Digital Twinning in Advanced Cruise Control (ACC): Enhancing Traditional Simulations for Modern Vehicle Systems

Hao-yu Wang

August / 2023

Supervisor Name: David Wagg

MSc Individual Report

submitted to the University of Sheffield in partial fulfilment of the requirements for the degree of MSc in Advanced Mechanical Engineering

ABSTRACT

The comparison of adaptive cruise control simulations using Digital Twin Technology for Improving advanced driver assistance systems (ADAS) Performance in the Automotive Industry. As vehicles become more autonomous, advanced driver assistance systems (ADAS), such as the ACC System, become increasingly important. To ensure successful real-world performance, ACC systems must be thoroughly validated. ACC systems have traditionally relied heavily on physics-based modelling and simulation. However, these traditional techniques are limited in terms of flexibility and realism.

The current research delves towards using digital twin simulation methods to improve ACC system assessment. Digital twins are virtual representations of physical items that are constantly synchronised. Regarding automotive systems, digital twins combined with real-time data and machine learning have enormous promise. It verifies the benefits of complementing traditional ACC simulations with data-driven digital twin methodologies through a comparative comparison. While physics-based models are still useful for initial testing, dynamic digital twin platforms are the future of ACC system development. Their ability to assimilate live data provides unrivalled representation accuracy. The technique for research includes MATLAB modelling of a fundamental ACC system, data synthesis, neural network integration, and digital twin simulation. The results show that the digital twin is superior at adjusting to various uncertain driving scenarios. This study demonstrates the significance of digital twins in developing smarter, safer autonomous cars.

CONTENTS

List of Figures

List of Tables

ACKNOWLEDGEMENTS

First and foremost, I wish to extend my deepest gratitude to Professor David Wagg for his invaluable guidance throughout this research journey. His unwavering support, profound insights, and constructive suggestions have been instrumental in shaping this study. His dedication to academic rigour and his commitment to his students are commendable, and I am privileged to have been under his mentorship.

I would also like to express my heartfelt thanks to my parents, whose unwavering belief in me and generous support made it possible for me to pursue my Master's degree in a foreign land. Their sacrifices, love, and encouragement have been my pillars of strength, motivating me to strive for excellence in all my endeavours.

Lastly, I would like to acknowledge the efforts of all those who indirectly contributed to this research, offering their perspectives, critiques, and encouragement. Your contributions have left an indelible mark on this work, no matter how small.

CHAPTER 1: BACKGROUND

1.1 Introduction

Advanced Driver Assistance Systems (ADAS) are now standard features in new automobiles, improving comfort, safety, and driving automation [9]. Adaptive Cruise Control (ACC) is a critical ADAS technology that allows vehicles to independently adjust velocity based on traffic circumstances [10]. As vehicles move closer to complete self-determination, the performance of ACC and other ADAS systems becomes increasingly important. Considerable testing and validation are required to verify that ACC systems operate successfully in a variety of real-world scenarios [11]. Traditionally, physics-based modelling and simulation approaches have been heavily used in the development of ACC systems [12]. However, there are drawbacks to utilising predetermined models, which lack flexibility and may fail to represent real-world uncertainty [13].

A developing simulation technique founded on Digital Twin concepts for ACC system testing appears promising. Digital Twins are virtual reproductions of their physical counterparts that are constantly connected [14]. Because of this link, Digital Twins can give an accurate, real-time representation of the physical system and its interactions [7]. Once applied to automotive systems, Digital Twins, combined with sensor data and machine learning, can recreate significantly more realistic scenarios [15]. This raises an exciting research question: Would digital Twin simulation techniques improve the assessment of ACC systems compared to traditional modelling approaches? Considering this topic will give automotive engineers and researchers vital insights into exploiting computer simulations for ADAS testing and development as cars move towards greater autonomy.

Considering the research on ACC systems, comparisons between models based on physics and data-driven Digital Twin simulations are rare. This study tries to fill that gap by conducting a technical assessment of the capabilities of both simulation systems. The findings are expected to validate the relevance of enhanced virtual environments in the development of smarter, safer automobiles.

1.2 Aims & Objectives

Aim:

To critically analyze and compare the traditional simulation methods with the Digital Twin simulation approach for the validation and testing of ACC systems

Objectives:

1. Understanding of ACC Systems:

To thoroughly understand the mechanics, features, and complexities of ACC systems.

2. Elucidation of Simulation Paradigms:

To analyse existing simulation approaches by comparing them with the ideas behind Digital Twin simulations.

3. Development and Testing:

Design and deploy conventional and Digital Twin-based ACC simulations and assess their effectiveness in various circumstances.

4. Comparative Assessment:

To generate a detailed comparison of conventional and Digital Twin simulation results, emphasising each technique's benefits, limits, and issues.

