2023

Framework for Data Acquisition and Fusion of Camera and Radar for Autonomous Vehicle Systems

Clay Edward Vincent
West Virginia University, cev0001@mix.wvu.edu

Framework for Data Acquisition and Fusion of Camera and Radar for Autonomous Vehicle Systems

Clay Vincent

Thesis submitted to the Statler College of Engineering and Mineral Resources at West Virginia University in partial fulfillment of the requirements for the degree of

Master of Science in

Electrical Engineering

Brian Woerner Ph.D., Committee Chairperson,

Andrew Nix, Ph.D.

Daryl Reynolds, Ph.D.

Department of Computer Science and Electrical Engineering Morgantown, West Virginia

2023

Keywords: Data Acquisition, SAE Level 2, Verification and Validation, Adaptive Cruise Control, Hybrid Electric Vehicles, Autonomous

Abstract

Framework for Data Acquisition and Fusion of Camera and Radar for Autonomous Vehicle Systems

Clay Vincent

The primary contribution is the development of the data collection testing methodology for autonomous driving systems of a hybrid electric passenger vehicle. As automotive manufacturers begin to develop adaptive cruise control technology in vehicles, progress is being made toward the development of fully-autonomous vehicles. Adaptive cruise control capability is classified into five levels defined by the Society of Automotive Engineering. Some vehicles under development have attained higher levels of autonomy, but the focus of most commercial development is Level 2 autonomy. As the level of autonomy increases, the sensor technology becomes more advanced with a sensor suite which includes radar, camera, and vehicle-to-everything radio. Sensors must detect the objects around the vehicle to be able for communicate the data to the adaptive cruise control algorithm. If a vehicle is in an accident, the driver is typically responsible for the damages, but with an autonomous vehicle, there might not be a driver. A process to guarantee a vehicle will perform as it was developed is critical to a vehicle's development and testing. The goal of this work is to implement a verification and validation system that can be used on adaptive cruise control systems. The system developed in this paper used different testing environments such as model-in-the-loop, hardware-in-the-loop, and vehicle-in-the-loop, to fully validate an autonomous vehicle. A systematic data acquisition process has been developed to support autonomous vehicle development. The data that was taken had an organized way of comparing the results in each environment. Requirements management, vehicle logbook, and test case creation played a vital role in combining the information across the environments. The method produced a consumer-ready adaptive cruise control system in a 2019 Chevrolet Blazer RS. The vehicle was able to perform at an Advanced Vehicle Technology Competition where the adaptive cruise control system placed 1st in Connected and Automated Vehicle Perception System & Adaptive Cruise Control Drive Quality Evaluation. Results are presented that illustrate the utility of the data acquisition and multi-layer testing process for autonomous vehicle development.

Acknowledgements

I would like to thank my faculty advisor and committee chair, Dr. Brian Woerner, for his continued support and guidance throughout my academic career as a graduate research assistant for the WVU EcoCAR team. Dr. Woerner supported and answered questions whenever needed. His devotion to the team as well as the education of his students ensures that everyone is performing at his best to accomplish each task given. In addition, I would like to thank Dr. Andrew Nix, the head faculty advisor for the WVU EcoCAR team. The amount of time he dedicated to the team is unforgettable. He was always available to assist in any aspect of the research the team completed throughout the year and his wide range of automotive research continues to impact new students each year. I would also like to thank Dr. Daryl Reynolds, a committee member who has taught me signals and systems throughout my undergraduate and graduate years at WVU. He has influenced me and the way I think to approach communication problems at my time with the WVU EcoCAR team.

I would like to thank the WVU EcoCAR General Motors mentor Dr. William Cawthorne for his involvement with the team. His expertise in the automotive industry was invaluable to the team. Throughout the two years that I was able to work with Dr. Cawthorne, I accomplished so much more than I could have imagined working on autonomous vehicles as well as moving into a project management role for my second year of graduate school. He scheduled meetings with the team to help troubleshoot any part of the vehicle with which the team was having difficulties.

Lastly, I would like to thank my fellow graduate students in the WVU EcoCAR program. My first project manager, Benton Morris, offered numerous ways to solve a problem when I was working as the lead for the CAVs team. Secondly, I would like to thank my previous engineering manager Aaron Mull, who spent countless nights in the lab teaching me new ways to approach problems as well as systems engineering processes. Thirdly, I would like to thank the third year's CAVs team, as well as, my co-lead Alexander Colon

for helping me start the process of testing our algorithms for adaptive cruise control and sensor fusion. After that, I would like to thank the fourth year's CAVs team and their leads, Jared Diethorn and Zachary Flanigan, for their continued support of my process for software testing. These two individuals were able to use the system that was created in year three and provide feedback to refine the system for future use on the EcoCAR team. Collin Kellett, Dawson Dunnuck, and Holden Fraser, along with Diethorn and Flanigan, made my last year as project manager of the EcoCAR Mobility Challenge enjoyable and a highlight in my academic career. Under their leadership the team successfully build a level 2 SAE Hybrid Electric Vehicle with the WVU EcoCAR team.

Dedication

My thesis is dedicated to my parents and family who provided unwavering support throughout my academic career. They were always there to keep me on track and provide me with positive reinforcement on difficult decisions that I needed to make. I was taught a strong work ethic can complete any goal in life no matter how big or small the goal may be. Without their dedication and motivation this paper would not be possible.

Contents

Table of Figures

List of Tables

List of Acronyms

ACC	Adaptive Cruise Control
AVTC	Advanced Vehicle Technology Competition
CAN	Controller Area Network
CAVs	Connected and automated vehicles
DOE	Department of Energy
eAWD	Electric all-wheel drive
FOV	Field of View
GRA	Graduate Research Assistant
HEV	Hybrid Electric Vehicle
HIL	Hardware-in-the-Loop
HMI	Human Machine Interfacing
HSC	Hybrid Supervisory Controller
ICE	Internal combustion engine
IoT	Internet of Things
kW	Kilowatt
MIL	Model-in-the-Loop
MRR	Mid-Range Radar
m	Meters
ms	Milliseconds
mph	Miles per Hour
mps	Meters per Second
m/s^2	Meters per Seconds Squared
PCM	Propulsion Control and Modeling
PHEV	Plug-in Hybrid Electric Vehicle
PSI	Propulsion System Integration
RMD	Requirement Management Documentation
SAE	Society of Automotive Engineering
SIL	Software-in-the-Loop
UDP	User Datagram Protocol
V2I	Vehicle to Infrastructure
V2X	Vehicle to Everything
VIL	Vehicle-in-the-Loop
WVU	West Virginia University

1.0 Introduction

The objective of this research was to design and implement a testing strategy for autonomous vehicle applications. The main goal of the implementation was to verify and validate the requirements set in each software environment such as model-in-the-loop (MIL), hardware-in-the-loop (HIL), and vehicle-in-the-loop (VIL). The reason for the research was to build a standard work documentation for future WVU EcoCAR students to be able to test their algorithms in those different environments safely before being deployed on the road. This research focused on connected and autonomous vehicle (CAV) systems which included a supervisory algorithm for data collection and parsing, a sensor fusion algorithm and an adaptive cruise control algorithm. The WVU EcoCAR team was tasked with designing a level 2 SAE autonomous vehicle with donated components such as a mid-range radar, camera, and vehicle-to-vehicle and vehicle-to-infrastructure radio.

This work explains the methodology to track test cases in each testing environment to satisfy all the requirements that were developed before the algorithms were designed. The following pages delineate the ways in which the WVU EcoCAR team tested autonomous vehicle systems in each environment and the software that was used to validate the adaptive cruise control system in the vehicle. The adaptive cruise control algorithm as well as the sensor fusion algorithm were designed for a level 2 SAE system but the procedure at which the test was completed can extended to different levels of autonomous driving.[1]

1.1 Hybrid Electric Vehicles

Hybrid electric vehicles (HEV) are currently being produced by many manufactures to extend the range of a vehicle that is normally powered by an internal combustion engine (ICE). These vehicles provide a reduced carbon footprint that consumers can find appealing when using the vehicle. A hybrid electric vehicle or HEV contains an internal combustion engine as well as an electric motor to produce propulsion. Electric motors operate in a variety of ways to reduce emissions. Regenerative braking is an advantage to

HEVs as well as adaptive cruise control systems. The electric motor will provide negative torque to an axle of the vehicle to capture energy while the vehicle is decelerating. Adaptive cruise control can focus on this aspect of hybrid vehicles more than an ICE only vehicle.

1.1.1 Hybrid Vehicle Classification

A plug-in hybrid electric vehicle (PHEV) and a non-plug-in hybrid electric vehicle are the two classifications of the hybrid vehicle. A non-plug-in hybrid has small battery packs that allow them to work with an internal combustion engine to provide propulsion. The electric battery will not power the vehicle in electric only operation as the battery charges through regenerative braking. A PHEV has large high voltage batteries that can be charged by plugging the vehicle into a vehicle charger or a standard 120-volt outlet. The high voltage batteries can propel the vehicle in electric operation only and can also work with the internal combustion engine to provide propulsion to the vehicle. In addition, the batteries can also charge using regenerative braking as well.

1.1.2 Hybrid Electric Vehicle Architecture

Series, parallel, and series-parallel hybrids are the three different architectural types of HEV. First, a series hybrid occurs when the engine does not drive the power to the wheels. This means that the engine is connected to an electric generator that serves the purpose of charging the high voltage batteries. The electric motor provides the power to the wheels similar to a battery electric vehicle. Secondly, in a parallel architecture, the ICE and electric motor are working together to provide propulsion to the vehicle. In a parallel architecture the battery size is smaller in comparison to the other configurations which means the vehicle relies on regenerative braking to keep the battery charged. The IC engine is not mechanically coupled to the HV battery for charging which reduces the loss of converting mechanical energy to electrical energy for the electric motor to use and then back to mechanical energy to provide propulsion for the vehicle. Lastly, the series-parallel configuration combines the two architectures and can use the

engine and motor to either propel the vehicle separately or at the same time. At lower speeds such as city traffic, the use of the series configuration is beneficial compared to a parallel due to the use of a larger battery usage to enable the vehicle to consume more electric energy than a smaller parallel battery. When the vehicle is operating at highway speeds, the vehicle benefits from a parallel configuration due to the number of parts and systems the vehicle must control when determining the best conditions in which the ICE runs [2].

Each configuration has the benefit of recharging the HV battery with regenerative braking as well as opportunity charging. Opportunity charging occurs when the internal combustion engine provides the difference between the torque commanded from the accelerator pedal and the negative torque that the electric motor is producing which charges the electric motor. The series hybrid (configuration B) is efficient in stop-and-go traffic. The parallel hybrid (configuration A) is efficient at higher vehicle speeds. The series-parallel hybrid (configuration C) takes the best of both architectures so the vehicle can increase fuel economy as well as reduce emissions. [3]–[6]

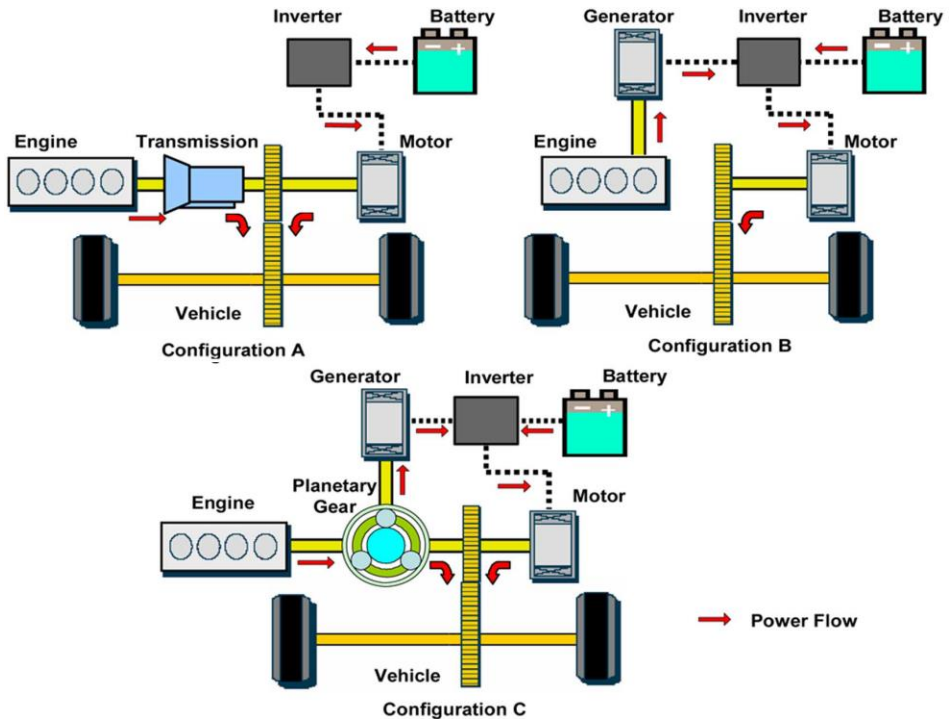


Figure 1: HEV Architecture Parallel (A), Series (B), Series-Parallel (C)

1.2 Society of Automotive Engineer (SAE) Levels

The Society of Automotive Engineers mission is to advance mobility knowledge and solution for the benefit of humanity [7], [1]. SAE provides publications and standards for the automotive industry as well as for automotive collegiate competitions such as advanced vehicle technology competitions, Formula SAE, and Baja SAE. This paper delineates one specific standard to levels of automation which is the J3016 standard for vehicle driving automation systems that perform part or all the dynamic driving task on a sustained basis [1]. The taxonomy outlined in the standard provides six levels of driving automation. It starts with level 0 with no automation to level 5 which is full driving automation. The levels are as follows and shown in Figure 2:

- Level 0: No Driving Automation
- Level 1: Driver Assistance
- Level 2: Partial Driving Assistance

- Level 3: Conditional Driving Automation
- Level 4: High Driving Automation
- Level 5: Full Driving Automation

Within the levels of automation there are two roles-the driver and the vehicle that help determine the level of driver automation based on the actions the two are expected to maintain. At level 0 the driver is tasked with the performance of the vehicle with the entire dynamic driving task even when active safety systems are featured into the vehicle indicating that the vehicle has no role within a trip. A vehicle started and driven to a desired location and turned off again is known as vehicle trip. Level 1 features include driver assistance either in the lateral steering portion of the vehicle or the longitudinal motion of the vehicle. The driver is assisted for a portion of the trip either with adaptive cruise control or parking assistance. In conjunction with level 2, these two levels are a subtask of the dynamic driving task. Level 2 is partial driving automation where the driver supervises the subtask of the lateral and longitudinal control of the vehicle in which case the driver might need to obtain control where automation cannot perform. Level 3 is conditional driving automation and is the first level with driving automation for the dynamic driving task. This level is used for routine or normal based operations while the driver is considered a fallback ready user in case of an emergency. Level 4 is high driving automation of the vehicle. The driver has become removed of almost all duties throughout the entire dynamic driving task that is in the operational design domain. The fallback user is operated by the vehicle as well. Lastly, level 5 is full driving automation. The vehicle is in control of the entire dynamic driving task even when an unconditional performance is not within the scope of the operational design occurs. The driver does not need to supervise level 4 or level 5 driving automation and the vehicle may not include a steering wheel in the vehicle design. A graphic provided by SAE is used as a quick reference guide to help detail the level of automation.

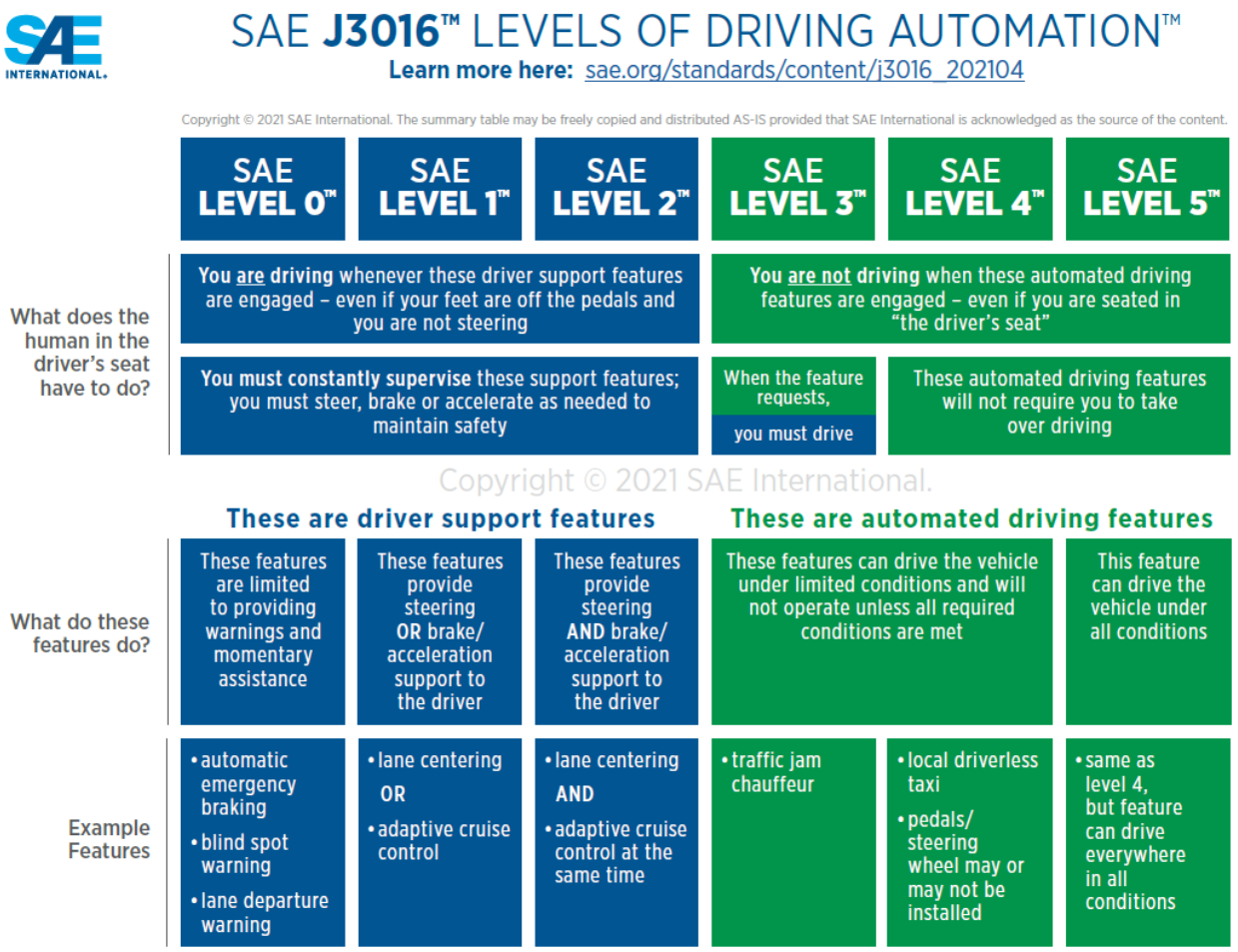


Figure 2: SAE Levels of Driving Automation [8]

1.3 Testing Environments

In this paper three environments were used to develop a parsing, sensor fusion and adaptive cruise control algorithm. In the automotive industry more environments can be used to develop any level of automation, but the most used testing environments are Model-in-the-Loop, Software-in-the-Loop (SIL), Hardware-in-the-Loop, and Vehicle-in-the-Loop. [9]

MIL is the first stage of the testing process that an algorithm will go through. MIL houses a model of the actual hardware that will be used later in the vehicle. The model should capture most of the prominent features of the hardware that each system will have such as a radar or camera. Once the hardware is properly modeled then the development of the control algorithm can be written. The tester will stay in

this testing environment until the controller can control the plant model as per the requirement made earlier in the design phase of the algorithm. When all the requirements are completed, the tester will take notes of the inputs and outputs for later verification.

The next environment is Software-in-the-Loop. Once the controller model is verified in MIL simulation the next step is to generate code for the controller. The generated code will have all the requirements written for the controller and include more for the connection being made in future environments. This testing environment will give the tester an idea of whether or not the controller can be code generated and implemented into the hardware. For verification of the controller the inputs and outputs should be compared to the MIL model to see if the values are acceptable.

