the simulated environment as seen in the diagram on the right. The projectors have been arranged and installed to achieve a range of view of about 200 degrees. The user controls consist of a Logitech steering wheel, set of foot pedals, and stick shifter. These controls are installed in the positions where the original components were stripped from the vehicle and can be seen in the figures below. The steering wheel consists of several buttons which allow the operator to adjust certain parameters of the virtual environment, such as the point of view, autonomous/manual driving mode, and other essential functions. All of the user controls are connected directly to the computer running CARLA and control movement within the simulation software.

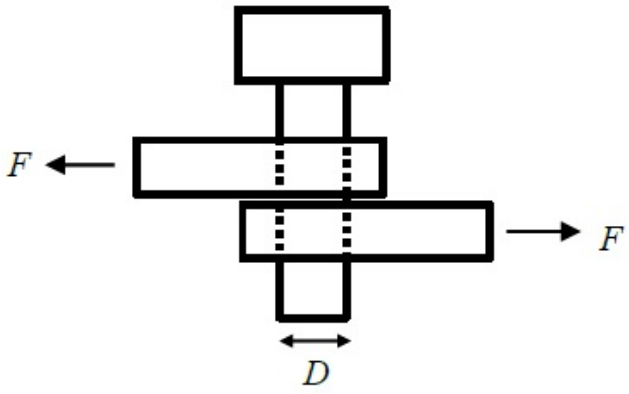
The final design of the software portion starts with the python scripts running through the CARLA software. There are various scripts including synchronous driving, multiple manual

drivers, steering capabilities with the keyboard or steering wheel, generating traffic and pedestrians, etc. Each driving script incorporates various CARLA API's to function along with sending commands to the MOOG in order to send it to certain locations. There are also many different driving features ranging from large changes such as an autonomous driving feature to smaller details such as being able to use the headlights. This allows for the system to showcase an advanced system while also not forgetting the little details that make the entire experience more realistic and immersive. For the steering wheel, because Ubuntu are incompatible with Logitech wheels, many drivers were either created or found in order to make them function seamlessly in the system by providing options such as autocenter, force feedback, spring friction, gain, etc.

5 Mathematical/Numerical Analysis

5.1 Mechanical

The main goal of mechanical numerical analysis was to assure the safety of the chassis. Because the frame of a car was mounted onto a moving platform, two modes of failure could potentially occur that must be mathematically predicted. Firstly, static failure could occur due to plastic deformation of any of the parts associated with bolting. The kinds of static failure that are possible are compressive failure resulting from the bolts stress concentrations on the platforms or shear failure of the bolts. To analyze the designed system two different single shear stress equations can be used. Double shear stress will not be used because no bolt passes through the MOOG, wood, and car chassis. Rather, bolts either pass through the MOOG and wood, or wood and car.



$$\boldsymbol{\tau} = \frac{F}{A} = \frac{F}{\pi r^2}$$

Half inch bolts are used as the geometric criteria of the equation. A force can be determined assuming 400 pounds for two people and 600 pounds for the chassis of the car. At a five degree tilt angle, the applied force is 87 lbs. With five bolts from the chassis to the wood, and a minimum shear strength of a ½ inch bolt being 11,000 lbs a safety factor of 632 is achieved.

Knowing that there are more bolts from the wood to the MOOG and because of Newton's Second Law, the same force is applied in this scenario. However, it is distributed across more bolts thus failure would occur between the chassis and the wood and not the MOOG and the wood. Next, static failure could occur due to stress concentrations in the wood because it is the weakest link in the scenario. Again, the applied force of 1000 lbs and a number of bolts is five, dand the documentation of the plywood's minimum compressive strength is 4500 psi. This results in a safety factor of 51. The real potential mode of failure will be fatigue failure from motion, cycles and vibrations.