5. Issue Resolution through Digital Twins:

To investigate specific obstacles or concerns in traditional simulations and see how the Digital Twin technique can help to overcome or reduce them.

6. Recommendations for Future Work:

To forecast the future direction of ACC system testing, focusing on prospective upgrades, enhanced sensor integration, and the expanding role of Digital Twins in autonomous driving.

CHAPTER 2: LITERATURE REVIEW

2.1 Inroduction

In recent years, the emergence of 'digital twin' technology has presented transformative potential across many industries, including automotive. As defined by Grieves in 2014 [16], a digital twin constitutes a virtual representation of a physical asset, continually updated with real-world data to enable optimization and predictive analytics. To realize this, the adoption of cutting-edge technologies is essential, serving as the bedrock of the contemporary industrial revolution. A visual representation of these can be discerned in Fig.2.1.1

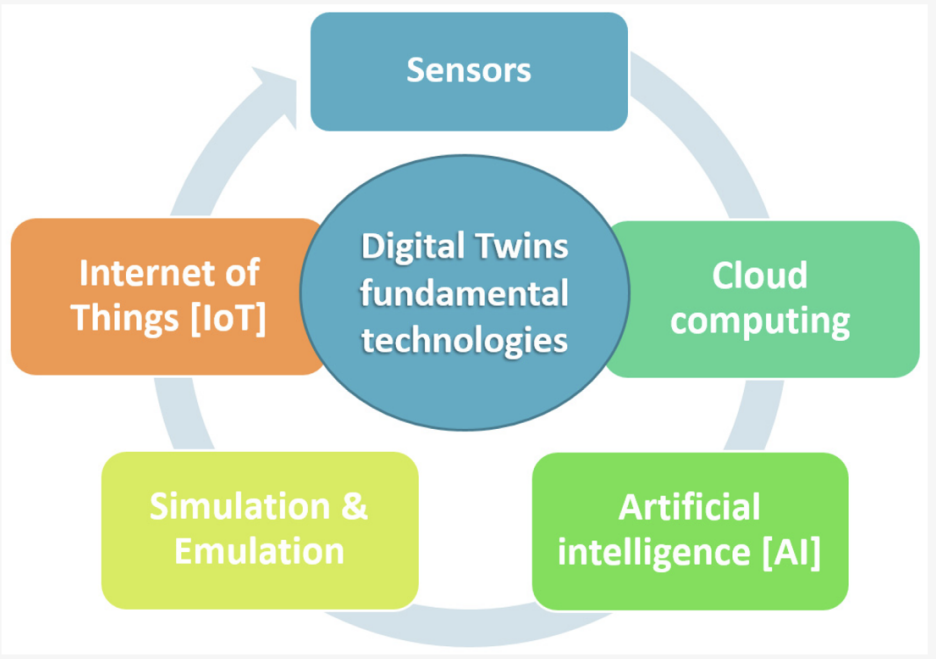

Figure 2.1.1: Digital twin fundamental technologies [1]

While the concept is gaining traction across sectors, the automotive industry stands to derive immense value by embracing digital twins. The overarching aim is to harness this digital representation to optimize design and operational efficiencies. Moreover, focusing on the intended ways a digital framework enables them is more important. This mapping in the digital world is facilitated by IoT platforms and software that is leveraged to create a digital representation of the physical asset. According to an analysis presented by Research and Markets, it is anticipated that in excess of 94% of all IoT Platforms will encompass a digital twinning capability. Moreover, 42% of executives from diverse industry sectors acknowledge the advantages of digital twinning, and 59% aim to integrate it into their processes by 2028 [17] . The industrial sector, in particular, is poised to benefit immensely, a significant 96% of vendors underscore the imperative nature of integrating IIoT APIs with digital

twinning functionalities, emphasizing its pivotal role in industrial verticals [17] . Gartner, a leading research and advisory company, has provided insights into the track of Digital Twin adoption in the corporate landscape. Their projections suggest that by the year 2027, over 40 percent of large-scale global corporations will be integrating Digital Twin technologies into their strategic initiatives. This adoption is not merely for the sake of technological advancement but is driven by the potential these technologies have to significantly enhance revenue generation [18]. Financial metrics further accentuate this optimism. A report by Global Market Insights evaluated the digital twin market at an impressive USD 8 billion in 2022. The future trajectory seems promising with projections indicating a Compound Annual Growth Rate (CAGR) of over 25% from 2021 to 2032. This meteoric ascent is attributed to the confluence of rapid IoT advancements and the ongoing march of Industry 4.0, both acting as catalysts stoking market demand [19].

2.2 Digital Twin Technology

The concept of digital twins originated at NASA in the early 2000s, where they were proposed as an innovative approach to monitor the condition of spacecraft using virtual models. As outlined by Grieves (2014), NASA envisaged digital twins as a way to mirror the lifespan of assets from inception to disposal [16]. It has various uses, from real- time remote monitoring and control in industry to risk as- sessment in transportation to smart scheduling in automotive industry, therefore it has received a lot of attention recently. As per Z. Hu and colleagues [2], Figure 2.2.1 illustrates the principal milestones in DT development.