The Hardware-in-the-Loop environment occurs when the controller model is run on the hardware that will be used in the vehicle. Digital and analog inputs and outputs such as CAN or ethernet interfaces are physically connected to the hardware which can send deterministic situations that the tester can verify. Any communication problems can show up in this phase as well as instability in the hardware. This environment validates safety critical components in the controller as well as in the hardware.

Lastly, VIL is the final environment before the controller and hardware are ready for full on-road use. This environment provides a safety net to evaluate dangerous maneuvers that require a risk of collision. Using repeated scenarios are tests for validation of the system. This environment is performed on a closed course test track with obstacles or simulated data to provide the vehicle a real time feel of the vehicle. One example is simulating a vehicle using the brakes while a pedestrian walks in front of the vehicle while a crash obstacle such as an inflatable car will test an automated approach to another vehicle. Each step is consecutive and if any step fails the tester can go back to the MIL environment and repeat the process shown in Figure 3.[10]

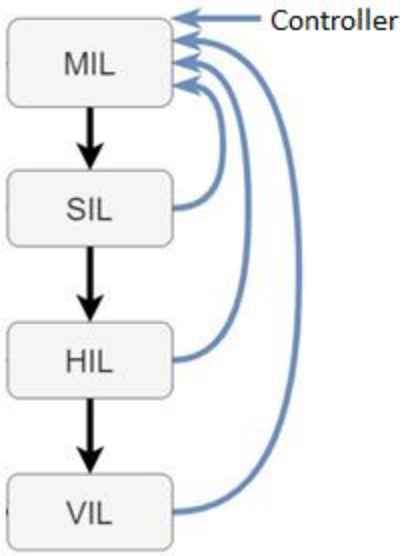


Figure 3: Environment Testing Process

1.4 EcoCAR Mobility Challenge

The United States Department of Energy (DOE) started to sponsor Advanced Vehicle Technology Competitions (AVTCs) in 1988. The DOE partnered with North American automotive manufactures and Argonne National Laboratory to provide a challenging competition as well as a real-world training ground for North America's future automotive engineer, business, program management, and communication students. The opportunities that are featured in the competitions can challenge graduate and undergraduate students beyond the classroom environment. These students are donated a production vehicle which is re-engineered to improve energy efficiency and stay within the emission standards. The students are also tasked with maintaining or improving the performance, consumer acceptability, safety, and cost of the vehicle. As of today, AVTCs have had 20,000 students from 93 educational institutions in North America participate in the competition gaining the necessary real-world hands-on experience to jump start a student's career [11]. The first competition started in 1988 with the Methanol Marathon and culminated with the completion of the most recent competition EcoCAR Mobility Challenge in 2022. There

have been 12 completed competitions so far as number 13 begins in Fall of 2022 with the EcoCAR EV Challenge [12].

The EcoCAR Mobility challenge finished the most recent competition in May of 2022. Starting in 2018, the competition challenged 11 universities to apply advanced connected and automated systems as well as propulsion systems to the donated vehicle. The focus was to improve safety, consumer appeal, and overall vehicle efficiency. General Motors donated each team a new 2019 Chevrolet Blazer and tasked them with redesigning the vehicle into an HEV with level 2 SAE autonomous features. The first year of the competition is the design phase, the second year is the integration phase, the third year is software integration with propulsion controls and connected automated vehicle features, and year 4 is the refinement of all the integration and design of the past years. The scope of the driver automation of year 4 was for the automation hardware to provide the propulsion and braking of the vehicle along with vehicle-to-infrastructure control.

The EcoCAR Mobility Challenge consisted of four technical swimlanes as well as two non-technical swimlanes. The technical swimlanes included propulsion systems integration (PSI), propulsion controls and modeling (PCM), connected and automated vehicles (CAVs), and Human Machine Interfacing (HMI). The subteam tasked with integration of the vehicle components was the PSI subteam. The subteam was split into mechanical and electrical integration. The mechanical group swapped the 3.6-liter engine to the smaller 2.5-liter engine that was donated by GM. The electrical group wired the HV battery and electric motor as well as any switches that were added into the vehicle by the mechanical team.

The PCM team was tasked with designing the control strategy, power moding, component interfacing, and diagnostic systems in the vehicle. Once the team completed the design phase, the subteam integrated the controls into the vehicle and performed the vehicle testing at a closed course for validation. Multiple strategies and designs were developed throughout the years of the competition to increase the fuel

economy of the vehicle as well as the implementation of team designed safety systems. The subteam used the same strategy for designing controllers going through MIL, SIL, HIL, and VIL. Each version of the controller had to pass all the requirements in each environment before the software was implemented into the vehicle for testing. In Year 4 of the competition the organizers required the team to go through 10 hours of closed course testing without any faults before the vehicle was approved for on-road use.

The CAVs team was tasked with designing an adaptive cruise control (ACC) algorithm along with a sensor fusion algorithm with the team added sensors. The subteam was split into a hardware, ACC, and sensor fusion sections. The hardware section was tasked with implementing the team added sensors and CAVs processor. Each section of the subteam worked together to automate the vehicle to level 2 SAE autonomous by combining dynamic sensor data that detected surrounding vehicles. These objects or vehicles were then passed to the ACC system which provided the hybrid supervisory controller with a torque command that can be either affect the ego vehicle by accelerating or braking. If the sensors do not detect anything on the road the ACC system should act as a normal speed controller or a regular cruise control system. In the final two years of the competition, the team was tasked with adding Vehicle to Everything (V2X) radio for the control of Vehicle to Vehicle (V2V) and Vehicle to Infrastructure (V2I). The radios provided information of other vehicles or function as a stop light at an intersection so they can create a network vehicle communication system.

The HMI team started in Year 2 of the competition and was tasked with educating drivers about the autonomous features of the vehicle as well as the team added features for the consumer appeal factor. The HMI team had the creative freedom to design any method that they deemed necessary to teach the required audience about ACC. The ways to educate the driver could be in the form of a video, app, interactive display, or pamphlet. The final design must be used in vehicle while it was stationary to not distract the driver while the vehicle was in operation.

The team was organized with team members on each subteam with graduate research assistants (GRA) leading each subteam. The subteam leaders or GRAs then reported to an engineering manager who overlooked the work from each technical swimlane. In conjunction with the engineering manager, a project manager was on site to assist in scheduling meetings, events, or testing sites. The engineering and project manager then reported to faculty advisors that oversaw the entire project and provided support for the team as needed. The team was also assigned a General Motors mentor who provided guidance to the team as needed as the EcoCAR Mobility Challenge is a learning opportunity for students in the automotive industry. The structure of the WVU EcoCAR team is shown in Figure 4.

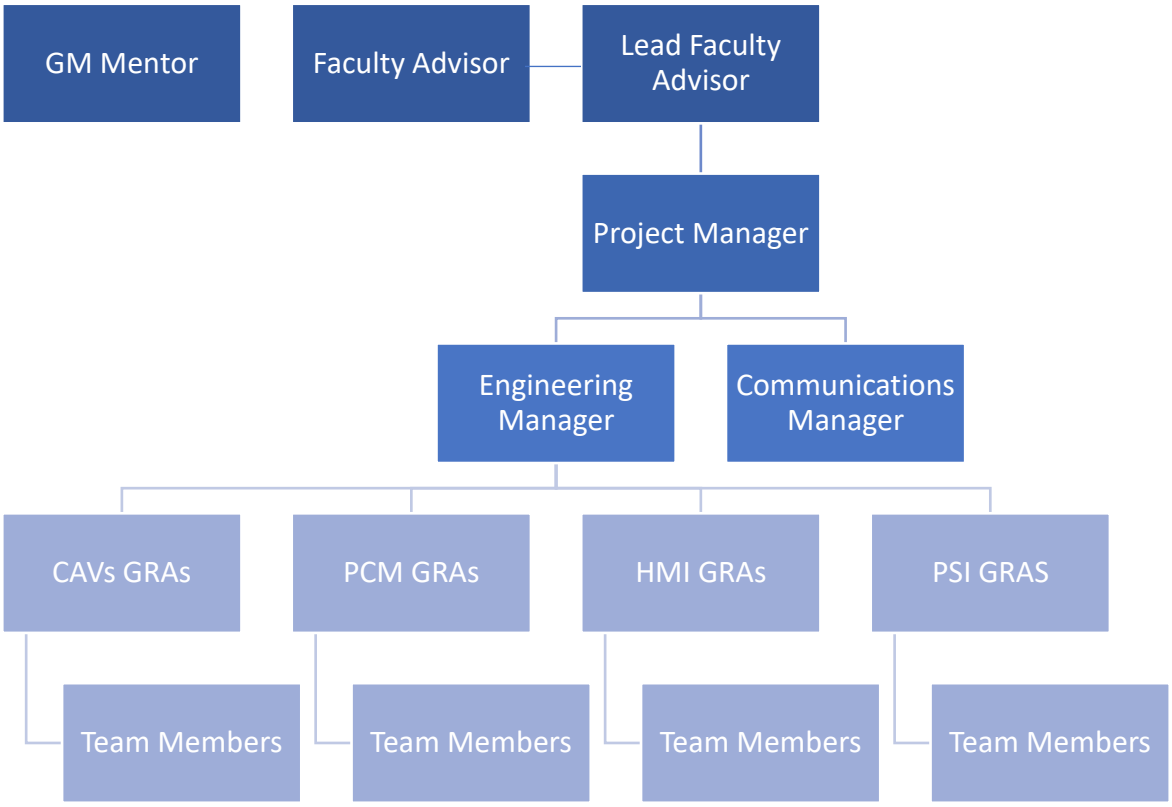


Figure 4: WVU EcoCAR Team Structure

1.5 West Virginia University Vehicle Architecture

Starting in Year 1 of the competition the West Virginia University (WVU) EcoCAR team had two considerations when choosing the architecture for the vehicle. The team focused on fuel economy results

as well as ease for integration. The team deliberated certain HEV classifications and control strategies to improve the fuel economy as well as simulating them to get an idea of the pros and cons associated with them. Since the team was required to have a fully functional hybrid in Year 3 of the competition, the team went with a parallel P4 hybrid architecture. Parallel hybrids can come in 6 different forms from P0 to P5. The team chose the P4 architecture because of the simulation results and integration ease of the electric motor which is mechanically separated from the IC engine. The electric motor provides the torque for an axle while the other axle is propelled by the IC engine.

1.5.1 General Motors Donated Components

At the beginning of the competition, General Motors donated a variety of vehicle hardware. The WVU EcoCAR team selected the 4-cylinder GM 2.5 LCV engine which is rated for a maximum torque of 259 Nm at 4400 revolutions per minute (rpm), and a maximum power of 151 kilowatts (kW) at 6300 rpm and a maximum engine speed of 7000 rpm [13]. The engine was then mated with a GM 9-speed M3D 9T50 transmission. The HV battery for the electric motor was the GM HEV4 battery pack that had a nominal voltage of 300 volts that can store 1.5 kilowatt hours of energy with a peak performance of 50 kilowatts.

1.5.2 Third Party Donated Components

The other propulsion component was the electric motor provided by Magna Powertrain. The electric axle included a differential gearbox and integrated motor for ease of integration which was used in the Volvo V60 hybrid. This system is an electric all-wheel drive (eAWD) unit that can provide 60 kW of peak power as well as a maximum torque rating of 200 Nm with the continuous torque rating of 90 Nm. This unit has a maximum speed of 12000 rpm and combines with an inverter that is made for this eAWD.

1.5.3 Connected and Automated Vehicle Hardware Architecture

Onto the CAVs hardware which provided the torque commands to the propulsion components, the team utilized a Bosch Mid-Range Radar (MRR) and an Intel Mobileye 630 camera for longitudinal control. These

sensors were used to independently detect objects ahead of the ego vehicle. The two sensors provided a level of redundancy for a safe and reliable object detection sensor suite. The Bosch MRR radio has a detection range from 0.36 m to 160 m with a field of view (FOV) of 12 degrees at 160 m, 18 degrees at 100 m, and 20 degrees at 60 m. and a message time cycle of 60 ms and interfaces by High-Speed CAN. The Mobileye camera has a detection range from 0 m to 150 m in daylight for vehicle and 0 m to 40 m for pedestrians. In the dark the camera can detect vehicles up to 90 m using the taillights of vehicles. The FOV of the camera is 38 degrees while the message time cycle can range from 66 ms to 100 ms. This sensor also interfaces by High-Speed CAN.

The primary CAVs processor was the IEI Tank Developer Kit Internet of Things (IoT) Tank-870-Q170. The IoT Tank facilitates the communication of the sensors as well as houses the parsing, sensor fusion and ACC algorithms. The tank has access to analog ports as well as a Kvaser CAN card that can process low and high-speed CAN channels. WVU EcoCAR team's tank had two of these cards which allowed the processor to interact with 8 different CAN channels. The Tank is connected by CAN to the hybrid supervisory controller to transmit ACC torque commands. The hybrid supervisory controller was donated by dSPACE and called the MicroAutoBox II. This processor housed the PCM subteams control strategy as well as the safety subsystems that have been designed by the team.

Cohda Wireless donated a V2X radio, model MK5, to the WVU team. This radio communicates to other vehicles as well as infrastructures. This unit consists of two parts the MK5 box and an antenna that is placed on top of the vehicle. The radio has an operating system of a Linux 4.1.15. It has data rates between 3 – 27 Mbps with a Global Navigation Satellite System that is 2.5 m accurate and is capable of reading and writing data to High-Speed CAN or ethernet interfaces. The team connected the CAVs processor to the radio by the ethernet interface via a user datagram protocol (UDP).

1.5.4 Software Development Tools

During the four years of the competition the team was donated multiple software to test the team designed algorithms. One of the headline sponsors was MathWorks who gave the team access to MATLAB and Simulink technologies. The team utilized this software for designing algorithms and testing the designs in the MIL and SIL environments. The software offers sensor packages as well as basic ACC modules to begin development for on-road use. Once the software moved to the HIL environment the team used a Vector CANoe license to verify the interfaces of the CAVs processor as well as gather real world data from the sensor suite. The team also utilized open source socketCAN programming to send and receive information into C++ code which was the language used in the vehicle.

2.0 Literature Review

The following sections summarize previous work related to the development in autonomous vehicle systems. Various work details ways to execute verification and validation in different testing environments as well as the methodology of testing adaptive cruise control. This section starts with requirements found to be the starting point of all engineering designs [14], [15].

2.1 Requirements

Requirements start the design process for a vehicle system. Each requirement is made to achieve the overall goal of the system that is being designed. Different types of requirements are prepared for each system in the vehicle especially with adaptive cruise control. Requirements, such as functional, performance, safety, and comfort serve as inputs to the overall vehicle design. These requirements are utilized in the technical requirements for the system, a sub-system, or a component that is in the vehicle. Each requirement that is made will be completed at every testing environment and should pass checks before the system is tested at the next level. A requirement tracking system is usually implemented to track certain systems as well as certain requirements based on safety. Safety requirements must adhere to the standard ISO 21448:2022 and ISO 26262-1:2018. The finishing thought when making and verifying all the requirements is “How good is good enough?” [16]. By contrast, our approach integrates safety as one factor in a more comprehensive requirements testing framework [17].

2.2 Adaptive Cruise Control Disengagements

Designing the lower levels of SAE adaptive cruise control must include the ability to disengage ACC based on the requirements of the systems as well as driver disengagements. The higher levels of automation control the vehicle for the whole drive cycle and require no use of a driver. A disengagement must happen when there is a failure in the autonomous technology or when the system cannot operate in a safe manner. These disengagements should be reported as well as logged for the reason of disengagement.

Driver disengagements that are considered routine or common do not have to be reported as this was intended in the design of the system. Disengagements can be divided into passive disengagements and active disengagements. Passive disengagements occur when the system identifies a failure and requires the driver to have immediate control over the vehicle whereas active disengagements involve the driver taking control over the vehicle, but the system does not recognize any failure [18]. ACC disengagements and failures are necessary to track when testing the vehicle in each environment.

2.3 Verification and Validation Methodology

The verification and validation model provides a plan for model-based development using the V-model in Figure 4. This process has been applied to vehicles over the past 20 years and continues to be used today when developing systems and autonomous cars. The left side of the V-model follows the waterfall method approach as testers or observers verify the system while the right side is a bottom-up approach for validation. At any stage of this approach the process may be iterative depending on the failures or problems that may occur while developing or testing the system. Traceability information as used in this research bridges the gap of information exchange between different phases to compensate between real world and bench testing. In reference [19] issues that the verification and validation method may have in the future of autonomous vehicles are presented by an architecture with a monitor and actuator that carries out the V-model. Considering this architecture aligns with an iterative approach shown in Figure 5. The authors in this reference developed a framework for testing of autonomous vehicle control software. However, in contrast to our approach, this thesis deals more with autonomous automobiles, rather than autonomous robots [20], and this one deals with testing of heavy vehicles rather than passenger vehicles [21].

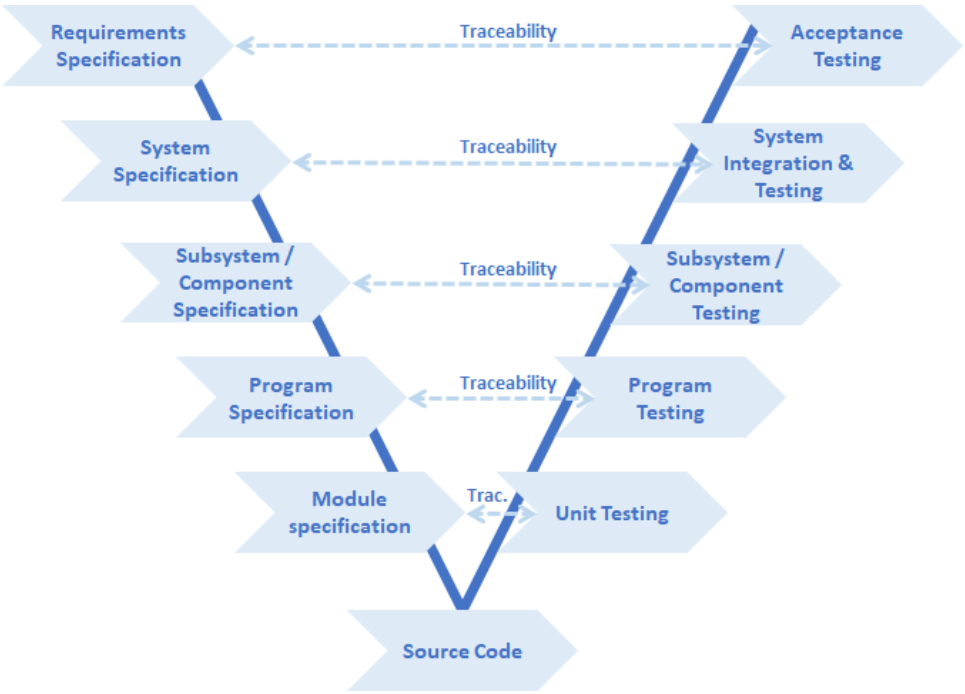


Figure 5: V-model

2.4 Model-in-the-Loop

Model-in-the-loop starts within a simulation environment that models the hardware of the system. In SAE level 2 systems the vehicle may have multiple radars as well as cameras that the plant model is simulating. Many academic studies have used computer generated data for MIL testing [22], in contrast our approach integrates actual measured sensor data. The donated components mentioned above are being modeled for the plant while a controller is being developed to fuse the sensors and control the vehicle's longitudinal motion. The system must verify that each requirement is passed without any failures to move on to the next environment of testing which could be software-in-the-loop or hardware-in-the-loop. The platform used in this paper is MATLAB Simulink to develop the plant and controller models.

2.5 Hardware-in-the-Loop

Hardware-in-the-loop means the controller model is running on the hardware used in the vehicle. The hardware used to control the vehicle in ACC mode is the Intel IoT Tank. The real time systems are tested

with the communication interfaces that the hardware will encounter such as User Datagram Protocol or Controller Area Network (CAN). This environment is important when verifying safety critical features in autonomous vehicles in case of communication failures and to be able to report hardware or ACC failures to record for test cases. In recent years, hardware-in-the-loop testing method has become an integral part of control validation in any product development cycle [23], [24]. A simulation platform is used to simulate the vehicle powertrain in conjunction with the hardware used for ACC. The HIL environment uses as much hardware as possible while still being flexible and easy to repeat tests [25].