Based on a solidworks static analysis using an applied force of 87 lbs at a five degree angle of tilt, the bolt was well within the minimum shear stress which was in agreement with the

above analysis of the bolted system. Using that static analysis, a fatigue study was run with the same force value using fully reversed loading. The results of the analysis showed there was no damage to the bolt and that the stress placed on the bolt was below the applicable values on the SN-curve. This is understandable as the system is running at a low degree of tilt so the applied force being distributed among the bolts is minimal. This is further supported by the safety factor calculated above for the shear force.

Hexadecimal to Decimal Conversion Chart

Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal	Hexadecimal	Decimal
1	1	41	65	81	129	C1	193
2	2	42	66	82	130	C2	194
3	3	43	67	83	131	C3	195
4	4	44	68	84	132	C4	196
5	5	45	69	85	133	C5	197
6	6	46	70	86	134	C6	198
7	7	47	71	87	135	C7	199
8	8	48	72	88	136	C8	200
9	9	49	73	89	137	C9	201
A	10	4A	74	8A	138	CA	202
B	11	4B	75	8B	139	CB	203
C	12	4C	76	8C	140	CC	204
D	13	4D	77	8D	141	CD	205

5.2 Software

MOOG only takes in binary information so in order to communicate with it, we have to send in hexadecimal information from the python scripts. First, a 5 percent range of tilt is calculated based on the total range of motion the MOOG can perform. Once finding out the decimal number amount equivalent to 5 percent of tilt, we can convert it into hexadecimal numbers through an online converter. This process is repeated for each actuator in order to find the entire command to send and the commands are written as the following:

cmd = xff/x82/x1F/x3F/xFF/x3F/xFF/x3F/xFF/x3F/xFF/x3F/xFF/x29/x00

Determines what the MOOG does. 82 means new position, B4 engages, etc
Checksum: All hexadecimals needs to add up to this
Individual actuators. Actuator A. 1st hexadecimal is low byte, 2nd is high byte

6 Experimental Validation

Parametric study and experimentation was conducted to test many different parameters. Firstly, with safety of the simulator being of utmost importance, safety experiments were run. Aggressive python scripts that move quickly and go to angles much more extreme than will be used in simulation were run. By doing this, the experiments confirmed that the chassis of the vehicle was properly mounted to the MOOG. No shifting or any signs of failure occurred at any bolt locations. The MOOG platform does move slightly, but does not shift significantly or dangerously. By testing the MOOG to its extremes with the chassis on it and people in the car, experimentation determined safety during simulation.

Other parameters that were experimentally tested include immersivity. By sitting in the car and visually testing all points of view and angles the screens are all fully enclosed and immerse the driver. This parametric experimental testing identified areas for improvement. The frame rate of the display ranged from eight frames per second to fifteen. The human eye being able to see up to 60 fps, a newer and stronger graphics card is recommended to maximize immersivity. Also, minor adjustments to display angles could remove display gaps.

Other parametric experimentation included user input testing and autonomous driving. User inputs respond appropriately during experimentation with a latency of 4 ms**.** User inputs are interpreted and sent to the MOOG in an appropriate manner in accordance with a vehicle motion model, however the MOOG outputs in a jerky manner leaving room for algorithm improvement in that area. Also, CARLA's default autonomous vehicle algorithm was tested and situationally self-driven effectively. Being open source, another area for improvement would be in making a

new self-driving algorithm or even partnering with and testing another company's proprietary software.

Additionally, the addition of the new simulated environment software has allowed for the addition and prospect of improved autonomous driving. The new software CARLA has additional sensors that were not readily available nor user friendly on previous softwares utilized by past groups.

7 Operations Manual

In order to operate the system with the MOOG motion platform, first connect the MOOG USB and keyboard from the computer located in the bottom of the MOOG system to the computer located outside of the CAVE system. Turn on the both switches on the MOOG computer located on opposing sides of the computer. One appears as a plastic toggle switch and the other appears as a small metal lever toggle switch. Plug in the green plug to the wall outlet and the MOOG computer power plug to the power strip. This starts up the MOOG system and the monitor on the right side should appear with all the actuators location and the sound of the computer should sound like a fan turning on. If both the monitor is not on and the sound of a fan is not present, not all the plugs and switches have been engaged. If you want to test with the keyboard go to part 1). If you want to test with CARLA or with other python scripts go to part 2).