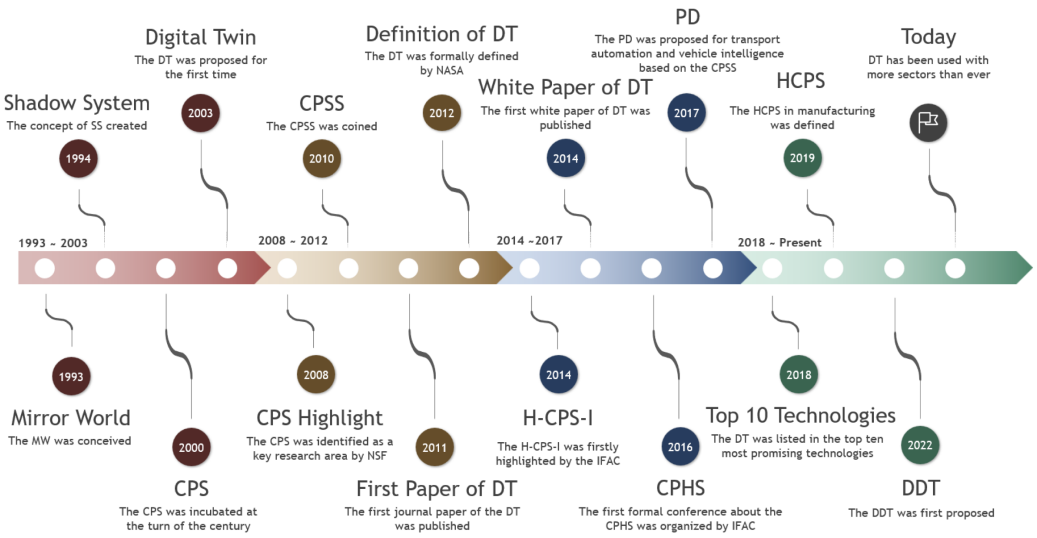

Figure 2.2.1: The vision of the digital twin throughout the product life-cycle. Source: [2]

In the rapidly advancing realm of automotive technology, high-fidelity virtual environments

have become increasingly pertinent. Simulation-based digital twins, within these environments, emerge as groundbreaking tools. Their primary advantage lies in their ability to expedite the verification process for Autonomous Vehicles (AV) [16]. A Digital Twin refers to a comprehensive simulation of a constructed vehicle or system. This simulation integrates multiple physical phenomena and scales, and incorporates probabilistic models. It utilizes the most advanced physical models, updates from sensors, historical data from the fleet, and other relevant inputs to reflect the experiences and conditions of its real-world counterpart in flight,as shown in Fig. 2.2.2.

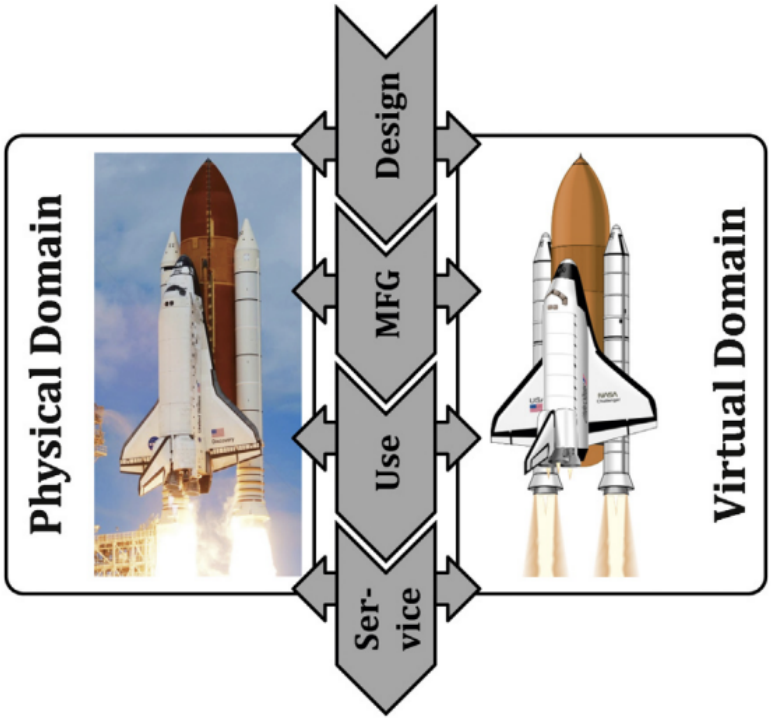

Figure 2.2.2: The vision of the digital twin throughout the product life-cycle. Source: [3]