2.6 Vehicle-in-the-Loop

The vehicle-in-the-loop environment is designed for validation of the ACC system. If at any time the system fails, the system can iterate back to a previous environment to test the failure. This is the last environment before the system is fully validated. The vehicle must have all the hardware installed and functioning properly for VIL testing. In VIL testing, the environment must be a safe location where the vehicle is free from any accidents. This is usually completed at a closed course testing track that can satisfy the test case requirements. Depending on the requirements of the system, multiple test cases may need to be run to validate the system. VIL allows the system to test critical and dangerous driving situations without the use of pedestrians and vehicles in the environment. This special environment saves time and guarantees functionality while enabling the system to be repeatedly tested and saving time [21], [26].

3.0 Development Tools

The documentation and testing environment methods presented in this section were designed to increase testing performance and information gathered in each stage of verification and validation. The documents were used by the WVU EcoCAR team to verify and validate the sensor fusion and adaptive cruise control algorithms for a level 2 SAE autonomous vehicle. The testing environment for model-in-the-loop was MATLAB Simulink. Hardware-in-the-loop environment connected the Intel IoT Tank to a dSPACE vehicle simulator while real time data was used from the forward-facing sensor suite. Lastly, the algorithms were tested in the 2019 Chevrolet Blazer RS for the vehicle-in-the-loop environment on a closed course track.

3.1 Requirements Management

The beginning of the adaptive cruise control and sensor fusion algorithm development of the requirements for each system were documented in Microsoft Excel to track systems and sub-systems separately. The document contains the requirement number, ID, the location of the source requirement, the requirement priority, and the full requirement statement. This is paired with the validation test case, status, who validated the requirement and when the requirement was validated. An example of a passed validation test is in Table 1.

Table 1: Requirement for ACC on Driver Interface

WestVirginiaUniversity.

Requirements Management Documentation

REQUIREMENT DATA				
System	Sub-system	#	ID	Source
Driver Interface	ACC Controls	893	Driver Interface-ACC Controls-893	STPA

			VALIDATION DATA				
Full Source Reference	Requirement	Swimlane	Validation Test Link	Validation Status	Who Validated	Validation Date	Safety Priority
STPA_CA-1_UCA-03_R02	Cruise control system shall prevent activation above 85 mph	PCM,CAVs	ACC Controls Test	PASSED			1

Figure 6 is the V-Model showing the flow of the designed requirements and testing explained in the next sections.

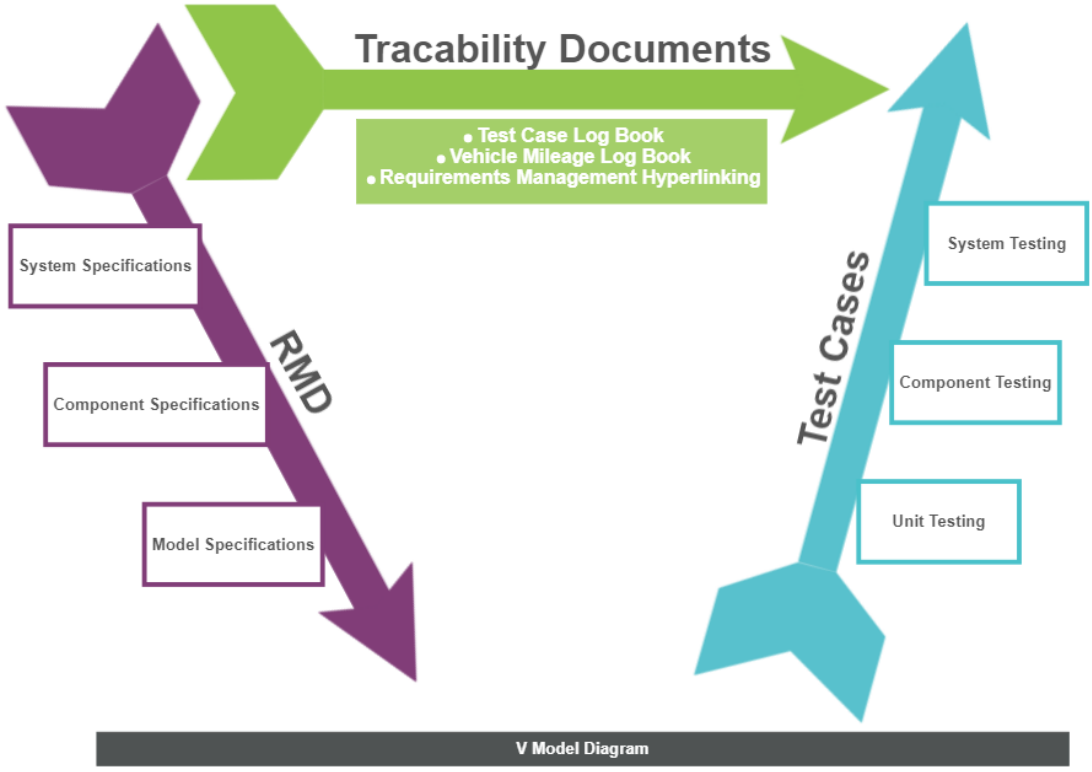


Figure 6: V-model of the WVU EcoCAR Team

3.2 Test Case

The test case creation followed the MIL, HIL, and VIL cycle. Starting with the MIL environment, the test cases were based on the requirements made before the algorithms were created. The test cases showed basic function of the sensor fusion and adaptive cruise control algorithms before communication was added in the HIL environment. The HIL environment had the same test cases as the MIL environment, but it started to incorporate hardware communication between the sensors and Intel IoT Tank. These tests included some on road vehicle data as the dSPACE simulator can model the Chevrolet Blazer. Lastly, the VIL environment included in-depth vehicle dynamics tests that are performed at a closed course test site. These dynamics included following distance and jerk of the adaptive cruise control system to validate the MIL models for higher fidelity.

3.2.1 Test Case Creation

The procedure to create a test case starts with a requirement in MIL. An example requirement such as "Vehicle shall not decelerate greater than 3.5 m/s^2 in ACC without object proximity warnings" can be tested in MIL by manually lowering the set speed of the ACC algorithm. This same test can be saved and updated in the form in the Appendix to show if it passes in each environment respectively. The test case form can be updated to show a failure or pass by extending the sheet and by detailing where the test was located. An example of a test case created for the HIL environment is "Cruise control system should respond to speed increase switch request within 50 ms". This example is a communication requirement that is tested in the HIL phase by watching the output of the ACC algorithm. The system must respond under 50 ms going from the detected objects by the sensors to the sensor fusion algorithm then lastly as an output from the ACC algorithm across CAN. The VIL environment still uses the same test cases from the previous environment which must be passed while also adding dynamic tests as well for model validation. A VIL test case includes "The ACC shall maintain a minimum safe distance of 5m when the lead

vehicle comes to a complete stop." This requirement can be tested in the MIL environment but is validated in the VIL environment at a closed course.

3.2.2 Test Case Tracking

Figure 7 shows the hierarchy in tracking test cases and where information lies within the verification and validation model. The requirements management documentation (RMD) is the first document to track requirements and validation status. The RMD then points to where the test case of the requirements resides within the storage structure in Microsoft Teams. This method can be saved in any file storage saving system if hyperlinking is available. Once the test case is created the hyperlink is updated in the RMD as well as updated in an excel log of all test cases available for future useA folder saving format, such as "TestCaseID"_"YYY"_"ZZ," is used to separate the tests. "YYY" is the description of the vehicle used as the target, and "ZZ" is the ego or target vehicle speed at which the test is conducted such as 10 mph, 10% pedal, or VR for varying speed. This separate log is used for later validation to be able to examine the tests to provide analysis of the system that is being tested. The testing team can also reference the test case creation guide for all the variables that have been created. The test case and vehicle logbook in the Appendix are updated with the links to the separate case files as well as documenting mileage driven, hours of testing, and system safety levels.

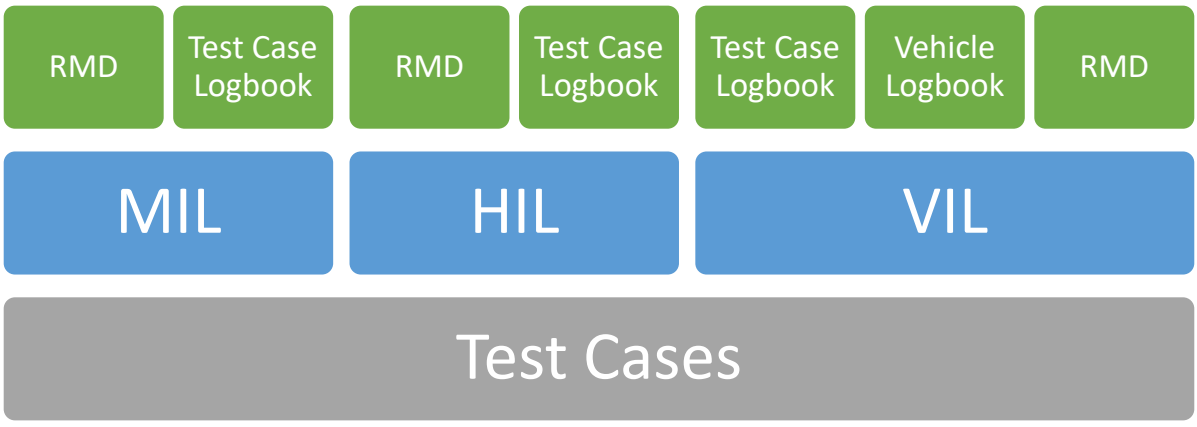


Figure 7: Test Case Tracking Hierarchy

4.0 Engineering Development Process

Throughout the four years of the competition the subteams followed an engineering development process shown in Figure 8.

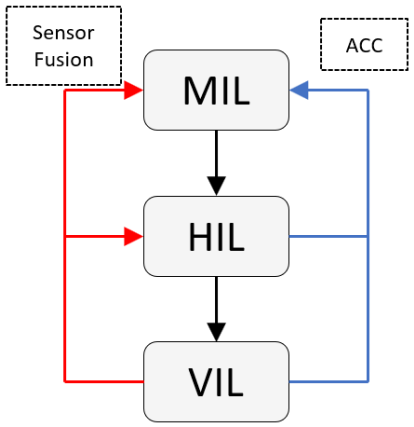


Figure 8: Engineering Development Process

The sensor fusion group followed the left path while the ACC group followed the right path. Each group tested in three environments—MIL, HIL, and VIL. The sensor fusion group had the ability to separate the algorithms into two parts such as the sensor fusion algorithm and the parsing algorithm. This allowed the group to streamline testing between communication networks or the detection algorithm. The separation of the algorithm allowed the team to test the algorithm by jumping back one environment instead of starting back in the MIL testing process. These environments created a rapid prototype development process that allowed the group to debug any problems in each environment.

The ACC group developed in a cyclical cycle starting from MIL to HIL then to VIL. The algorithm started development in the MIL environment where the algorithm consisted of Simulink blocks and MATLAB code. Simulink was able to generate C++ code to allow the algorithm to be tested in HIL and moved into VIL when HIL testing was completed. If any error was encountered at any environment, a note was made in the test case and the group transitioned back to the MIL environment. The algorithm was then tested again with those test cases to see if modifications needed to be made to capture edge cases.

4.1 Test Procedure

The test procedure started with some type of testing that needed to be done based off a requirement or calibration to the system. The WVU EcoCAR team followed the procedure document in the Appendix to streamline the process to ensure the test cases were properly made and submitted. Once the test cases were created, they were approved by the GRA in charge of the subteam. The next step was to have all subteam leaders and GRAs meet to see if the test case was properly made and documented for future use. Once the test case was approved, the engineering manager assigned a naming convention which was then updated in the procedure document and saved to the test log repository. If the test case was already created, then the second process could be skipped since it was logged in the test case repository.

The test case could now be done by a testing team which scheduled a site and time with the project manager. The testing team was then tasked to test all equipment for proper functionality. If a team member was not familiar with the equipment, the team leader was to provide that member with training. The day before testing, the testing team, engineering, and project manager met to assign roles and jobs for the day of testing. The team made sure the test cases were available for each member so the test cases could be filled out for filing later in the test case repository. The vehicle and equipment as stated in the test case was gathered the day before the testing commenced.

When the day of testing arrived, the testing team checked that all the equipment was present, and the roles of the members were clearly defined, and nobody had any questions. The team arrived at the location and proceeded with testing by following the test procedure. Multiple tests could be completed so the team members made sure that the data gathered was organized with each test case. This was done with proper labeling using the naming convention provided by the engineering manager.

Following the completion of testing, the testing team post processed data recorded from ground truth or CAN networks. The data was then converted into usable files to validate any future open loop simulations

that the team may need to run. This data was then paired with the test case to be saved in the test log repository and updating of the RMD and vehicle logbook, if necessary, indicated by the test case. Once the documentation was completed, another meeting was held by the testing team, engineering, and project manager to discuss the data that was gathered. The meeting provided insight into the successes and failures of the test case. If any failure happened, the team must determine why, find a solution for the issue, and update the test case to be completed in the future.

4.2 Model-in-the-Loop and Sensor Testing

In Year 2 of the EcoCAR Mobility Challenge, the team was provided the sensor suites and software for simulations and testing and began testing the sensors using Vector software on the bench and on a mule vehicle. The team started working on test cases for the Mobileye Camera and the Bosch Radar. The purpose of these test cases was not requirement based but rather to see the functionality of the sensor suite. Many tests such as approach tests and stationary objects tests were conducted to learn how accurate the sensors were outputting. The sensors were also tested for different safety faults to understand what mitigation plan the team might build to avoid those faults or to correct them.

Along with prior testing of the sensors, the ACC and sensor fusion algorithms were the main testing components in the MIL environment. The vehicle model used in the MIL environment is shown in Figure 9. The model was developed in a MATLAB/Simulink environment.

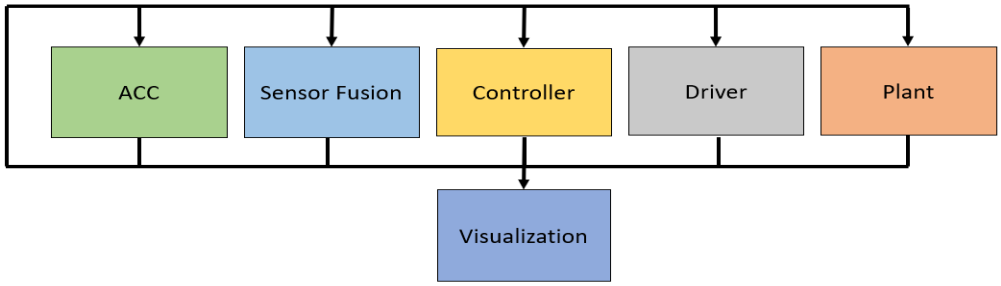


Figure 9: Vehicle MIL Model

The model contained the adaptive cruise control algorithm, the sensor fusion algorithm, a longitudinal driver model used to simulate driver behavior, and a plant model containing the vehicle longitudinal dynamics and propulsion components. The Controller subsystem contained the energy management strategy designed by the propulsion controls and modeling team that was used in vehicle to provide realistic vehicle performance. The Controller outputted torque commands into the Plant subsystem to allow the lower-level control actuators to produce component torque for the electric motor or engine in the vehicle. The Driver subsystem used a Simulink integrated longitudinal driver control to provide ACC operating modes. The functionality included a driver coming to a desired speed and setting cruise. The accelerator and brake pedal inputs used by the driver were to remain zero to mimic the human interaction while the system was engaged. [Year4]

4.3 Gathered Data from Sensors Using Vehicle Scenarios

In the middle of Year 2 and the beginning of Year 3 the sensors were added to a mule vehicle to provide realistic vehicle data for the algorithms to be tested in HIL. The team created test cases using the procedure to create a variety of situations the vehicle would experience on the road. An example of a test case would be the ego vehicle approaching a target vehicle that was at a dead stop several meters away. This test could be re-run at varying distances and speeds to capture highway and city traffic. The sensor suite needed to provide longitudinal and lateral distances of objects for the algorithm to calculate the torque commands of the vehicle. The sensor suite was able to provide distances, relative velocities and accelerations, and the object's classification whether it was a car, truck, or two-wheel vehicle. Figure 10 shows an example of an approach test of an ego vehicle moving towards a stopped target vehicle.

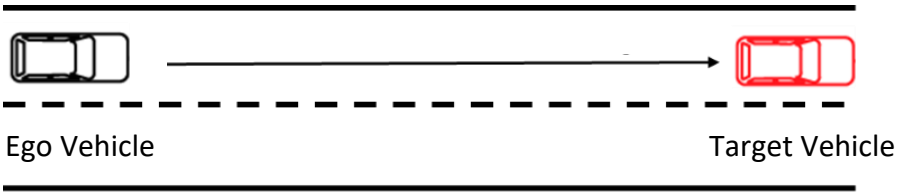


Figure 10: Vehicle Approach Test

4.4 Hardware-in-the-Loop Testing

Hardware-in-the-loop testing starts when both algorithms have passed all the requirements in the MIL environment. This testing started in Year 3 of the competition where each algorithm was coded in C++ and loaded onto the Intel IoT Tank. Communication of these algorithms as well as the sensor suite was crucial in this step to make sure all the time requirements were met for the CAN network. The hardware architecture tested in the HIL environment is shown in Figure 11.

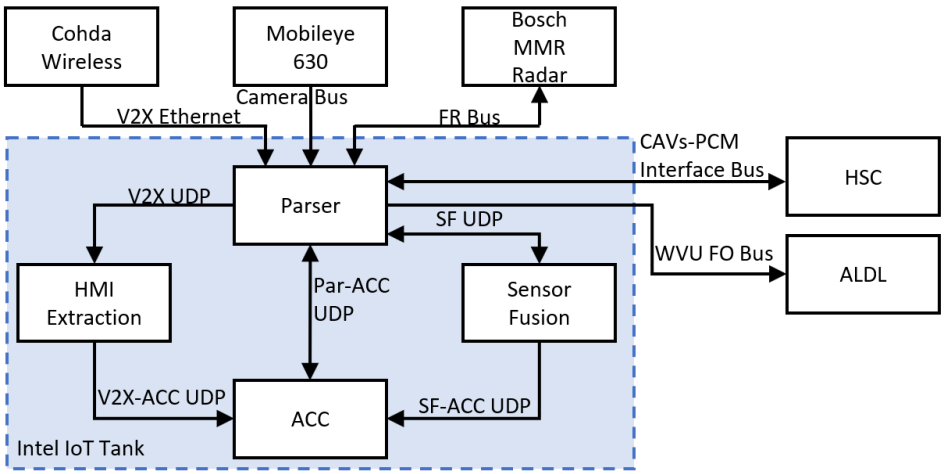


Figure 11: Hardware Architecture

Each algorithm communicated through a UDP connection while the sensor suite and hybrid supervisory controller (HSC) were connected with individual CAN networks. The Cohda Wireless radio was connected to the Intel Iot Tank via ethernet. The team used the free and open-source software Linux tools when testing in HIL to simulate real time scenarios. The command used to replay data through CAN networks

was “canplayer”. This tool allowed the team to reassign CAN interfaces to permit a replay of a log file created by gathering data from the sensor suite using different testing scenarios.

The FOSS used to test the timing of the algorithms and CAN messages was “canutils.” The command to monitor the CAN busses was “cansniffer.” This program is terminal based and allowed the team to monitor the messages going across the bus without the need to connect to the HSC if it wasn’t available for HIL testing. This program recorded message times as well as outputs to be saved for testing results of the test cases. The team also used Vector loggers to make sure the outputs were sent across the physical bus as well. The benefit of having these different HIL testing methods was useful for when hardware wasn’t available or the team needed to work remotely from the lab.

4.5 Vehicle-in-the-Loop Testing

Vehicle-in-the-loop testing began at the end of Year 3 and continued through Year 4 of the competition. The functional requirements set by the competition are outlined in Table 2 for the sensor fusion algorithm and in Table 3 for the ACC algorithm.