Make sure all projectors are turned on for the CAVE system and also check that displays are in the correct order if you plan to showcase the CAVE design. This can be done in settings in the displays section.

*Note: Any sudo command's password is 'password'

*Note: Carla does not need the MOOG to run and if you don't want the MOOG on, leave it off but still complete each step excluding the step 1 in part 2.

Part 1) MOOG Keyboard controls

If wanting to test individual actuators, click 'q' and type in '6D2MAINT' and once the screen turns back to the actuators, click 'e' and then hold the '+' or '-' to lift or drop the MOOG respectively. In order to lift the MOOG with desired actuators, click either '1', '2', '3', '4', '5', and/or '6' to lock certain actuators then use the previous '+' and '-' buttons to move the MOOG. Other commands in the mode are listed within the MOOG manual binder.

Part 2) CARLA Setup/MOOG

In order to set up the MOOG

1) click 'q' and type in '170-122' on the MOOG connected keyboard/monitor.
    a) This prepares the MOOG to take in serial commands from python scripts on the other computer.

Turn on the other computer outside the CAVE system and login to drivesim user. The password is 'password'.

2) In the directory /home, type in these commands in order
    a) sudo su
    b) cd /
    c) cd dev
    d) chown <username> ttyUSB0
        i) <username> is 'drivesim' for our computer
        ii) Leave this terminal tab open and open a separate tab

Open up the terminal and type in the following command to turn on CARLA in any directory.

3) sudo docker run --privileged --gpus all --net=host -e DISPLAY=$DISPLAY carlasim/carla:0.9.13 /bin/bash ./CarlaUE4.sh
    a) If you run into this error: sh: 1: xdg-user-dir: not found
        i) Red herring error. Don't worry about this
    b) If you do not want the screen open, add the following to the end of the command '-RenderOffScreen' so it should look like sudo docker run --privileged --gpus all --net=host -e DISPLAY=$DISPLAY carlasim/carla:0.9.13 /bin/bash ./CarlaUE4.sh -RenderOffScreen'
    c) If you want to load it in low quality mode, add the following to the end of the command '-quality-level=Low'

d) Do not close this terminal and open new terminal tabs for other setup and scripts

Part 3) Steering Wheel Setup

With either steering wheel, plug in all corresponding plugs. The gear shifter, pedals, power source, and USB to the computer. Go to the section for the steering wheel you decide to use.

*Note: The connection between the Logitech G920 and the integrated gear shifter is not compatible so a new steering wheel may be needed. The gear shifter works with the old Logitech G27 wheel but the wheel has issues with clicking random buttons at random intervals. We recommend using the Logitech G920 as it still has all functionality, but it loses some of the immersiveness.

Logitech G920

Currently the driving scripts are made for the Logitech G920. In order to set up the steering wheel to work with the scripts, go into the downloads/usb_modeswitch directory and type in the following command

4) sudo usb_modeswitch -c /etc/usb_modeswitch.d/046d:c261
    a) This should say the switch was successful and you can check by typing in the following command 'jstest-gtk' to see if there is a Logitech G920 joystick.
    b) If you want to change button configuration on the steering wheel later, 'jstest-gtk' helps you keep track of the button layout.

In order to get force feedback for the Logitech G920 type in the following command which should open an application

5) oversteer
    a) More documentation can be found at https://github.com/berarma/oversteer
    b) At the top, go to the tab for the wheel and slide the autocenter force feedback slider at the bottom to desired feedback. Our group set it around 30. Check if the steering wheel is no longer slack.

Logitech G27

This was the previous steering wheel which was the Logitech G27. The modeswitch is unnecessary for this wheel. Can check and configure the steering wheel buttons with the command 'jstest-gtk' to map buttons on the gear shifter.