Over the past two decades, the scope and maturity of digital twin technology has evolved considerably. Transitioning to the manufacturing sector, digital twins are enabling a fundamental shift from reactive to predictive systems. As noted by Schleich [20], traditional manufacturing workflows rely on detecting equipment failures and then reactively scheduling maintenance. In contrast, digital twins integrated with IoT sensor data and machine learning algorithms can continuously monitor equipment health and predict potential failures before they occur. The idea of mirroring is at the core of the digital twin concept. Every change, update, or event in the physical world gets reflected in the digital twin in real-time. This reflection is facilitated by a continuous flow of data between the physical entity and its virtual representation [21]. While traditional simulations of ACC systems have served their purpose, they often fall short in several areas, primarily due to their static nature and limitations in adaptability. According to the dynamic of digital twins in reflection of reality, which can be

addressed challenges in the following ways

Turning to the automotive sector, digital twin adoption is gaining momentum. In a recent study, Capgemini (2021) found over 60% automotive companies expect to implement digital twins by 2027 [22] . Pointing to real-world implementations, the BMW Group stands as a trailblazer in the automotive sector. In its production, the firm employs over 200 AI-driven solutions. Such advanced artificial intelligence technologies are harnessed to streamline operations in both logistics and manufacturing [23]. Ford uses digital twin technology to accurately detect energy losses, pinpoint areas where energy can be conserved, and improve the overall performance of production lines [24]. However, automotive companies still face challenges in digital twin adoption around factors like data management, integration complexity, standardised modelling and model/algorithm application [25]. As the technology matures, these issues are being actively addressed through solutions like cloud-based digital twin platforms, open standards, and out-of-the-box solutions. On the horizon, the proliferation of IoT devices and 5G connectivity will rapidly accelerate digital twin capabilities and use cases across the automotive value chain. Digital twins are poised to become a core enabler of data-driven design, manufacturing, and customer experiences for automotive firms. However, overcoming organizational and technical hurdles on the road to full-scale implementation remains an ongoing journey.

2.3 Digital Twin in the Automotive Industry

2.3.1 Industry Context

The automotive industry, long recognized as a linchpin of global innovation and technological progress, stands at a transformative crossroads. The current metamorphosis is spurred by environmental concerns, stringent emission regulations, and an accelerating global drive to transition from traditional combustion engines to sustainable electric vehicles (EVs) [13]. These shifts are not merely responses to environmental imperatives; they reflect a broader societal demand for greener, more efficient transportation solutions.

Amidst these tectonic shifts, the role of technology in shaping the future of mobility has never been more pronounced. Herein, the 'digital twin technology' concept emerges as a beacon. Originally conceived for aerospace applications, digital twins have found profound resonance in the automotive domain. These virtual replicas, synchronized with their real-world counterparts, are poised to revolutionize automotive design, testing, and manufacturing [16].

Digital twins, in essence, act as dynamic mirrors of physical vehicles. By continuously ingesting real-time data from myriad sensors embedded within a vehicle, combined with relevant regulatory and environmental data, they offer unparalleled insights [16]. This

fusion of data, as visualized in Fig.2.3.1, enables myriad applications—from predictive maintenance to real-time performance optimization. More crucially, in the context of user safety—a paramount concern in vehicular design—digital twins can simulate and predict vehicle responses in myriad scenarios, ensuring that vehicles not only meet but exceed safety standards.