Table 2: Sensor Fusion Performance Metrics

Performance Metrics	Defined Metrics
Longitudinal Distance Error	<= 5%
Longitudinal Velocity Error	<= 0.9 m/s
Determine in Path	>= 90%
Minimal First Detection Distance	120 m
Multiple Object Tracking Accuracy	>= 90%

Table 3: ACC Performance Metrics

Performance Metrics	Units
Speed Error	< 2.5%
Following Headway Error	+- 0.5 seconds
Tracking Distance Error	< 5m
Stopping Distance Error	< 1.5m
Response Time	< 1.5 seconds
Max Acceleration	<= 0.3 g
Max Deceleration	<= 0.5 g

Starting with the performance metrics for the sensor fusion algorithm the error for the longitudinal distance from the ego vehicle and target vehicle needed to be less than 5% error. The target vehicle is the vehicle in path of the ego vehicle when testing in VIL. The longitudinal velocity error needed to be less than or equal to 0.9 mps. The first detection of an object that is in the path of the ego vehicle needed to be detected beyond or at 120 m. Once the target vehicle is detected it must remain detected as long as it is in path of the ego vehicle for 90% of the time. This is to show that the algorithm can maintain tracking information of the target vehicle as well as track multiple objects around the ego vehicle. This metric was a team goal set at 90% to make sure that multiple objects were not fusing together and creating unnecessary torque commands in the ACC algorithm.

The ACC performance metrics combine safety related requirements as well as ride comfortability for the driver of the vehicle. The ACC algorithm should set a speed and maintain that set speed with less than 2.5% error and the vehicle should not accelerate more than 0.3 g or decelerate at 0.5 g. The response time was tested in HIL to make sure that the ACC algorithm would respond to an event less than 1.5

seconds which was demonstrated in vehicle with a target vehicle cutting in front of the ego vehicle which can happen on highways. The ACC algorithm has three distances the vehicle can ride behind a target vehicle. This is measured in seconds until collision with the target vehicle. The distances are short, medium, and long. Each of these times was reported to the EcoCAR competition to validate the following headway error that needed to be within 0.5 seconds of that time. The target vehicle reported by the ACC needed to be within 5 m of the ground truth data when both vehicles are in motion. When the target vehicle comes to a stop the ego vehicle should come to a stop behind the target vehicle within 1.5 m.

The last process tested in the VIL environment was the startup procedure of the sensor fusion and ACC algorithms. The added components to the vehicle should act as a stock system would in the Chevy Blazer. The startup scripts should boot the programs up when the Intel IoT Tank received power and should command the vehicle when the ACC buttons were pressed on the steering wheel of the vehicle. Due to competition safety guidelines both algorithms must be tested on a closed course environment and not be driven on the public road.

5.0 Results

The results reported in this next section include test procedures, environments, and competition results. Section 5.1 explains the effectiveness of the test procedure throughout the years of the competition as well as the effectiveness of the procedure used in any autonomous vehicle environment testing. The results from the test procedure in each environment are explained in section 5.2. Two examples, the approach and following test, are explained throughout the results sections. The first example is an approach test of the ego vehicle moving towards the target vehicle at 35 mph and coming to a stop behind the target at a set distance The second example is a following test of the ego vehicle moving behind the target vehicle while maintaining 35 mph or by keeping the set gap setting chosen by the driver. A sensor fusion and ACC algorithm are tested with the same parameters and sensor information to provide an accurate end-to-end simulation of the VIL environment. Section 5.3 discusses how well the team placed in the EcoCAR Mobility Challenge competition as well as how the teams' algorithms performed using the metrics discussed in section 4.5.

5.1 Test Procedure Results

A vehicle-in-the-loop example performed by the WVU EcoCAR is shown in the Appendix. The test procedure describes the ego vehicle approaching a parked SUV at 35 mph and coming to a stop behind the vehicle. The test case describes the setup, test, and teardown procedure using a step-by-step process. The document also includes pass/fail criteria, hard and soft pass actions, and what is considered a failure of the test. The test case is meant to be a generalized version where the test can be re-run with different parameters being recorded to the right of the test case procedure. A few examples of parameters include different speeds and vehicles which could affect the outcome of the test procedure. It is important to note all issues with the test for future improvements in the MIL and HIL environment. The appropriate CAN bus was recorded to be able to re-test in the HIL environment which helped in validating the sensor

fusion and ACC model. In this test case the ACC and sensor fusion algorithm performed a hard pass as the algorithms met all the performance metrics. Throughout the years of the EcoCAR competition, these test cases were continuously saved for future testing use as they saved time, effort, and planning. As students graduated throughout the years, the documentation could be passed down and used for onboard training for new members of the team. The team recognized this as an effective way to familiarize each testing team member as they might not be acquainted with the system that they were testing. This test case approach can be used for testing engineers in the automotive industry as a stepping stone to organizing test cases. The testing engineers had a way of communicating with the development teams throughout the development cycle of the software. As mentioned in Section 3.2.2, a test case and vehicle logbook are also managed by the testing team provide insight to the engineering manager and development team to correct any issues with testing or to contact the tester to schedule a meeting concerning the issues they were experiencing. A second example in the Appendix shows the use of the test case creation for a following test of a target vehicle. The process was carried out the same way as any other test but provided more information of how the vehicle will perform in city environment.

5.2 MIL, HIL, and VIL Results

The results below were gathered from the last year of the competition before heading to the final competition. The sensor fusion team was able to show each level of the environment while performing the same in HIL and in VIL. The results showed that these test procedures set up for the sensor fusion team could be repeated at either level if the team had real world collected data. The team duplicated several test scenarios that the vehicle would encounter on the road to make this possible. The ACC team provided results from the MIL and VIL environment while the HIL environment was mainly for connection setup and verifying connectivity to the sensor fusion algorithm as well as connecting to the hybrid supervisory controller.

5.2.1 MIL Results Sensor Fusion

The team used simulated values in the MIL model while showing the target distance with as little noise possible to test the algorithm's performance. Figure 12 and Figure 13 are the results from the MIL model. The sensor fusion team tracked a target vehicle greater than 120 meters away while maintaining detection in path greater than 90% of the time it saw the target vehicle. The algorithm kept track of the vehicle until the end of the test when the target vehicle came to a stop around 10 meters from the vehicle in each test. Figure 12 shows an approach test starting 250 m away and approaching the target vehicle at 35 mph. Figure 13 shows a following test of the target vehicle varying speed while the ego vehicle maintained a set speed of 35 mph to demonstrate the tracking algorithm within sensor fusion.

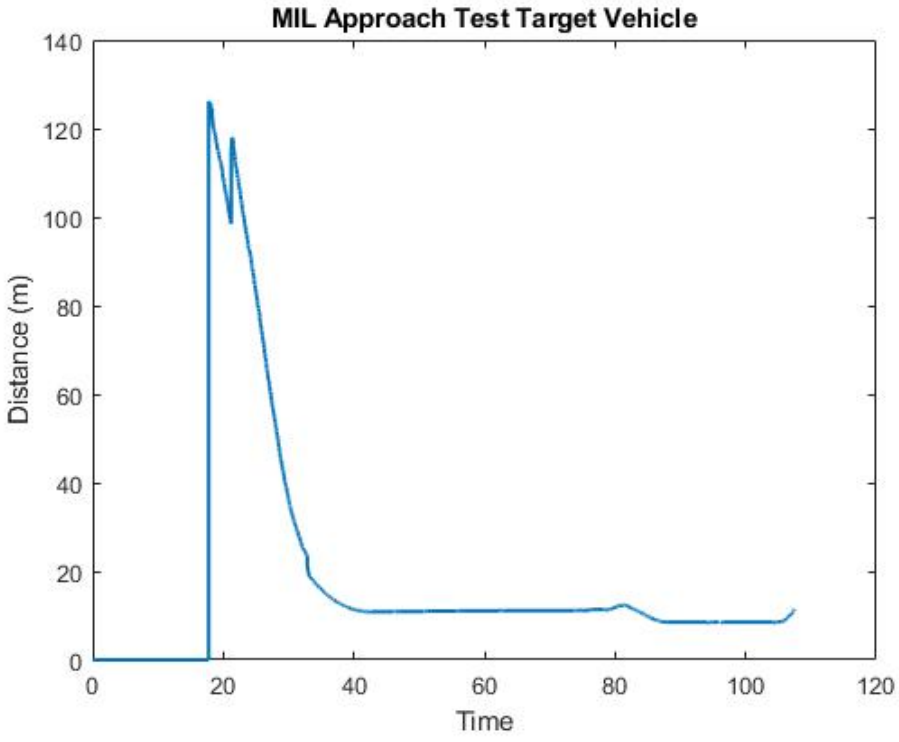


Figure 12: Sensor Fusion MIL Approach Test

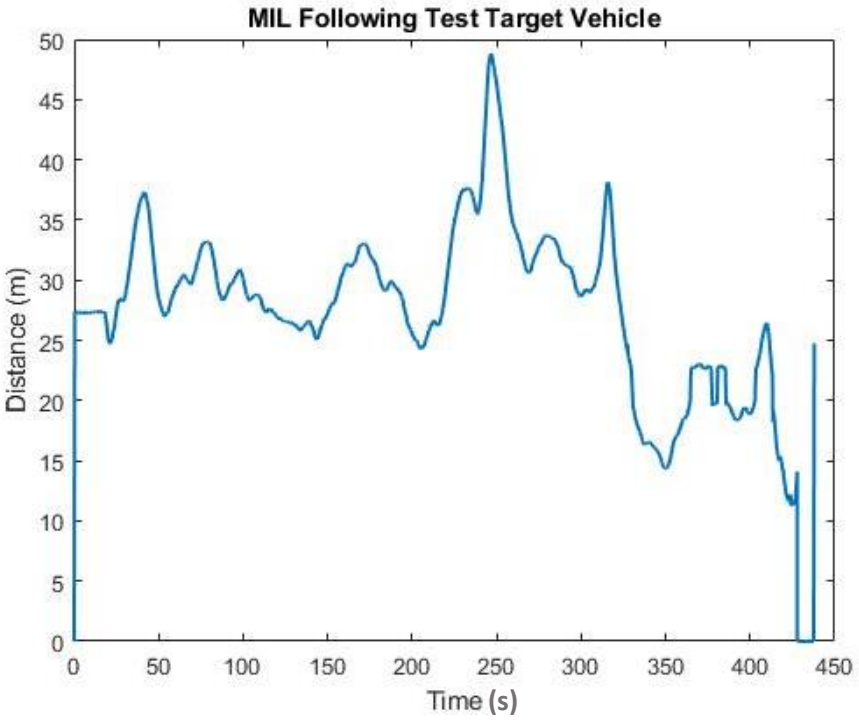


Figure 13: Sensor Fusion MIL Following Test

5.2.2 MIL Approach Result Adaptive Cruise Control

The first four figures, Figure 14, Figure 15, Figure 16, and Figure 17, are the results of the MIL approach test performed by the ACC team. The vehicle started at 0 mph and accelerated to 35 mph then maintained that speed until the target vehicle was in range of the ACC distance controller which started to slow down the vehicle around 60 seconds into the test. The set speed tried to maintain the ego vehicle at 35 mph while the distance controller wanted to keep the gap distance to prevent a collision with the target vehicle. Each graph shows a decrease in which the vehicle reacted to the target vehicle in front of the ego vehicle. The ACC wheel torque command went negative which implied a braking maneuver was happening to the ego vehicle. From 60 seconds and beyond, the ego vehicle began to slow down while approaching the target vehicle until it came to a complete stop around 90 seconds into the test. This distance error graph shows that the vehicle was able to perform an approach test at 35 mph while staying at least 14 meters from the target vehicle before a collision happened.