In order to fix the force feedback issue, refer to step 5 in the Logitech G920 section. If that does not work, there is a workaround. In the directory Documents/PythonAPI/new-lg4ff, type in the command ‘sudo make load’ and in the directory /sys/bus/hid/drivers/logitech/0003:046D:C29B.0005, open the autocenter file and change it to approximately 40000. More documentation can be found at https://github.com/berarma/new-lg4ff.

*Note: Depending on the pedals you use, you may need to change the wheel.ini file in the Documents/PythonAPI/examples directory to fix the brake and accelerate buttons.

Part 4) Python Scripts

There are multiple scripts within the Documents/PythonAPI/examples directory that run with CARLA. For the main one, if you want to drive with the keyboard, type in the following command in the Documents/PythonAPI/examples directory

6) python manual_driving.py
    a) This allows you to run the system with just the keyboard. Clicking ‘h’ will drop down some settings that CARLA has including but not limited to, autonomous driving, sensor data, radar visualization, switching sensors, car views, etc.
    b) In order to see sensor and car data, you must render with the screen in step 3 in part 2 as that is where the data is shown.

In order to run the system and drive with the steering wheel, type in the following command in the Documents/PythonAPI/examples directory

7) python demo.py
    a) If the MOOG will not work with the script, try ‘python manual_driving_newsteering.py’ instead.
    b) If you want to see CAVE view, type in the command ‘python demo_cave.py’
        i) This will open a resizable window and you need to stretch the pygame window across the 3 projectors to make the CAVE immersion.
        ii) This script is very slow with the current hardware so we recommend using it only when showing off the CAVE functionality or once a better GPU is purchased.
    c) The buttons have been labeled on the wheel with different icons but functionality can also be found in the code.

d) This has most of the functionality of the manual_driving.py script but allows driving with the wheel.

If you want to generate traffic for when you are driving, type in the following command in the Documents/PythonAPI/examples directory

8) python generate_traffic.py
    a) Sometimes the traffic may be frozen so you may need to close the scripts and rerun them.
        i) Soft fix:
            (1) Restart CARLA
            (2) run ./synchronous.py
            (3) run ./generate_traffic.py
            (4) run ./manual_control.py
            (5) close synchronous and generate_traffic
            (6) rerun ./generate_traffic.py

Part 5) Code and documentation

The CARLA scripts utilize CARLA's driving simulator's APIs in order to function. Much of there documentation can be found at the following link

9) https://carla.readthedocs.io/en/latest/python_api/
    a) This documentation also covers various sections for troubleshooting the software and working with it.

For MOOG movement, we sent commands directly to the platform using hexadecimals. This is found in the IMUSensor section of the code and can be improved upon to smooth out the movements of the simulator. In order to understand what each byte does in the hexadecimal command, the MOOG manual binder explains it. For our commands, each one works a separate actuator but in the future, a degree of freedom mode will help it be more exact.

Also, to improve communication, a ROS bridge and ROS can be connected to the CARLA software. Documentation and steps for this can be found on the CARLA website

10) https://carla.readthedocs.io/projects/ros-bridge/en/latest/

Part 6) Disassembling

To turn off the system, just unplug all MOOG outlet connections and turn off both switches. Turn off the computer and make sure the projectors are off before leaving. If you completely shut down the computer versus putting it to sleep, each step will need to be redone.

Troubleshooting

- If it says '# USB permission denied' when running a script, it means the MOOG serial commands have not been connected and to fix this, redo step 2 in part 2.
- If you load up the script and it says missing joystick 0, it means the steering wheel has not been configured correctly in part 3. Redo step 4 and 5 in part 3.
- If running with the MOOG and when running the driving scripts, the MOOG platform moves up then has a communications error, try resetting the MOOG.
    - Repeat step 1 in part 1, then run the script again.
    - If it still doesn't work after multiple attempts, try the other scripts in subsections of steps 7 of part 4.
- If the python scripts within the directory Documents/PythonAPI/examples are not working because there is no connection, check if CARLA is opened.
    - You can do this by typing in the command 'docker container ls' and checking if there is a CARLA application opened. If there is, restart it by removing it and reopening it using 'docker rm -f <ContainerID>' and then step 3 in part 2.