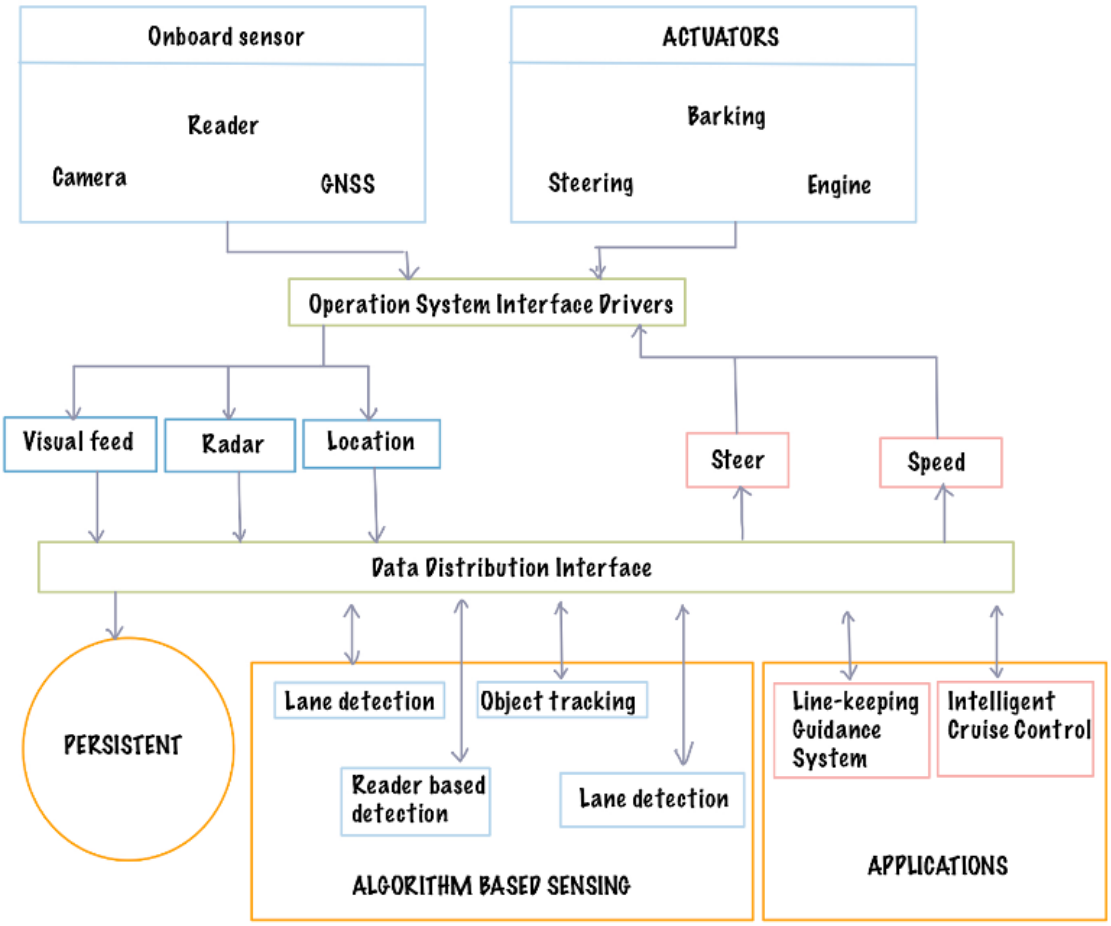

Figure 2.3.1: Driver assistant system.

.

2.3.2 Utilizing Digital Twins in Automotive Design and Testing

Digital twin applications related to automotive manufacturing and prototype testing have become essential across the sector. As noted by various researchers, digital twins enable enhanced efficiency, streamlined processes, faster time-to-market, cost reduction, and superior quality [1]. By generating insights not readily available from physical systems alone, digital

twins facilitate the optimization of new and existing automotive designs and production systems [2]. There are different digital twin configurations tailored to automotive applications, such as

- **Functional Prototype Twins (FPT)** - A simulation of a vehicle's operational behaviours and responses, providing foundational insights into its functional aspects using model-based systems engineering.
- **Geometry Twins** - Focused on the physical design and spatial arrangements of vehicle components, aiding in optimizing manufacturing and assembly processes.
- **Prototype Twins** - A comprehensive representation of a fully developed vehicle, valuable for scenario simulations and performance testing under various conditions.
- **Simulation Twins** - Concentrate on vehicles' software and electronic components, ensuring that embedded systems and software layers operate reliably and efficiently.

Based on Zsolt's study, a new X-in-the-loop (XiL) framework has been presented, utilizing Digital Twin (DT) and IoT capabilities [4]. This framework acts as a conduit between the virtual and actual representations of vehicles during the testing phase. It encompasses real-time vehicle simulation, authentication processes, synthetic testing configurations for automated vehicles, and a myriad of connectivity platforms. Figure 2.3.2 illustrates the simulation framework of the established scenario-in-the-loop (SciL) validation model, as derived from Zsolt's research.