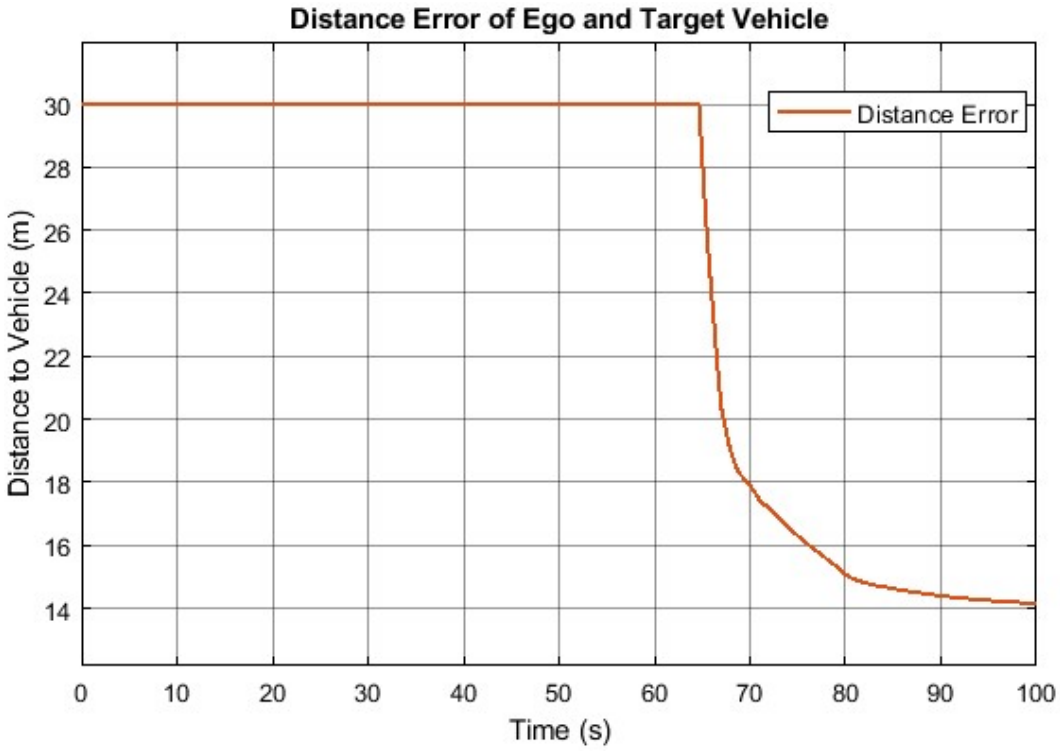


Figure 14: ACC MIL Approach Test Distance Error

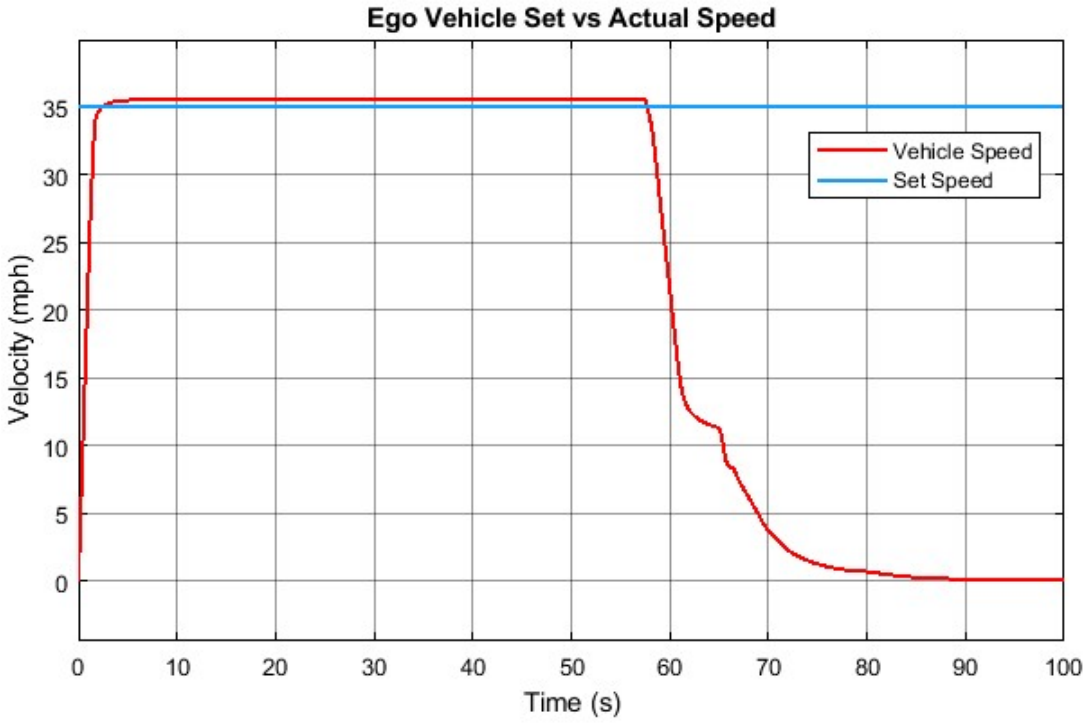


Figure 15: ACC MIL Approach Test Set vs Actual Speed

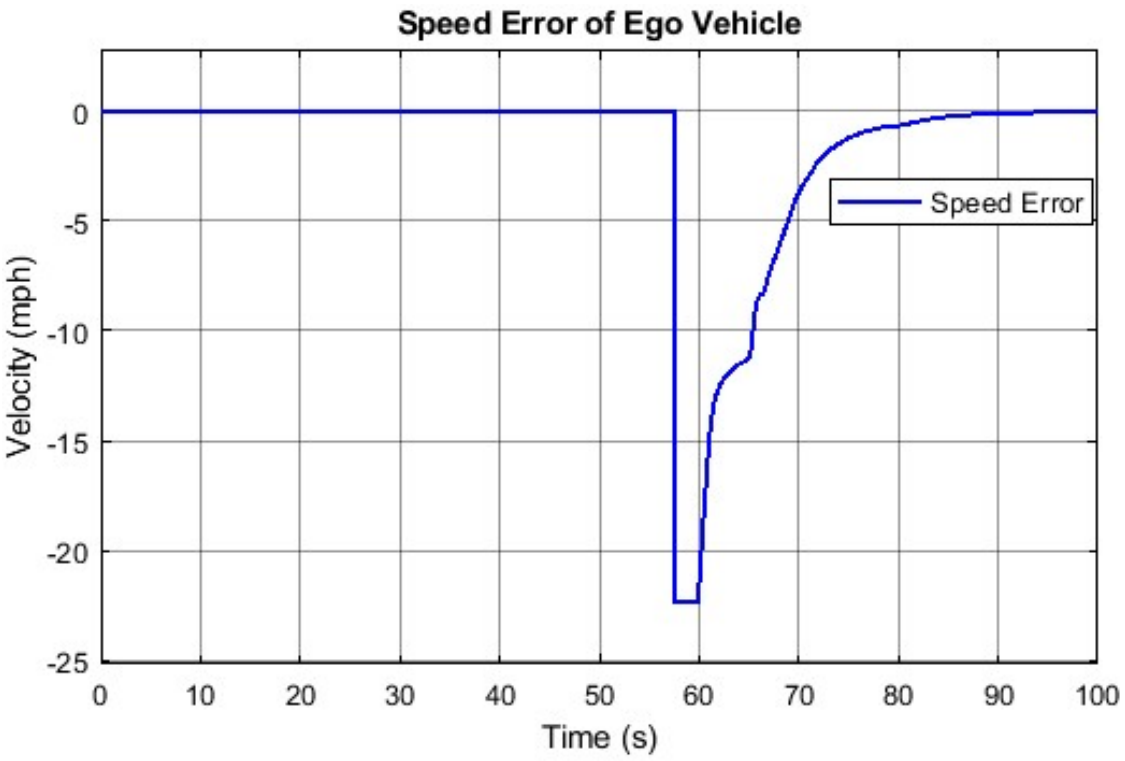


Figure 16: ACC MIL Approach Test Speed Error

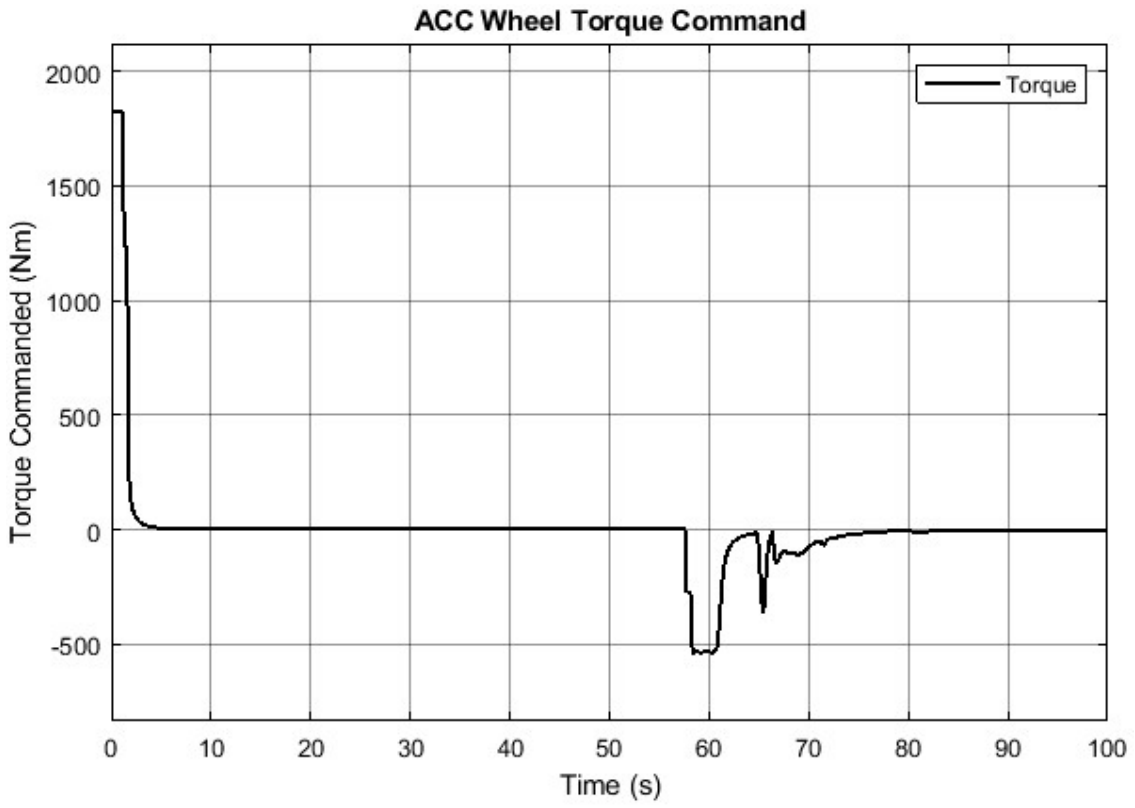


Figure 17: ACC MIL Approach Test Commanded Torque

5.2.3 MIL Following Result Adaptive Cruise Control

The next six Figure 18 through Figure 22 are data from a following test in the MIL environment performed by the ACC team. The ego vehicle started behind the target vehicle on the test which explained why there was a sudden drop in distance error, but Figure 21 shows the target vehicle starting out at a speed of 5 mph. The target vehicle accelerated to 35 mph in 10 seconds and maintained that speed for 40 seconds. The target vehicle finally slowed down to 15 mph in 25 seconds. Once the target vehicle passed the gap setting distance for the ego vehicle, it began to accelerate behind the target vehicle until it reached the set speed of 35 mph. The ego vehicle maintained a safe distance behind the target vehicle. The ACC algorithm then detected a deceleration from the target vehicle and proceeded to command a negative wheel torque at 50 seconds in Figure 17. As the target vehicle was approached a stop, the ego vehicle slowed down and stopped 3 meters away as shown in Figure 18.

Figure 18: ACC MIL Following Test Distance Error

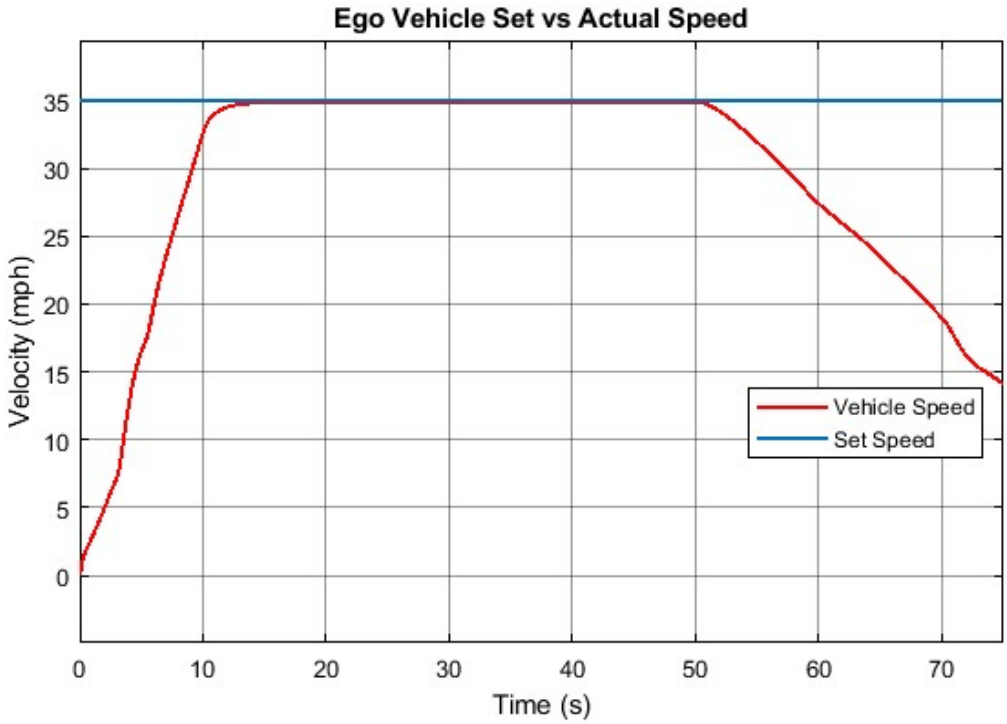


Figure 19: ACC MIL Following Test Set vs Actual Speed

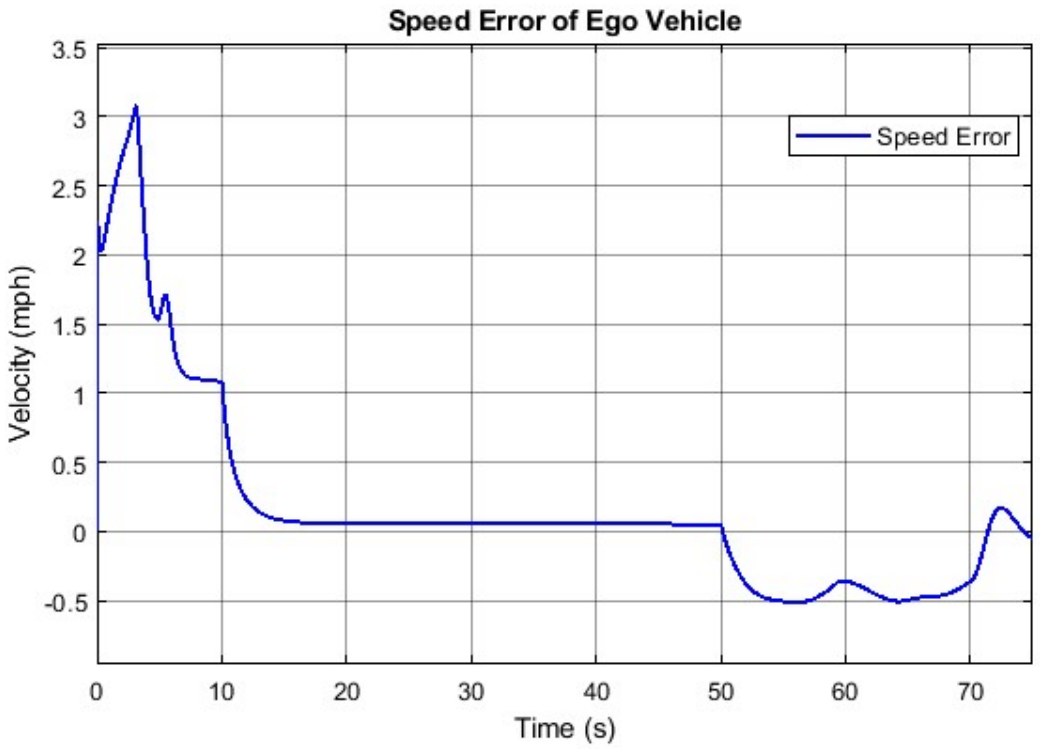


Figure 20: ACC MIL Following Test Speed Error

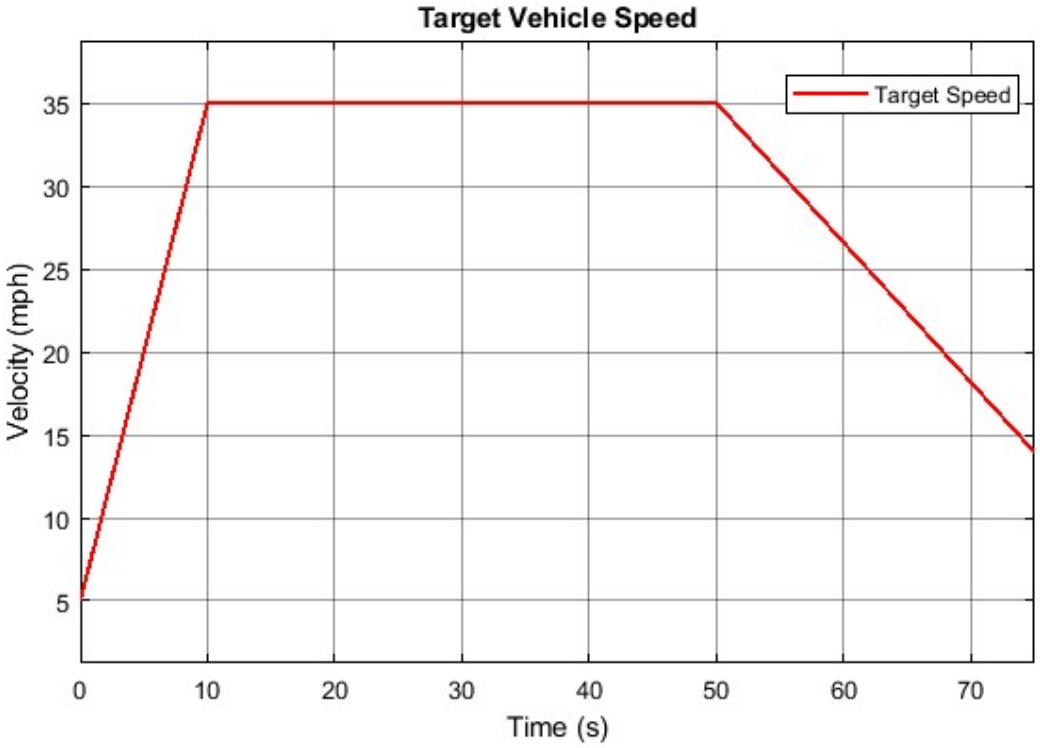


Figure 21: ACC MIL Following Test Target Vehicle Speed

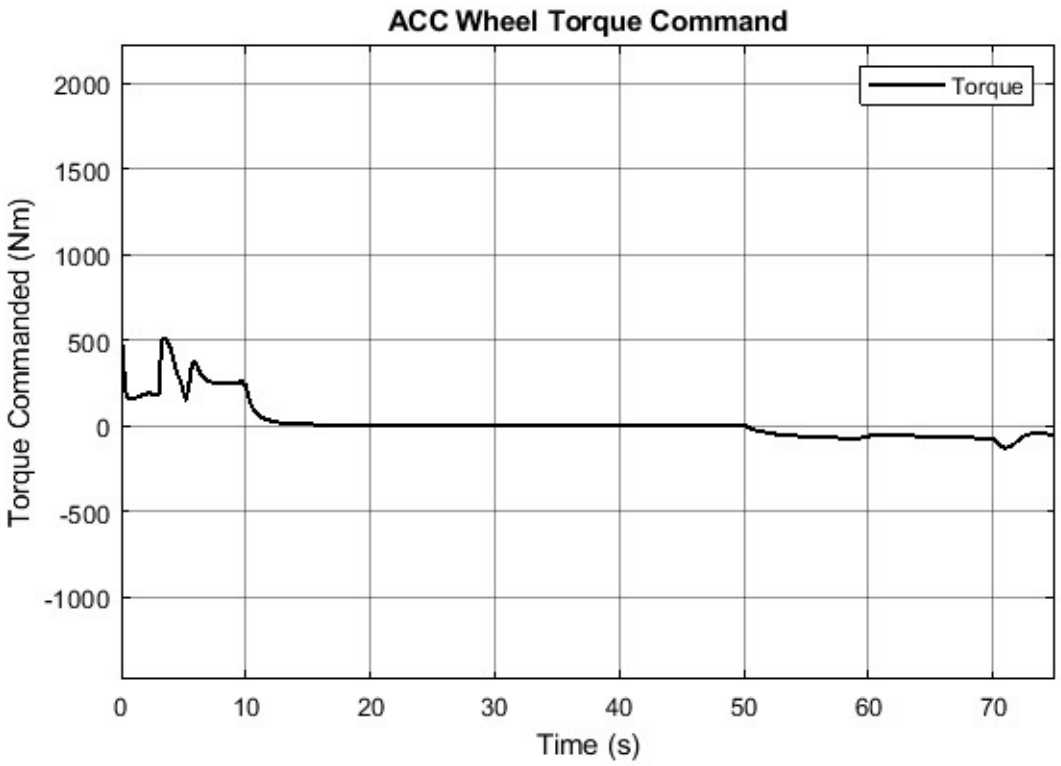


Figure 22: ACC MIL Following Test Commanded Torque

5.2.4 HIL and VIL Sensor Fusion Test

Figure 23 through Figure 26 demonstrate the similarity of the two environments developed over the last two years. Real world data was captured from the sensor fusion suite that was attached to a mule vehicle. The data was then used to model the sensor suite in the HIL environment. The approach test was completed exactly as the MIL test was performed to validate the team's sensor fusion algorithm in which the team saw similarities in all three environments. The following test was performed with a target vehicle reaching a set speed of 35 mph and maintained that set speed to the end of the test for safety consideration of varying speed at the time of development. Both tests were repeated in VIL on the ego vehicle to verify the two environments and their similarities. Figure 24Figure 26 in the following test vary in distance due to the accuracy of the speedometer and driver of the tests completed at the time. These two examples show how comparable the two environments have become for the team and can rely on

HIL testing before going out and VIL testing the Chevy Blazer. Since it took numerous hours to prepare each VIL test, HIL testing has become a staple in the team's workload while it freed up hardware and vehicle testing time.