Miscellaneous

- If you want to see the other CARLA files within the docker container, type in 'docker exec -t -i <ContainerID> /bin/bash'
- There is also an old version of CARLA downloaded if you chose to use that instead
    - sudo docker run -e SDL_VIDEODRIVER=x11 -e DISPLAY=$DISPLAY -v /tmp/.X11-unix:/tmp/.X11-unix -p 2000-2002:2000-2002 -it --gpus all carlasim/carla:0.9.10 ./CarlaUE4.sh -opengl
- There is a gazebo environment if you chose to use that driving simulator instead in the directory 'desktop/environment'.

8 Conclusions and Future Work

Our team has made significant progress towards reaching our goal of developing an autonomous driving simulator over the course of the past two semesters. We were able to create a fully enclosed and immersive driver space, incorporate a new virtual environment software in

order to decrease the latency between the graphics and our steering inputs, and develop a CAVE projection system with a 270 degree range of projection. We have had to make significant changes to our design over this year, resulting in our progress being slower than predicted. However, we were able to expedite our progress this semester and in turn remain on schedule and accomplish our established goals.

Future work for the mechanical side of the project includes mostly aesthetic appeal and safety improvements. The projection onto the right wall could be improved by constructing a wall similar to the wall utilized for the left screen. The right wall has a number of pipes running across it and the projector for the right wall is slightly blocked by overhead ducts and in turn can not fully project onto the wall. Building a taller wall that's closer to the projector could improve this. Furthermore, the placement of the MOOG platform could also be adjusted in order to ensure it's positioned on a more stable surface. Right now, we have it positioned on top of metal grates due to the lack of space we have in our lab. These metal grates cause the Moog to shake during operation, create loud noises, and lack a proper place to be bolted. Future work to be accomplished regarding software includes enabling a ROS-bridge with ROS in order to send Carla sensor information to the MOOG platform. Once we are able to send information to the MOOG, the platform should move in all 360 degrees corresponding to the physics of the virtual car.

Works Cited

Baltodano, Sonia & Sibi, Srinath & Martelaro, Nikolas & Gowda, Nikhil & Ju, Wendy. (2015). The RRADS platform: a real road autonomous driving simulator. 10.1145/2799250.2799288.

Bellem, H., Klüver, M., Schrauf, M., Schöner, H.-P., Hecht, H., & Krems, J. F. (2017). Can We Study Autonomous Driving Comfort in Moving-Base Driving Simulators? A Validation Study. Human Factors, 59(3), 442–456. https://doi.org/10.1177/0018720816682647

Chao, Q., Bi, H., Li, W., Mao, T., Wang, Z., Lin, M.C. and Deng, Z. (2020), A Survey on Visual Traffic Simulation: Models, Evaluations, and Applications in Autonomous Driving. Computer Graphics Forum, 39: 287-308. https://doi.org/10.1111/cgf.13803

Häuslschmid, Renate & Bülow, Max & Pfleging, Bastian & Butz, Andreas. (2017). Supporting Trust in Autonomous Driving. 319-329. 10.1145/3025171.3025198.

Koo, J., Kwac, J., Ju, W. et al. Why did my car just do that? Explaining semi-autonomous driving actions to improve driver understanding, trust, and performance. Int J Interact Des Manuf 9, 269–275 (2014). https://doi.org/10.1007/s12008-014-0227-2

Pérez-Gil, Ó., Barea, R., López-Guillén, E. et al. Deep reinforcement learning based control for Autonomous Vehicles in CARLA. Multimed Tools Appl 81, 3553–3576 (2022). https://doi.org/10.1007/s11042-021-11437-3

You, Y., Pan, X., Wang, Z., & Lu, C. (2017). Virtual to Real Reinforcement Learning for Autonomous Driving. ArXiv, abs/1704.03952.