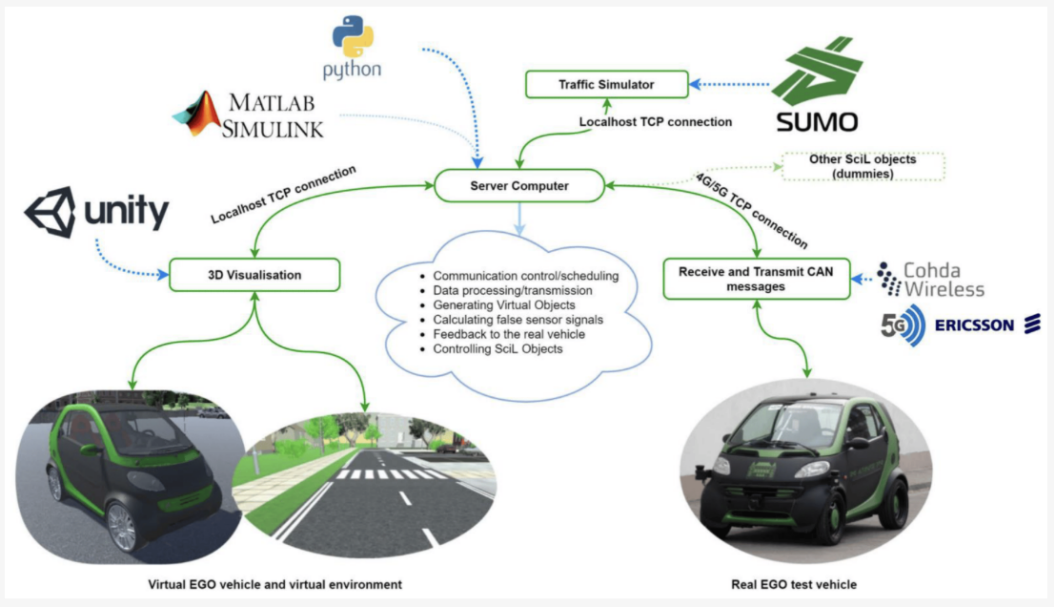

Figure 2.3.2: Co-simulation framework of the implemented proof-of-concept SciL validation model. [4]

Among these, Prototype Twins can have broad implications for reducing time and costs across testing phases and future design iterations. Similarly, Simulation Twins are crucial for developing and validating the increasingly complex software controlling modern vehicles [4].

While manufacturing and testing use currently predominate, the integration of digital twins across automotive value streams is increasing.

2.3.3 Digital Twins and Advanced Driver Assistance Systems (ADAS)

Advanced Driver Assistance Systems (ADAS) represent the forefront of technological advancement in the automotive industry. It offers supplementary data from the car's external environment to aid the driver and facilitate the execution of crucial actions. Designed to enhance both vehicle safety and driver convenience [9], among these, the Adaptive Cruise Control (ACC) system stands out as a seminal innovation. Adaptive Cruise Control refines the traditional cruise control by permitting the driver to trail a slower-moving vehicle ahead automatically. Consequently, the vehicle can integrate seamlessly into heavy traffic while maintaining a safe distance [10]. In the intricate world of ADAS, where real-time responsiveness is crucial, automotive radar and lidar sensors emerge as fundamental elements for the next generation of driver assistance features [26] Fig. 2.3.3.

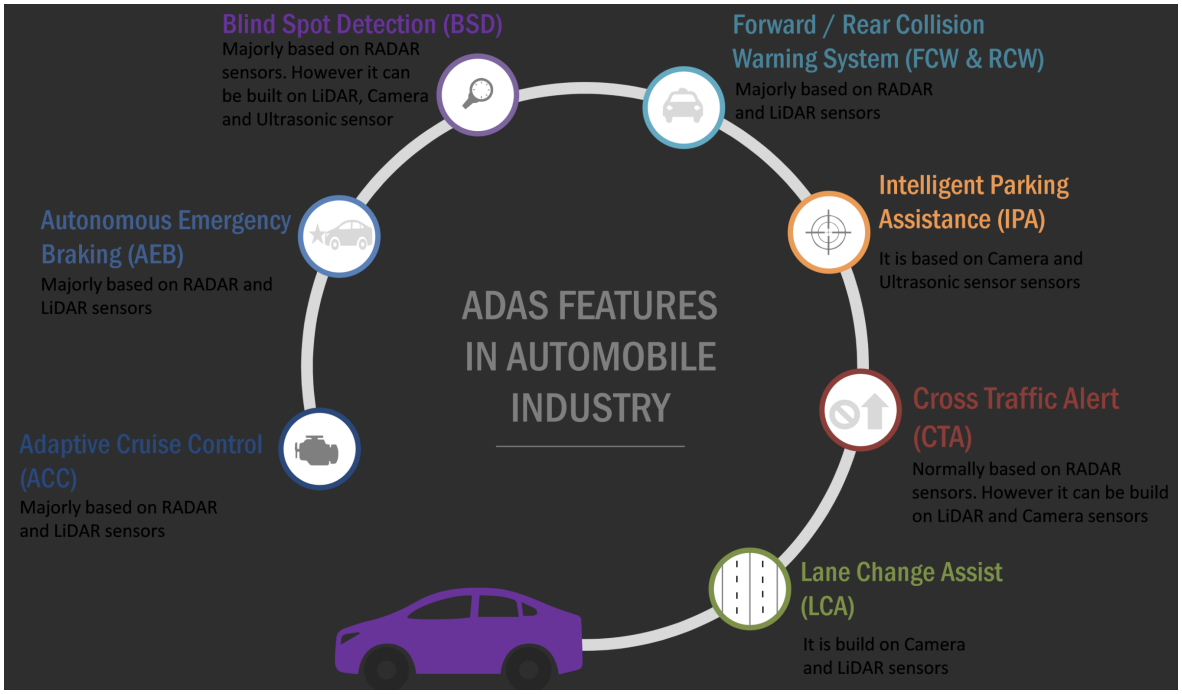

Figure 2.3.3: Applications of automotive LiDAR and RADAR sensors [5]