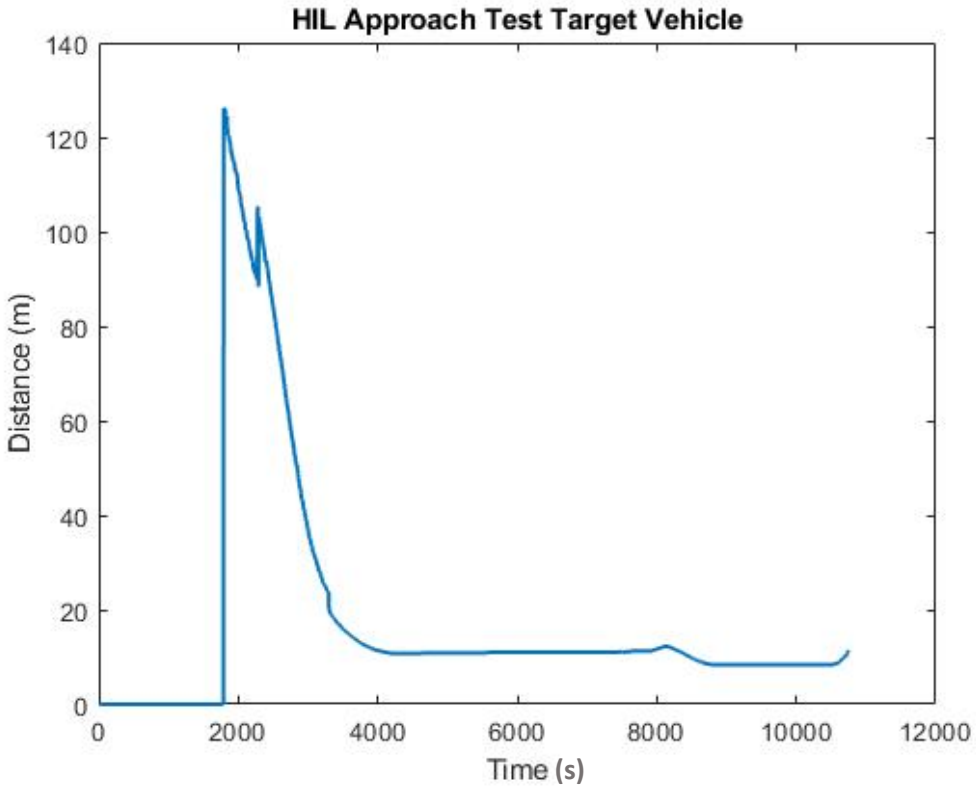


Figure 23: Sensor Fusion HIL Approach Test

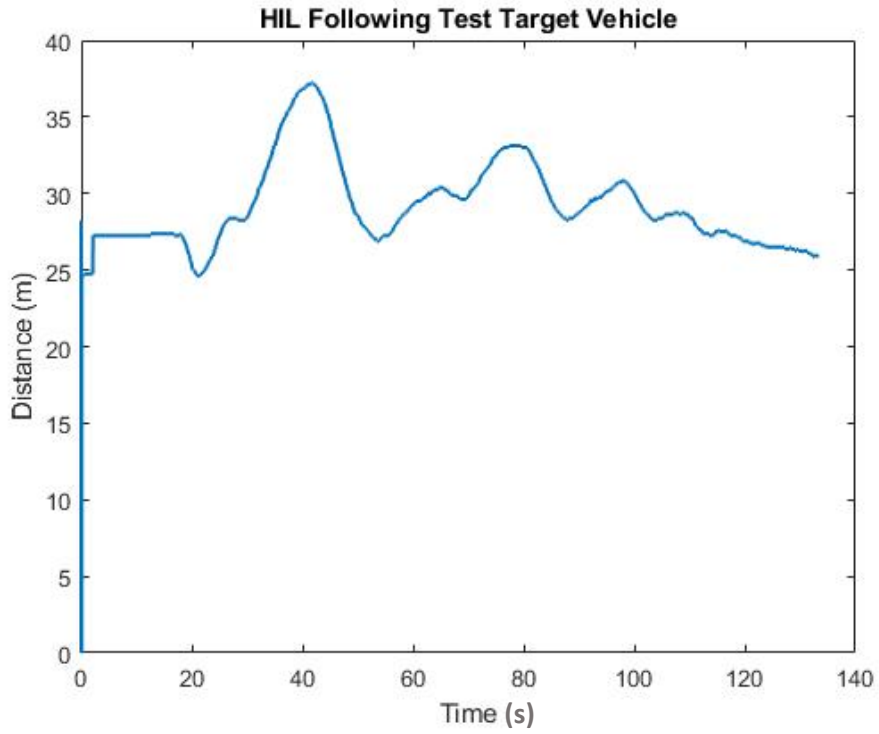


Figure 24: Sensor Fusion HIL Following Test

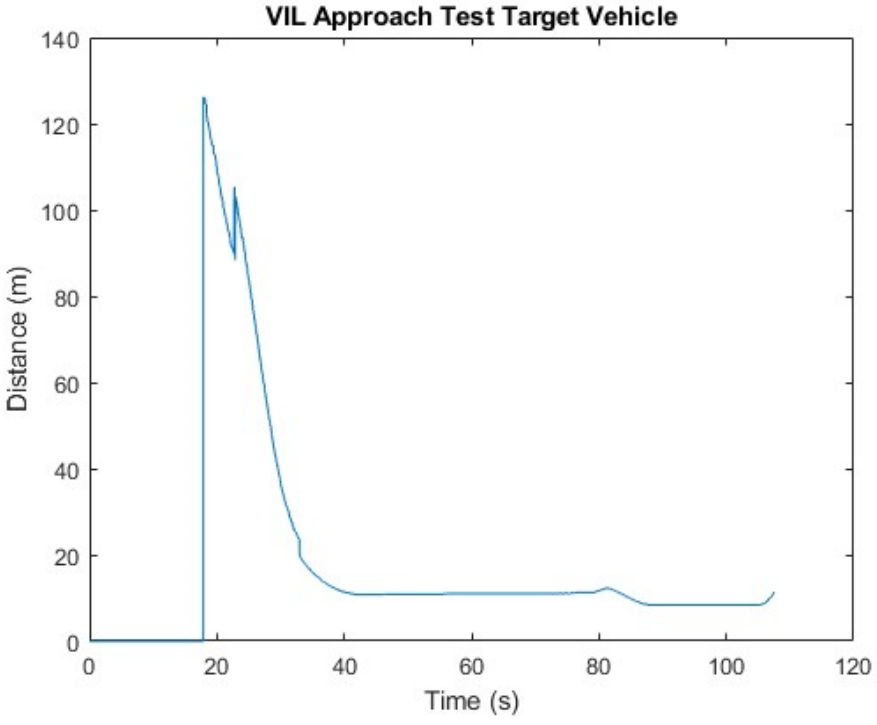


Figure 25: Sensor Fusion VIL Approach Test

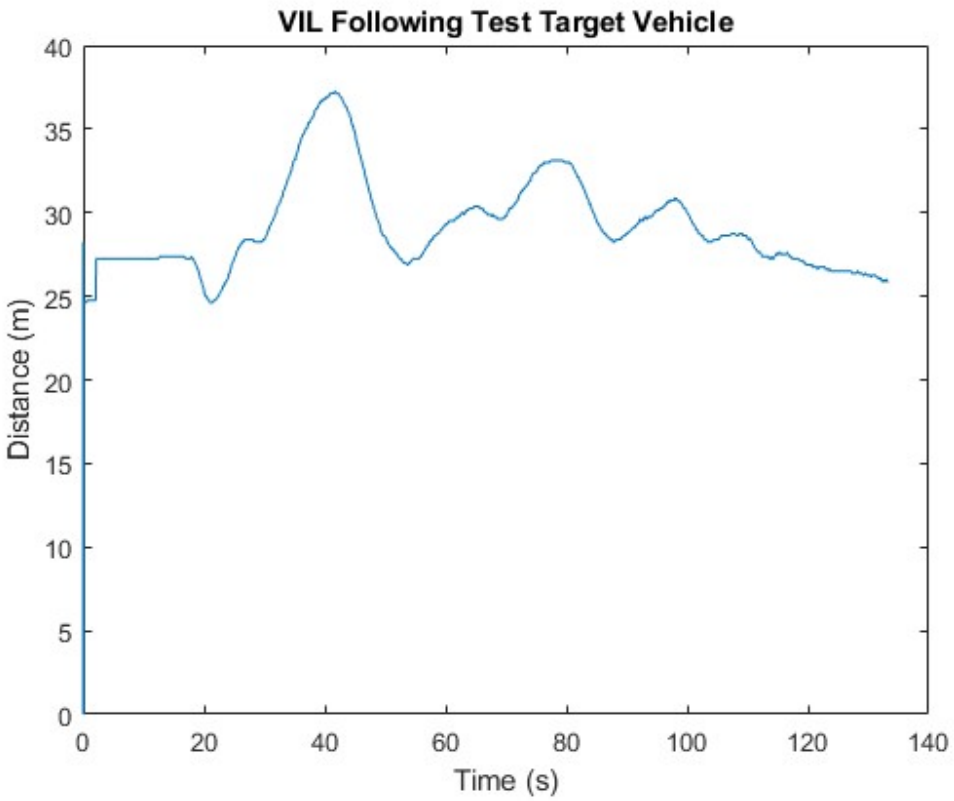


Figure 26: Sensor Fusion VIL Following Test

5.2.5 VIL Approach Test Result Adaptive Cruise Control

The VIL approach test for the ACC subteam was performed exactly as the MIL test was written. Figure 27 shows that the target vehicle was over 100 meters away before the ACC algorithm detected the target vehicle around 2175 ms. In Figure 28 the driver maintained the ego vehicle's speed at 35 mph and engaged cruise control around 1500 ms with a set speed of 28 mph. The ego vehicle decreased its speed to 18 mph and maintained that speed at 2100 ms until the target vehicle was in range of the ego vehicle. The ACC

algorithm demanded a negative wheel torque to slow down the vehicle at 2200 ms in Figure 29. This led to the vehicle creeping to a slow stop before the end of the approach test.

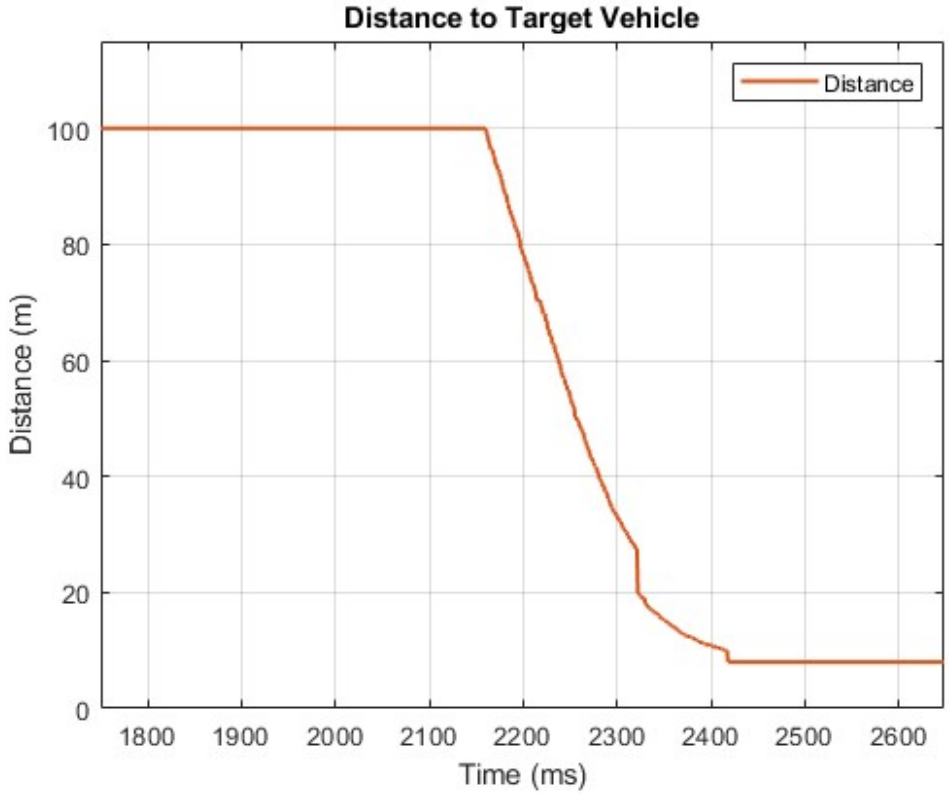


Figure 27: ACC VIL Approach Test Distance Error

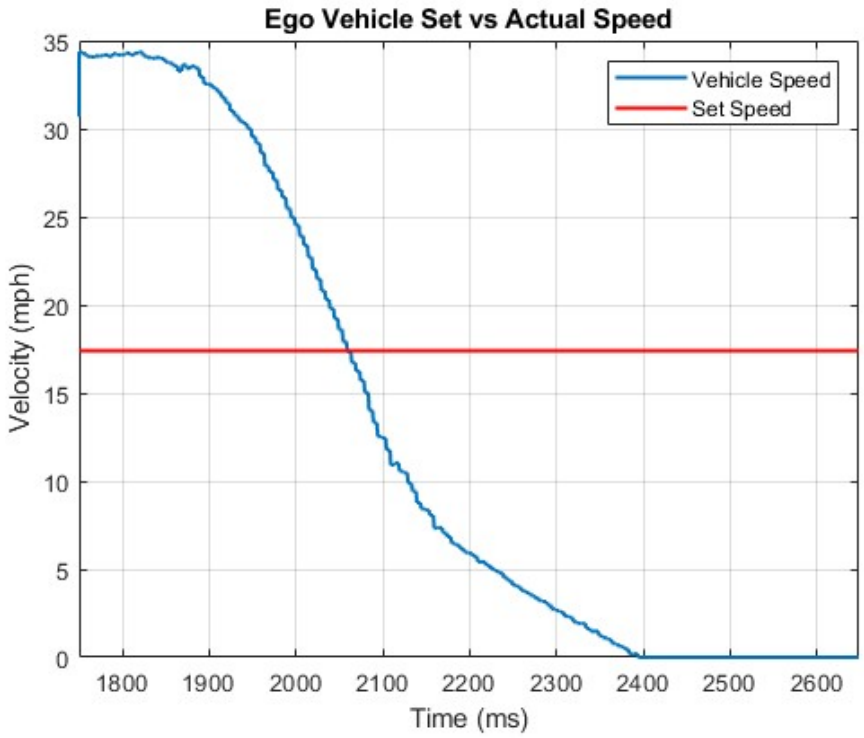


Figure 28: ACC VIL Approach Test Set vs Actual Speed

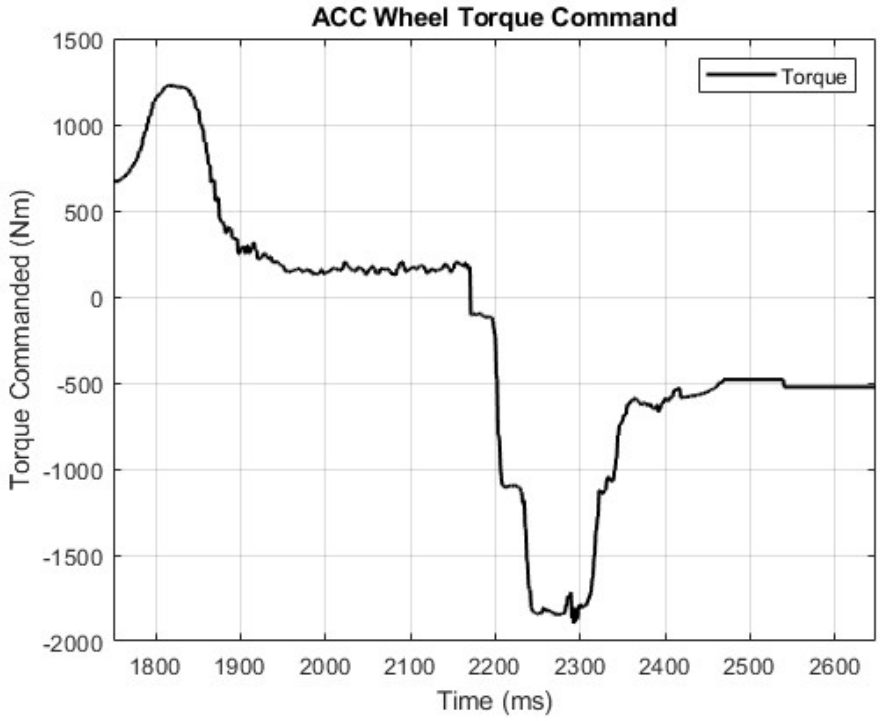


Figure 29: ACC VIL Approach Test Commanded Wheel Torque

5.2.6 VIL Following Test Result Adaptive Cruise Control

The VIL following test results are displayed in figures Figure 30 through Figure 32. In Figure 31 the ego and target vehicle started at 35 mph about 10 meters from each other. The target vehicle accelerated away which increased the gap distance of the ego vehicle and target vehicle. This resulted in the gap distance increase which accelerated the ego vehicle's speed and lead to a positive torque command as shown in Figure 32. The vehicle maintained its current speed since the set speed was preset at a highway speed. At the time the controller did not need to increase the speed of the ego vehicle as the target vehicle began to slow down increasing the distance error of the two vehicles. At 2400 ms the ego vehicle started to reduce speed as the target grew closer. This led to a hard deceleration event around 2650 ms and coming to a stop around 10 meters from the vehicle. The ego vehicle maintained a safe distance behind the target vehicle without unnecessary acceleration and deceleration until the target vehicle came to a stop. These tests were particularly harder to run due to the driver's experience at maintaining certain speeds as well as the closed course testing environment. Long straight-a-ways and ideal weather were helpful in perfecting these tests.

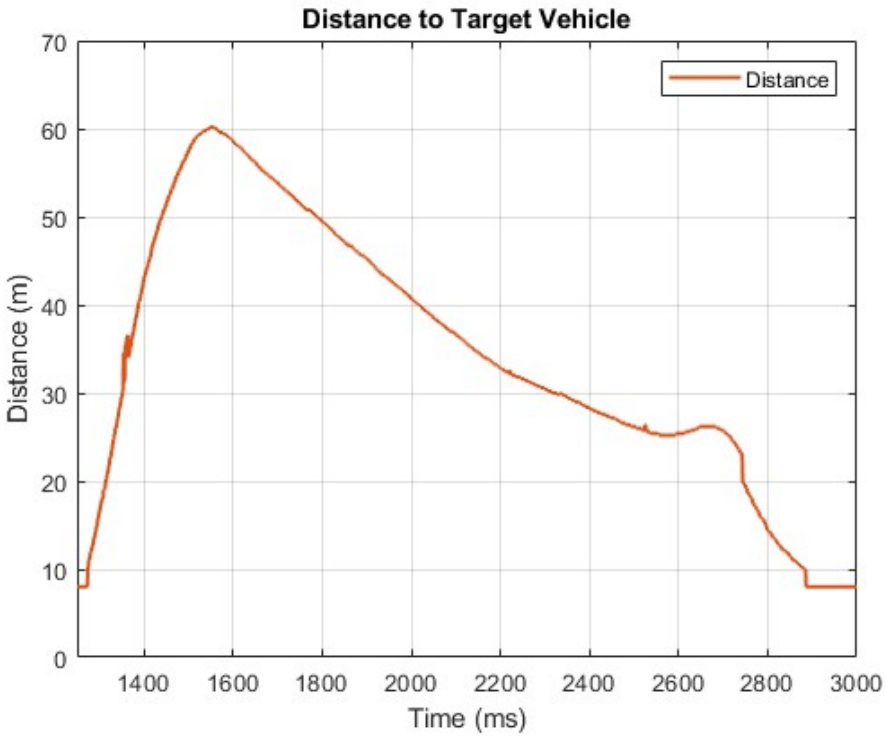


Figure 30: ACC VIL Following Test Distance Error

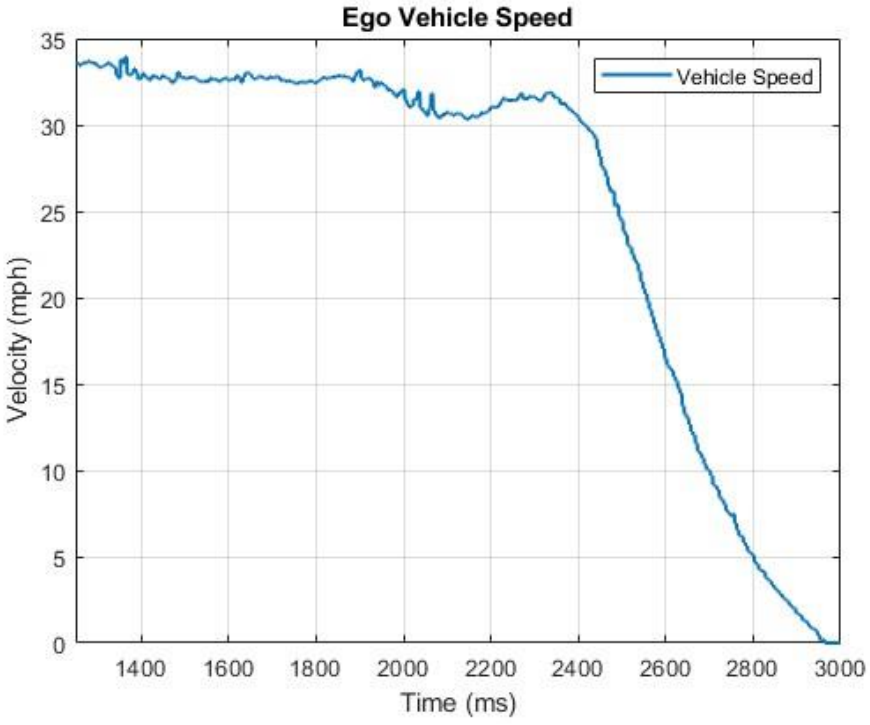


Figure 31: ACC VIL Following Test Vehicle Speed

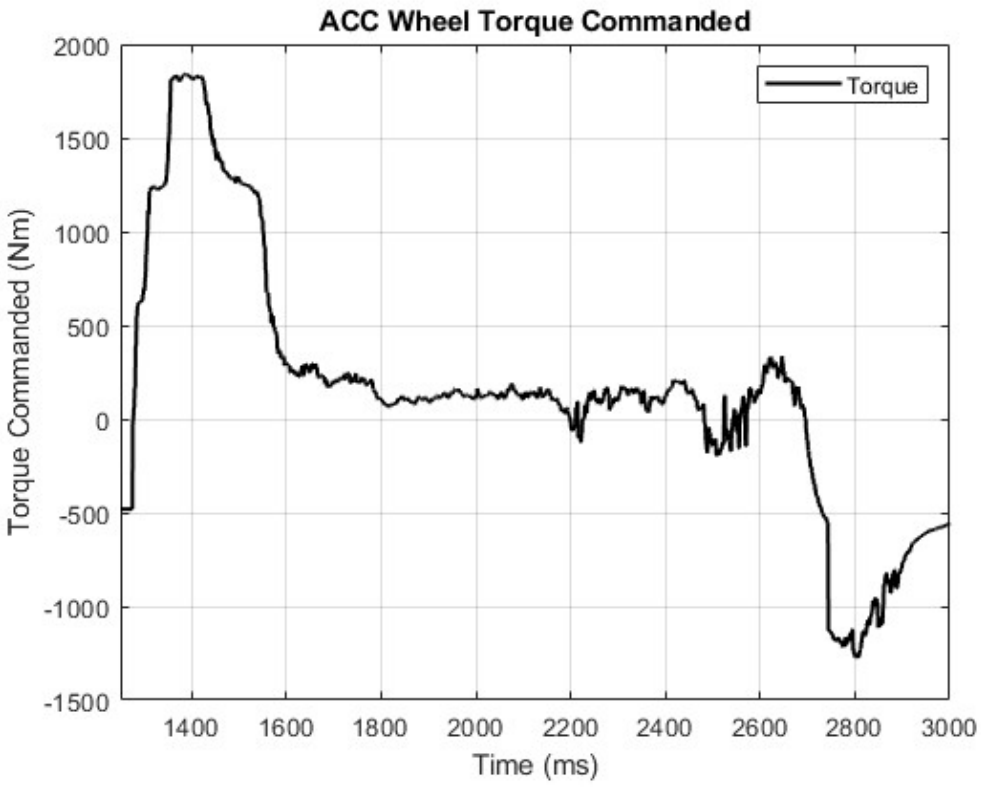


Figure 32: ACC VIL Following Test Torque Commanded

5.3 60-Mile Drive Cycle Results

Figure 33 provides an overview of the Jefferson circuit at Summit Point Motorsports Park in West Virginia where the ACC Endurance drive was conducted. This is a 60-mile drive, testing the Sensor Fusion and ACC System that has been developed using the development tools and processes described in this thesis. As shown in the figure, there are 4 points where the team had the potential to lose the lead vehicle while

navigating turns on the course. A drive cycle was designed to minimize the number of times the sensor fusion algorithm would lose the lead vehicle by slowing down when entering turns on the course.

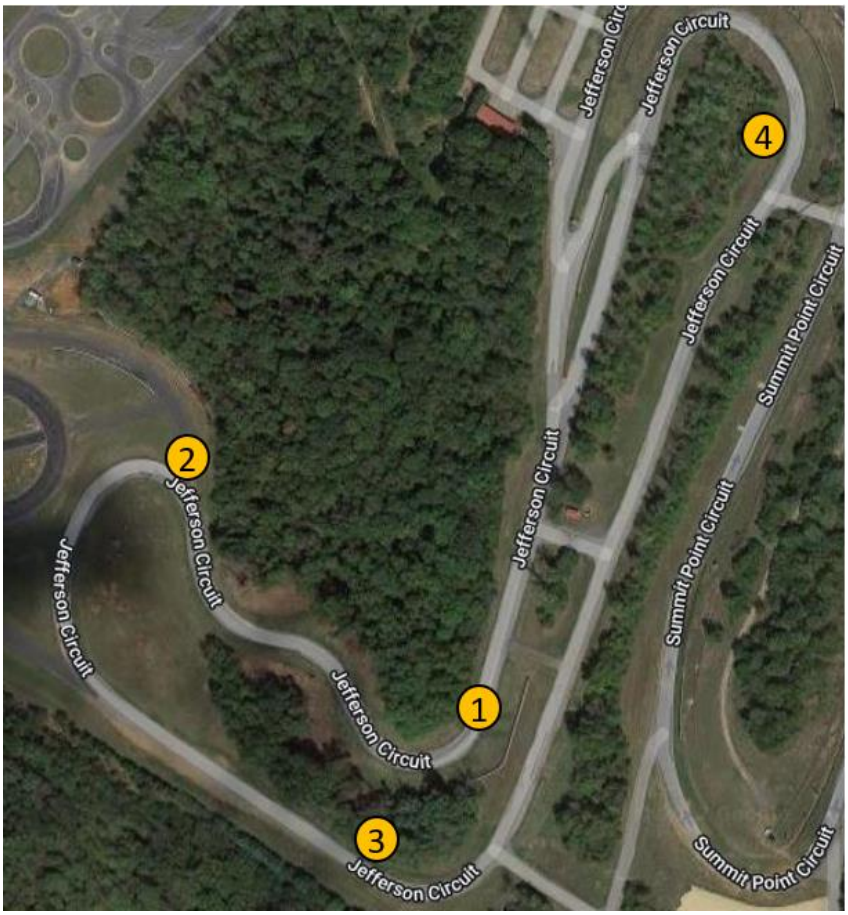


Figure 33: Summit Point Jefferson Circuit

Figure 34 illustrates the CAVs ACC Endurance drive performed on the Jefferson circuit for roughly 2.5 hours. The team achieved a maximum speed of 54 kph with an average of 41 kph.

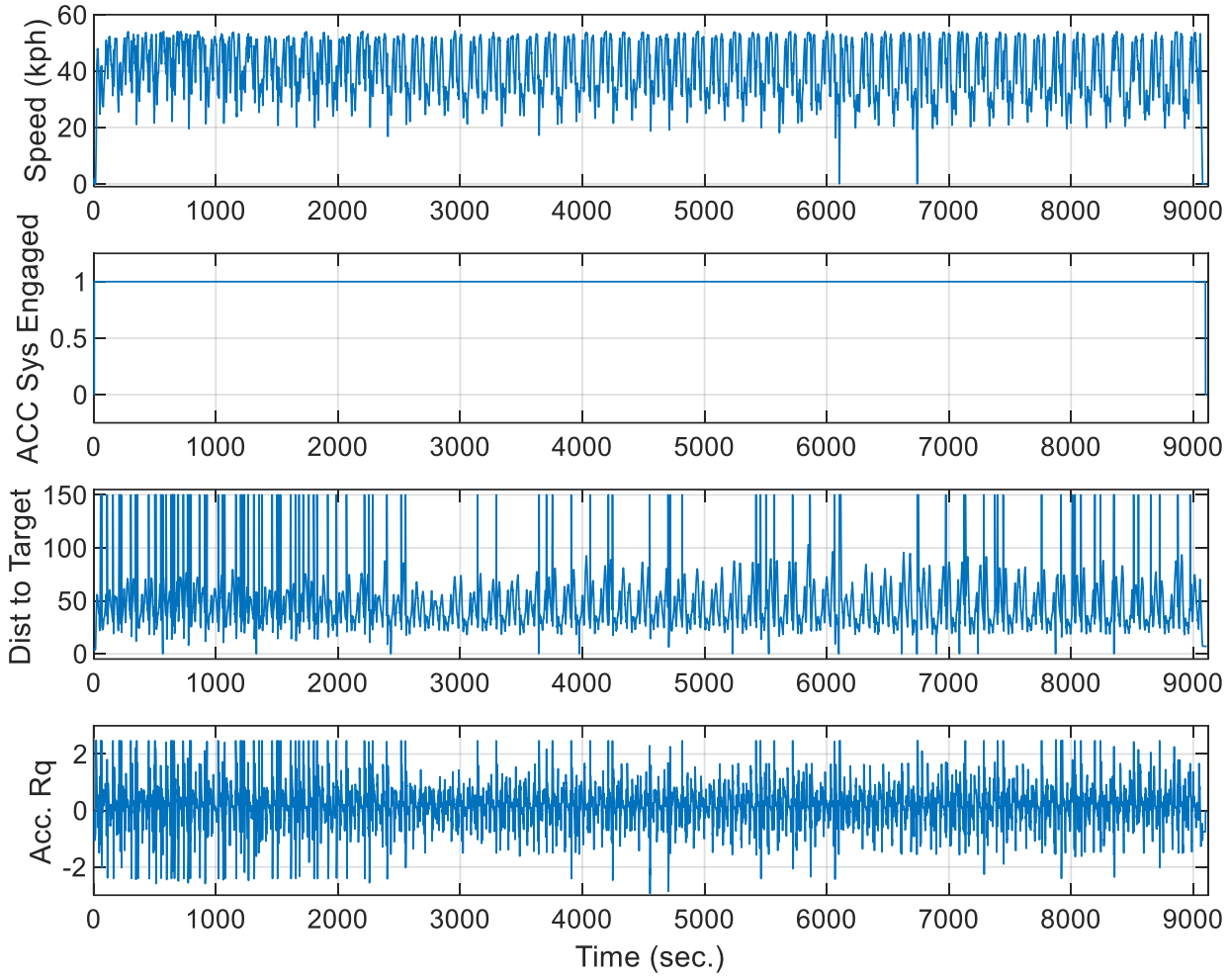


Figure 34: CAVs ACC Endurance Drive

Figure 35 provides a close-up of the first 1000 seconds of the drive cycle. The lead vehicle began roughly 4.4 meters away from the Blazer. The ACC algorithm was resumed at this point to prime the system for the start of the cycle. In subplot 3, the step changes to 150 meters in the distance to the target signal represent instances where the sensor fusion algorithm lost track of the lead vehicle around bends.

Regardless of this loss, the CAVs system was able to properly identify the lead vehicle when it came back into view and did not have to disengage ACC.

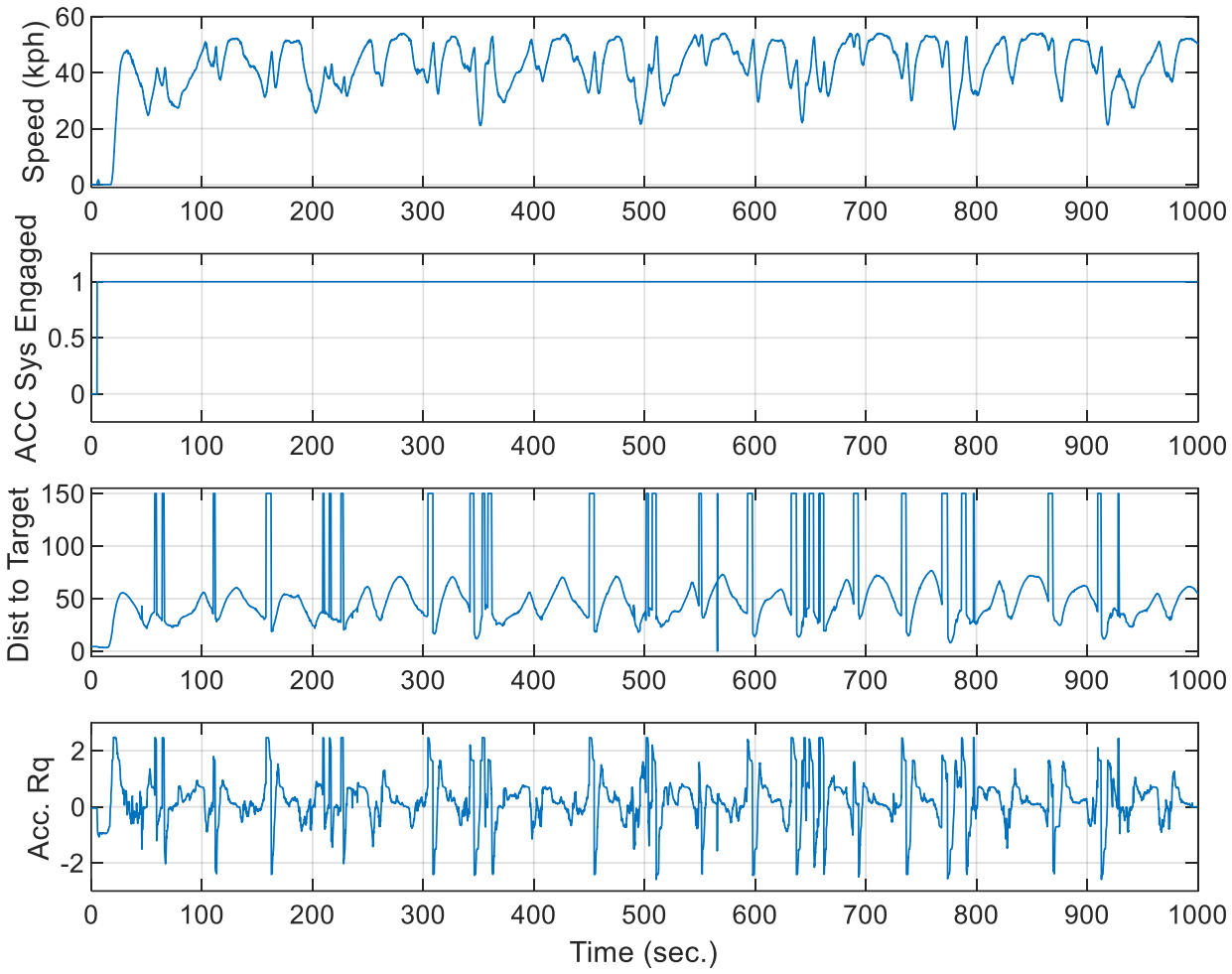


Figure 35: First 1000 seconds of Drive Cycle

Figure 36 provides the last 1000 seconds of the drive cycle. Similar instances occur when the sensor fusion algorithm loses track of the lead vehicle, but the driver did not have to interfere and disable ACC. At the end of the test, the lead vehicle comes to a stop and the ACC algorithm decelerate the Blazer until coming to a stop behind the lead vehicle. The system holds the vehicle in place until 9100 seconds when the driver steps on the brake pedal to officially end the endurance drive. At this instance, the ACC System Engaged signal (subplot 2) drops to zero.

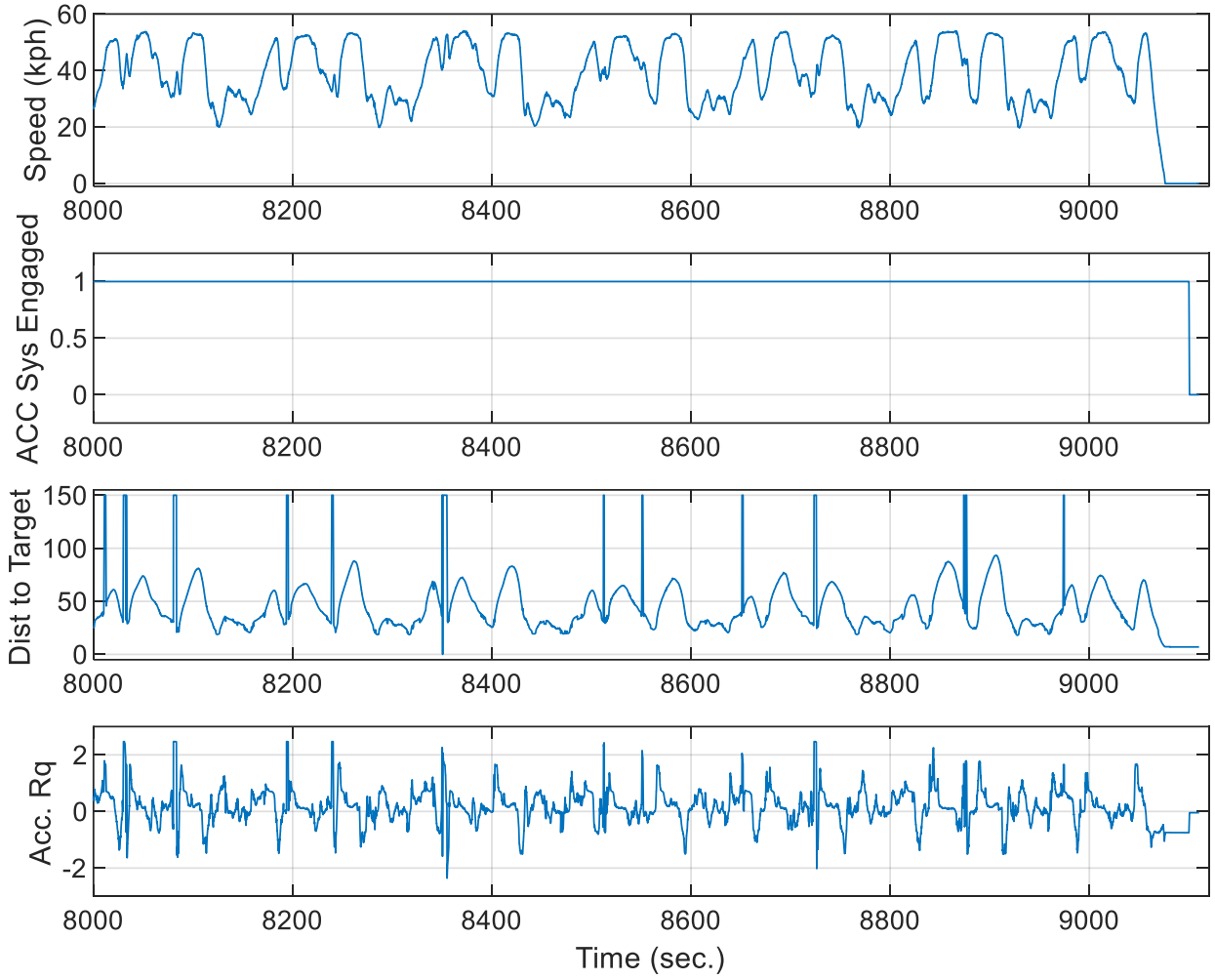


Figure 36: Last 1000 seconds of Drive Cycle

5.4 Competition Results

Each year of the competition the EcoCAR organizers held an award banquet for all the participating teams. There were awards for the overall placing teams as well as individual subteams awards and their accomplishments throughout the year. Years 3 and 4 concentrated more on the vehicle performance while driving the performance of adaptive cruise control. In Year 3 of the competition, WVU ended with an overall 3rd place finish while also finishing in 4th place for CAVs Perception Evaluation. The last year of the competition WVU finished 6th place overall while finishing 1st place in CAVs perception evaluation and

ACC drive quality. The team exceeded the performance metrics detailed in Section 4.5. Table 4 details that the sensor fusion team met their Year 4 metrics in every category while improving vastly in the last year in longitudinal distance error. The adaptive cruise control team completed their performance metrics as well as shown in Table 5. The was able to test and prototype as fast and regularly as possible which made an improvement on each system result in a complete solution by the end of the competition.

Table 4: Sensor Fusion Achieved Performance Metrics

Performance Metrics	Defined Metrics	Year 3	Year 4
Longitudinal Distance Error	<= 5%	24%	4%
Longitudinal Velocity Error	<= 0.9 m/s	1.7 m/s	0.7 m/s
Determine in Path	>= 90%	100 %	100 %
Minimal First Detection Distance	120 m	145 m	145 m
MOTA	>= 90%	N/A	98%

Table 5: ACC Achieved Performance Metrics

Performance Metric	Units
Speed Error	< 2.5%
Following Headway Error	+- 0.5 seconds
Tracking Distance Error	< 5m
Stopping Distance Error	< 1.5m
Response Time	< 1.5 seconds
Max Acceleration	<= 0.3 g
Max Deceleration	<= 0.5 g

6.0 Conclusion and Recommendations

In conclusion, this research showed the effectiveness of the methodology and procedures performed by the WVU EcoCAR team. The team consisted of college students wanting to gain real world experience in the automotive industry. The contribution of this thesis is the development of a data acquisition and testing methodology to support comprehensive testing and requirements verification using both model-generated and real-world data for this autonomous vehicle development environment. Previous academic work has largely focused on computer generated test data. More comprehensive and passenger-vehicle specific test methodologies have been used by industry in their vehicle development processes, but are not widely documented in the open literature.

The test case creation and tracking guided the advancement in making a comparable HIL and VIL environment to reduce the time it takes to prototype and test connected automated vehicle algorithms. As artificial intelligence continues to mature, AI will play an increasingly important role in generating and selecting test scenarios for each testing environment [15]. This research, as well as the competition, provided an insight into the workload required in a verification and validation role. The research enabled the team to compete at a high level in the competition and develop a consumer ready ACC system. This methodology and procedure are vital in that can be adapted in creating future successful products.

6.1 MIL, HIL, and VIL Improvements

The current MIL model developed by WVU EcoCAR is separated by the sensor suite, sensor fusion, and ACC algorithm. These areas can be expanded upon in various ways. The sensor suite could include different types of sensors as well as making Simulink blocks that can have multiple adjustments to the sensor suite model. The model for the sensors can have the ability of adding and subtracting different models of radar, lidar, or camera. The outer layer of the Simulink box could have outputs from the suite while inside that box are the sensors that the suite currently contains. This method could apply

containerization as well as the ability to write requirements and test for safety systems from the sensors themselves. The sensor fusion model could take the sensor suite model as inputs for a connected system. This would allow for faster prototyping with different sensors inside the sensor suite. The sensor fusion model could then use those inputs to model any algorithm that the developer may design. The outputs from the sensor fusion model could then input into the ACC model for a complete model of the ACC system. The ability of having these systems independent as well as connected inside a model would make the connectivity of the HIL interface easier. Breaking down each subsystem inside the model would allow for a more in-depth evaluation of the verification and validation method. This model would be easier to track changes and discuss issues with the individual subsystems in a repository. A change to a subsystem could be updated in the repository with comments of those changes. Lastly, the ACC system would connect to a model of the vehicle to get an overall picture of a simulated vehicle running adaptive cruise control.

Once the MIL environment is loaded onto the hardware as it would execute in the vehicle, the HIL environment could extend the ACC commands to the vehicle's control strategy hardware. The ACC system would have access to new information on vehicle operation in the VIL environment. This system would have the capability to monitor CAN bus and hardware loads of the vehicle before it is released on road for testing.

References

[1] SAE, “SURFACE VEHICLE RECOMMENDED PRACTICE Taxonomy and Definitions for Terms Related to Driving Automation Systems for On-Road Motor Vehicles,” 2021.

[2] J. Liu and H. Peng, “Modeling and control of a power-split hybrid vehicle,” IEEE Trans. Control Syst. Technol., vol. 16, no. 6, pp. 1242–1251, 2008, doi: 10.1109/TCST.2008.919447.

[3] J. A. Diethorn, “Implementation Of Fuzzy Logic Control Into An Equivalent Implementation Of Fuzzy Logic Control Into An Equivalent Minimization Strategy For Adaptive Energy Management Of A Minimization Strategy For Adaptive Energy Management Of A Parallel Hybrid Electric Vehicle Parallel Hybrid Electric Vehicle,” 2021. [Online]. Available: https://researchrepository.wvu.edu/etd

[4] A. R. Mull, “Powertrain Fuel Consumption Modeling and Benchmark Analysis Powertrain Fuel Consumption Modeling and Benchmark Analysis of a Parallel P4 Hybrid Electric Vehicle Using Dynamic of a Parallel P4 Hybrid Electric Vehicle Using Dynamic Programming Programming Recommended Citation Recommended Citation Mull, Aaron Robert, ‘Powertrain Fuel Consumption Modeling and Benchmark Analysis of a Parallel P4 Hybrid Electric Vehicle Using Dynamic Programming,’” 2021. [Online]. Available: https://researchrepository.wvu.edu/etd/10154

[5] P. Misra, “RoAdNet: Robust Adaptive Network for Information Diffusion in RoAdNet: Robust Adaptive Network for Information Diffusion in VANET VANET,” 2019. [Online]. Available: https://researchrepository.wvu.edu/etd/7476

[6] J. Liu and H. Peng, “Modeling and control of a power-split hybrid vehicle,” *IEEE Transactions on Control Systems Technology*, vol. 16, no. 6, pp. 1242–1251, 2008, doi: 10.1109/TCST.2008.919447.