The role of Digital Twins is becoming indispensable. They are more than digital imitations; they serve as dynamic virtual representations within the cyber domain that reflect real-world entities. Furthermore, these evolving models enable real-time tracking and alignment of actual-world activities, capturing the true behaviour of vehicles. [27]. When implemented in systems such as ACC, Digital Twins simulate a multitude of driving scenarios, and the collision warning functions were evaluated under a range of various scenarios [28]. During

the process, they provide engineers and developers with a controlled environment to examine, fine-tune, and optimize ADAS features without the hazards associated with real-world testing [12]. However, the usefulness of Digital Twins goes beyond simple simulation. They consistently absorb real-time information from a multitude of sensors integrated into the vehicle [29]. This capability allows them to emulate the influence of external factors—severe weather, deteriorating road quality, or unexpected traffic congestion—on ADAS functionalities. Given this, Digital Twins are instrumental in ensuring the ACC system's resilience and the robust performance of other ADAS features across varied driving conditions. Their value is underscored by the insights they provide, accentuating their key role in system adaptability and resilience. As vehicles edge towards full automation, the seamless integration of myriad ADAS features becomes crucial [30]. Digital Twins assist in this integration process. They allow developers to visualize how different systems interact, identify potential conflicts or inefficiencies, and fine-tune the overall vehicle behavior [13]. While the early applications of Digital Twins were primarily rooted in manufacturing and testing, their horizons are rapidly expanding [21]. They are now integrated across the automotive value chain, from design and development to post-sales services [13]. As the automotive sector gears up for transformative changes, underscored by electrification and automation, the significance of Digital Twins will only grow [31]. They present the industry with a powerful tool to navigate these changes, ensuring that the next generation of vehicles remains safe, efficient, and user-centric.

2.4 Future Prospects of Digital Twins in Automotive

2.4.1 Future Landscape

In the rapidly changing landscape of the automotive industry, digital twins are poised to reshape every facet of vehicle design, production, and operation [21]. As vehicles become more intricate, digital twins' real-time data replication capabilities can greatly amplify systems like the adaptive cruise control (ACC), ensuring heightened responsiveness and precision that lead to a safer and more efficient driving experience [31]. At the same time, as the need for accurate digital twin simulations rises, advanced vehicle sensors are progressing [32]. These sensors, capturing expansive datasets, furnish digital twins with intricate specifics, allowing them to emulate real-world scenarios with unmatched precision. Such comprehensive virtual testing effectively models diverse driving conditions, equipping vehicles to adjust to various environments without the extensive costs and challenges of real-world evaluations. Beyond individual vehicles, the broader automotive production process stands to gain immensely [33]. Digital twins, reflecting the entire production life cycle, can precisely anticipate material needs, optimize inventory levels, and reduce waste substantially [33]. This approach augments cost efficiency and aligns with the growing emphasis on sustainable production. Moreover, with manufacturing settings often navigating complex material transfers and pro-

cedures, there is sufficient scope for optimization [33]. According to the prowess of digital twin simulations, optimal pathways for material transit can be identified, and assembly line processes refined, maximizing productivity while conserving resources [33]. The research from the IEA [6] shows a clear global trend towards vehicle electrification. As depicted in Fig. 2.4.1 and Fig. 2.4.2, electric vehicles have seen a consistent rise from 2010 to 2020. Digital twins can play a pivotal role in this realm, from optimizing EV battery life to enhancing charging infrastructure designs. They can also aid in integrating renewable energy sources for EVs, ensuring efficient energy management.

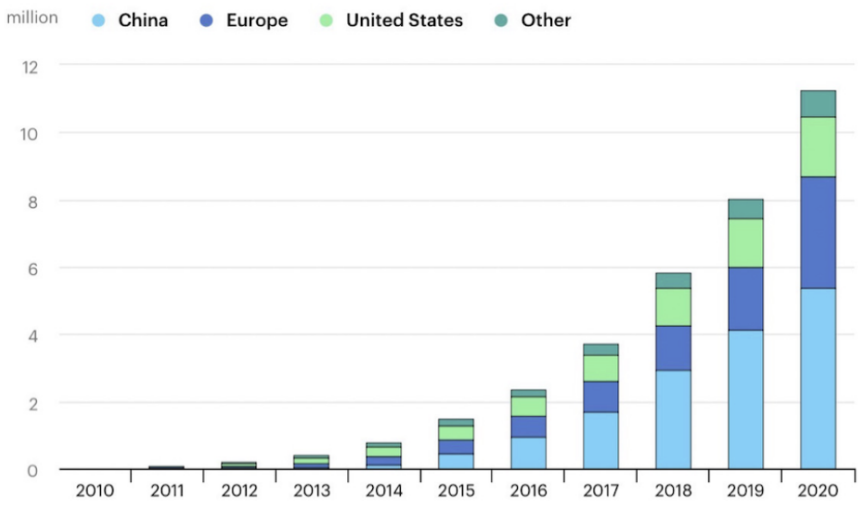

Figure 2.4.1: Global electric vehicle stock by region [6]