[7] SAE International, “About,” *https://www.sae.org/about*, Jul. 28, 2022.

[8] SAE. (2016). *SAE J3016 Levels of Driving Automation*. SAE International. https://www.sae.org/news/2019/01/sae-updates-j3016-automated-driving-graphic#:~:text=The%20J3016%20standard%20defines%20six,graphic%20first%20deployed%20in%202016

[9] T. Crain, P. Jaworski, I. Kyriakopoulos, R. Blachford and B. C. Fabien, "Prototyping EcoCAR Connected Vehicle Testing System Using DigiCAV Development Platform," 2020 IEEE 3rd Connected and Automated Vehicles Symposium (CAVS), Victoria, BC, Canada, 2020, pp. 1-5, doi: 10.1109/CAVS51000.2020.9334669.

[10] MathWorks Support Team, "What are MIL, SIL, PIL, and HIL, and how do they integrate with the Model-Based Design approach?," *https://www.mathworks.com/matlabcentral/answers/440277-what-are-mil-sil-pil-and-hil-and-how-do-they-integrate-with-the-model-based-design-approach*, Apr. 18, 2022.

[11] ANL, "AVTC History," *https://avtcseries.org/about-avtc/avtcs-history/*, Aug. 15, 2022.

[12] ANL, "Past Competitions," *https://avtcseries.org/about-avtc/past-competitions/*, Aug. 15, 2022.

[13] GM Authority, "GM 2.5 Liter I4 Ecotec LCV Engine," *https://gmauthority.com/blog/gm/gm-engines/lcv/*, Jul. 28, 2022.

[14] W. Huang, Kunfeng Wang, Yisheng Lv and FengHua Zhu, "Autonomous vehicles testing methods review," 2016 IEEE 19th International Conference on Intelligent Transportation Systems (ITSC), Rio de Janeiro, Brazil, 2016, pp. 163-168, doi: 10.1109/ITSC.2016.7795548.

[15] J. Gao, W. Wu and O. -E. -K. Aktouf, "Adequate Testing Unmanned Autonomous Vehicle Systems - Infrastructures, Approaches, Issues, Challenges, and Needs," 2022 IEEE International Conference

on Service-Oriented System Engineering (SOSE), Newark, CA, USA, 2022, pp. 154-164, doi: 10.1109/SOSE55356.2022.00025.

[16] P. Junietz, K. Klonecki, W. Wachenfeld, and H. Winner, *Evaluation of Different Approaches to Address Safety Validation of Automated Driving; Evaluation of Different Approaches to Address Safety Validation of Automated Driving*. 2018. doi: 10.0/Linux-x86_64.

[17] C. Lv *et al.*, "Analysis of autopilot disengagements occurring during autonomous vehicle testing," *IEEE/CAA Journal of Automatica Sinica*, vol. 5, no. 1, pp. 58–68, Jan. 2018, doi: 10.1109/JAS.2017.7510745.

[18] J. Hu, T. Xu and R. Zhang, "Testing and Evaluation of Autonomous Vehicles Based on Safety of the Intended Functionality," 2021 6th International Conference on Transportation Information and Safety (ICTIS), Wuhan, China, 2021, pp. 1083-1086, doi: 10.1109/ICTIS54573.2021.9798586.

[19] N. Rajabli, F. Flammini, R. Nardone, and V. Vittorini, "Software Verification and Validation of Safe Autonomous Cars: A Systematic Literature Review," *IEEE Access*, 2020, doi: 10.1109/ACCESS.2020.3048047.

[20] M. Rodrigues, A. McGordon, G. Gest and J. Marco, "Developing and testing of control software framework for autonomous ground vehicle," 2017 IEEE International Conference on Autonomous Robot Systems and Competitions (ICARSC), Coimbra, Portugal, 2017, pp. 4-10, doi: 10.1109/ICARSC.2017.7964044.

[21] J. Gustavsson, "Verification Methodology for Fully Autonomous Heavy Vehicles," in *Proceedings - 2016 IEEE International Conference on Software Testing, Verification and Validation, ICST 2016*, Jul. 2016, pp. 381–382. doi: 10.1109/ICST.2016.42.

[22] V. De Oliveira Neves, M. E. Delamaro and P. C. Masiero, "An Environment to Support Structural Testing of Autonomous Vehicles," 2014 Brazilian Symposium on Computing Systems Engineering, Manaus, Brazil, 2014, pp. 19-24, doi: 10.1109/SBESC.2014.27

[23] J. E. Heikkinen, S. Gafurov, S. Kopylov, T. Minav, S. Grebennikov, and A. Kurbanov, “Hardware-in-the-loop platform for testing autonomous vehicle control algorithms,” in *Proceedings - International Conference on Developments in eSystems Engineering, DeSE*, Oct. 2019, vol. October-2019, pp. 906–911. doi: 10.1109/DeSE.2019.00168.

[24] U. Zivkovic, O. Dekic, Z. Lukac, and M. Milosevic, “HIL based solution for ADAS software development and verification,” in *IEEE International Conference on Consumer Electronics - Berlin, ICCE-Berlin*, Sep. 2019, vol. 2019-September, pp. 396–399. doi: 10.1109/ICCE-Berlin47944.2019.8966201.

[25] H. A. Kang and J. H. Lim, “The integration of multiple test benches for functional verification of the cooperative ADAS,” in *IEEE Vehicular Technology Conference*, Nov. 2020, vol. 2020-November. doi: 10.1109/VTC2020-Fall49728.2020.9348442

[26] T. Tettamanti, M. Szalai, S. Vass and V. Tihanyi, "Vehicle-In-the-Loop Test Environment for Autonomous Driving with Microscopic Traffic Simulation," 2018 IEEE International Conference on Vehicular Electronics and Safety (ICVES), Madrid, Spain, 2018, pp. 1-6, doi: 10.1109/ICVES.2018.8519486

Appendix

Date	Current Revision of PropSys	Current PropSys SSL	Current CAVs SSL
2/18/22	Integrate_ECMS_Controller_v30		A
2/22/22	Integrate_ECMS_Controller_v31		A
2/22/22	Golden Flash V3		A
2/23/22	Integrate_ECMS_Controller_v32		A
2/23/22	Golden Flash V3		A
2/24/22	Golden Flash V3		A
2/26/22	Integrate_ECMS_Controller_v33		A
3/1/22	Integrate_ECMS_Controller_v35		A
3/1/22	Golden Flash V3		A

Vehicle	Swimlane	Team members	Test Location
Blazer	PCM	Fraser, Kellett	WVU Airstrip
Blazer	PCM	Fraser	Parking Lot
Blazer	CAVs	Diethorn, Flanigan	WVU Coliseum
Blazer	PCM	Fraser	Parking Lot
Blazer	CAVs	Diethorn, ACC Subteam	WVU Coliseum
Blazer	CAVs	Diethorn, Flanigan, Vincent	WVU Coliseum
Blazer	PCM	Fraser	WVU Airstrip/Parking Lot
Blazer	PCM	Fraser	Parking Lot
Blazer	CAVs	Diethorn, Flanigan	WVU Coliseum

Start Mileage	End Mileage	CAVs Mileage	Total Hybrid Testing Time on Flash (Hours)	Link to Test Case
14595	14688	0	2 hours	Click Here
14688	14708	0	1 hour	Click Here
14708	14711	3	V3 - 0.5 hours in ICE	Click Here
14711	14716	0	30 minutes	Click Here
14716	14723	7	V3 - 2 hours in ICE	Click Here
14723	14736	13	V3 - 3 hours in ICE only	Click Here
14751	14845	0	3 hours	Click Here
14845	14847	0	20 minutes	Click Here
14847	14849	2	V3 - 30 minutes in ICE Only	Click Here

Link to Test Case Results	Testing Outcome
Click Here	Still see dead pedal occasionally. Need to create better torque shaping for motor.
Click Here	Testing dead pedal, saw spike in engine torque command, look in the controller for issue
Click Here	Low Pass filter needs adjusted in the ACC Cascading Controller
Click Here	Improvement with loss of torque command to the engine
Click Here	Continued to tune filters in the Cascading controller
Click Here	Mobileye Calibration
Click Here	No issues with loss of torque command from the engine
Click Here	Saw one instance of the torque command reaching its max and got kicked out
Click Here	Mobileye Calibration

Test Case Creation Guide

Test Case Writing:

1. Determine what you are testing and your goals.
2. From your goals, what are the hard fails, soft fails, and hard pass criteria?
3. Create folder for log files using the *Folder Saving Format* below.
4. Submit test case to GRAs and create a naming convention for the test case.
5. If approved, execute tests using the proper testing plans.
6. Save log files to the previously made folder using the *Log File Saving Format* below.

Folder Saving Format (Use **KEY** below to fill out):

- "TestCaseID"_"YYY"_"ZZ"

Log File Saving Format (Use **KEY** Below to fill out):

- Readable logs: "XX"_"TestCaseID"_"YYY"_"ZZ"_Read.ext
- Raw Logs: "XX"_"TestCaseID"_"YYY"_"ZZ"_Raw.ext
- Test Case: "TestCaseID"_"YYY"_"ZZ"_Log.xlsx

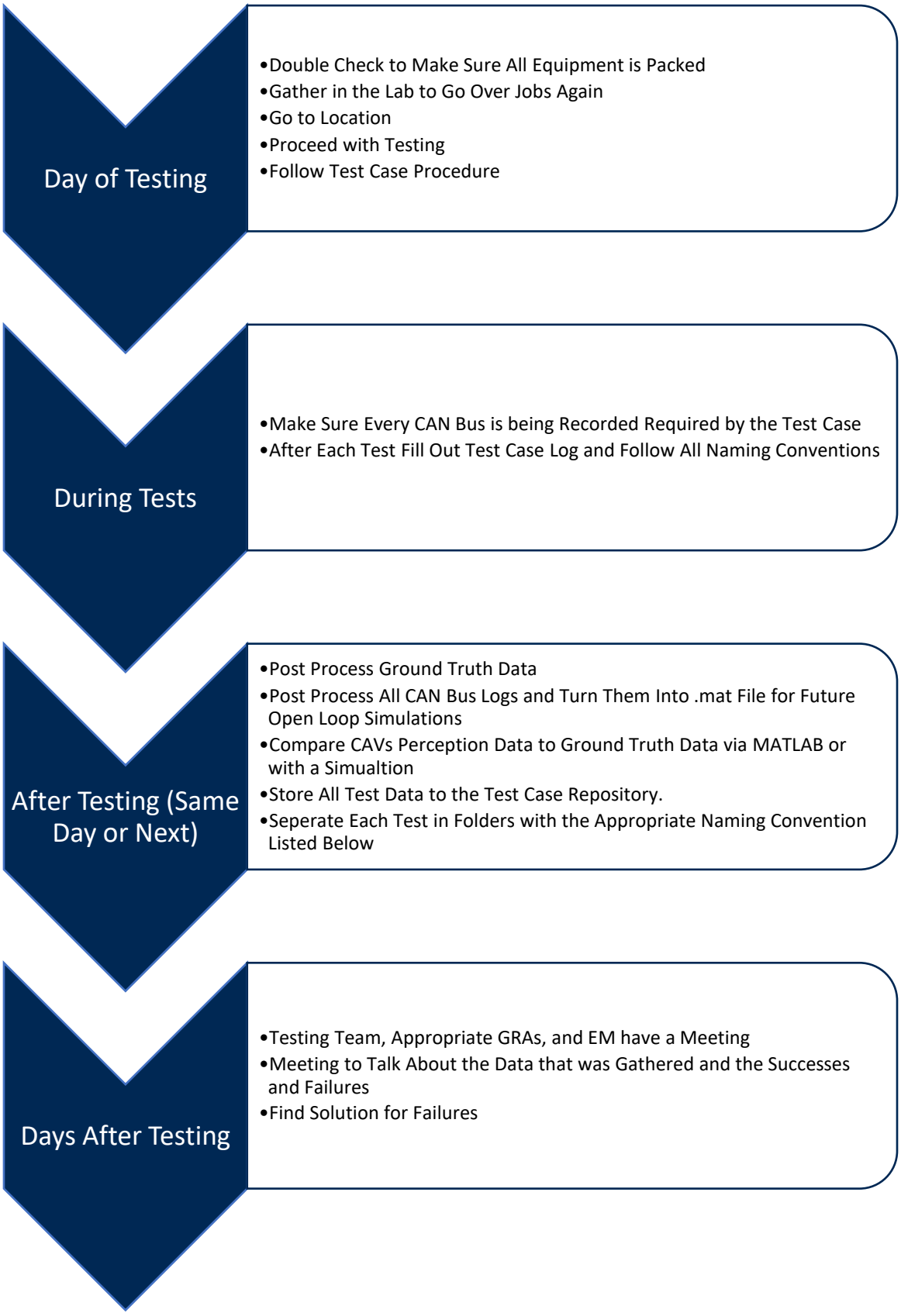
KEY(For use of naming conventions)**:**

XX: Device or Bus Name	**YYY**: Target Vehicle Description	**ZZ**: Set point
Front Radar = FR	Truck = TRK	10 mph = 10
Rear Radar = RR	SUV = SUV	20 mph = 20
Right Corner Radar = RC	Sedan = SDN	Varying Speed = VR
Left Corner Radar = LC	Truck Trailer = TRT	10 % pedal = 10
Webcam Footage = WF	Pedestrian = PED	20 % pedal = 20
Mobileye = ME	No Target Vehicle = NTV	Etc.
Fusion Object Bus = FO		
CAVs-PCM Interface = CP		
GM HighSpeed = HS		
Powertrain Expansion = PE		
Chassis Expansion = CE		
WVU_EV BUS = EV		
Multiple Buses = MB		

Test Case File and Log Locations:

Test Case ID	Test Case Description	Link to the Test Plan
0001	Approach Test	Here
0002	Follow Test	Here
0003	Passing Eco Vehicle From behind	Here
0004	Overtaking Eco Vehicle	Here
0005	Passing Eco vehicle Horizontally	Here


Some Type of Testing Needs to be Done
• Make Test Cases
• Approve Test Case with CAVs GRA
• If Test Case Exists Skip to Step 3
Submit New Test Case to Other GRAs
• GRAs Meet to Talk About Test Case to See if it is Valid
• GRAs Create Naming Convention for that Test Case if Approved
• Naming Convention gets Updated to Procedure Document
• Save New Test Case in Test Log Repository
Determine Best Possible Testing Site
• Schedule Testing Site with PM
• Schedule Testing Time with PM
Pre-testing Equpiment
• Make Sure All Testing Equipment is Properly Functioning
• Provide Training to Each Person that is Testing on Each Equipment on How to Operate.
Day Before Testing
• Have a Meeting with Everyone that is Going to Test
• Meeting will Give People Roles and Jobs for the Day of Testing
• Print Out Each Individual Test Case for Filing Later
• Gather All Testing Equipment Stated in Test Cases


Day of Testing
•Double Check to Make Sure All Equipment is Packed
•Gather in the Lab to Go Over Jobs Again
•Go to Location
•Proceed with Testing
•Follow Test Case Procedure
During Tests
•Make Sure Every CAN Bus is being Recorded Required by the Test Case
•After Each Test Fill Out Test Case Log and Follow All Naming Conventions
After Testing (Same Day or Next)
•Post Process Ground Truth Data
•Post Process All CAN Bus Logs and Turn Them Into .mat File for Future Open Loop Simulations
•Compare CAVs Perception Data to Ground Truth Data via MATLAB or with a Simualtion
•Store All Test Data to the Test Case Repository.
•Seperate Each Test in Folders with the Appropriate Naming Convention Listed Below
Days After Testing
•Testing Team, Appropriate GRAs, and EM have a Meeting
•Meeting to Talk About the Data that was Gathered and the Successes and Failures
•Find Solution for Failures

0001		**Testcase Edition:**	0001_SUV_35
Field		**Speed**	35 mph
Approaching directly behind a parked target vehicle		**Test Date:**	4/20/2022
		Test Performer:	Zachary Flanigan
		Flashed Version:	Final
OxTS hardware, WVU Chevy Blazer, target vehicle, logging hardware		**Test Location / Environment:**	DPG Yuma
Karah Little		**Weather:**	Sunny
4/9/2021		**Setup Issues:**	none
1) Park target vehicle a minimum of 250m facing away from the team vehicle and determine a starting line for testing 2) Survey the location of the target vehicle with OxTS 3) Follow instructions on the OxTS Manual to setup and warm u the RT units in the team vehicle 4) . Utilizing the OxTS data logging functionality, test that all of the correct signals are being logged before starting the official test runs		**Test Results & Issues for Truck:**	N/A
1) Drive team vehicle to predetermined starting line 2) Setup logging software and begin logging data 3) Begin driving car, accelerating to and maintaining the appropriate speed 4) Begin braking as the team vehicle approaches the parked car, coming to a stop directly behind it 5) Stop logging data 6) Ensure proper data was collected, and if not, perform test again		**Test Results & Issues for SUV:**	Stopped within the acceptable range. Sensor Fusion met all metrics. See .mat file for performance results
		Test Results & Issues for Sedan:	N/A
Data must be processed into the following formats: - CAN Log: Team's CAV-PCM bus - CAN Log: Team generation fused object detections - CSV (10 Hz): With ground truth and team generated fusion objects and named "EMC_Y3_CAV_Perception_ScenarioA_<speed>.csv" - MP4: Raw video recorded with a webcam		**Teardown Issues:**	None
Test fails if data is logged improperly		**Test Result:**	Passed
		Background/ Suggestions:	No suggestions
Sensor Fusion meets all performance metrics			
Sensor Fusion meets some performance metrics			
Sensor Fusion meets no performance metrics			

Testcase ID:	0002	**Testcase Edition:**	0002_SUV_35
Test Type:	Field	**Speed**	35 mph
Testcase Description:	Following a target vehicle for a set distance	**Test Date:**	4/21/2022
		Test Performer:	Zachary Flanigan
		Flashed Version:	Final
Required Equipment:	OxTS hardware, WVU Chevy Blazer, target vehicle, logging hardware	**Test Location / Environment:**	DPG Yuma
Testcase Author:	Clay Vincent	**Weather:**	Sunny
Written Date:	4/10/2021	**Setup Issues:**	none
Setup Procedure:	1) Ego Vehicle park behind target vehicle 2) Determine a finish line for testing 3) Survey the location of the target vehicle with OxTS 4) Follow instructions on the OxTS Manual to setup and warm u the RT units in the team vehicle 5) . Utilizing the OxTS data logging functionality, test that all of the correct signals are being logged before starting the official test runs	**Test Results & Issues for Truck:**	N/A
Test Procedure:	1) Setup logging software and begin logging data 3) Begin driving both vehicles, accelerating to and maintaining the appropriate speed 4) Cycle through the gap settings near, medium, and far at appropriate intervals 5) Begin braking target vehicle as the end of finish line nears and come to a stop 6) Bring the ego vehicle to a stop behind the target vehicle 5) Stop logging data 7) Ensure proper data was collected, and if not, perform test again	**Test Results & Issues for SUV:**	Followed within the acceptable range. Sensor Fusion met all metrics. See .mat file for performance results
		Test Results & Issues for Sedan:	N/A
Teardown / Post Processing:	Data must be processed into the following formats: - CAN Log: Team's CAV-PCM bus - CAN Log: Team generation fused object detections - CSV (10 Hz): With ground truth and team generated fusion objects and named "EMC_Y3_CAV_Perception_ScenarioA_<speed>.csv" - MP4: Raw video recorded with a webcam	**Teardown Issues:**	None
Pass/Fail Criteria:	Test fails if data is logged improperly or does not react to the target vehicle at the different gap settings.	**Test Result:**	Passed
		Background/ Suggestions:	No suggestions
Hard Pass Actions	Sensor Fusion meets all performance metrics and changes gap settings fluidly		
Soft Pass Actions	Sensor Fusion meets some performance metrics and changes gap settings fluidly		
Fail Actions	Sensor Fusion meets no performance metrics or gap changes has no affect on the driving		