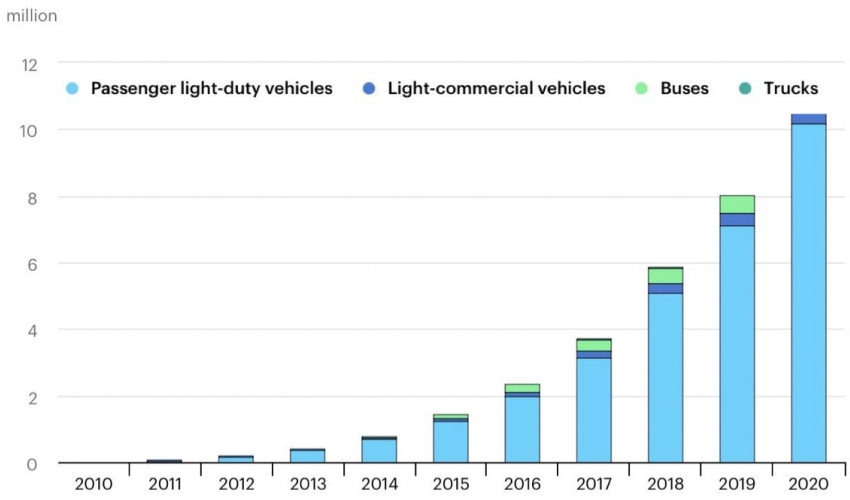

Figure 2.4.2: Global electric vehicle transport mode [6]

As the automotive realm transitions towards electrification and automation, digital twins solidify their role as an indispensable asset, offering the industry a robust roadmap for these transformative shifts [21].

2.4.2 Challenges in Autonomous Vehicles

In the domain of autonomous vehicles, safety and reliability are of paramount importance. Unlike traditional vehicles that rely on human reflexes and judgment, autonomous vehicles function through a complex interplay of sensors, software, and advanced algorithms. This change from human control to machine-driven operations brings distinct issues. A large array of sensors, such as LIDAR and RADAR, fuels the decision-making process in autonomous cars. Machine learning algorithms that understand and respond to sensor data supplement these. It is critical that these systems work flawlessly, both alone and collectively. For example, a defective sensor's misread signal might allow the program to make an incorrect judgement, potentially leading to harmful scenarios, according to koopman2017autonomous. [34]. Autonomous vehicles must navigate diverse driving conditions flawlessly, from bustling city streets to steady rural roads. When you factor in the unpredictability of weather conditions, such as rain, fog, or snow, it is necessary that vehicle software be equipped to manage an array of scenarios. Ensuring reliability across these varied conditions is a daunting challenge [35].

While simulations and controlled environments play a vital role in autonomous vehicle development, they might not entirely replicate the unpredictability of real-world conditions. A study by PLOS ONE mentioned that traditional simulations can sometimes overlook certain variables, causing differences between simulated outcomes and real-world results [36]. Addressing these differences is crucial as autonomous vehicles progress from controlled settings to real-world trials. Transitioning from simulations to actual driving introduces myriad challenges, necessitating a blend of technological advances, thorough testing, and stringent regulatory frameworks [37]. As vehicles edge closer to full autonomy, the tolerance for software errors diminishes. The software systems governing autonomous vehicle operations must showcase enough reliability. Even a minor failure or delay in real-time data processing can result in grave safety implications, underlining the importance of stringent software development and validation processes [11]. Incorporating Digital Twins can play a critical role, providing simulated landscapes to examine and fine-tune AV functionalities. This ensures that when these vehicles drive onto the streets, they do so with the highest levels of safety and efficiency.

2.4.3 Addressing Autonomous Vehicle Challenges through Digital Twin

The dynamic nature of digital twins enables the generation of realistic simulations, which can be continuously refreshed with real-world data, thereby enhancing the safety and reliability of autonomous driving systems [21]. This ensures that the scenarios tested are theoretical and based on actual driving conditions. Modern autonomous vehicles are equipped with amount of sensors, from LiDAR to ultrasonic sensors. Each of these feeds massive amounts of data

every second. Digital twins can efficiently integrate information from these diverse origins, offering a comprehensive perspective of the surroundings. It can process vast amounts of data in real-time, a vital capability for autonomous vehicles that need to make instantaneous decisions [38]. Digital twins can simulate myriad testing scenarios, from bustling urban landscapes to adverse weather conditions [21]. By continuously comparing the vehicle's real-time data with its digital counterpart, immediate alerts can be generated if a vehicle's response deviates from its digital twin's prediction. This diminishes the necessity for exhaustive real-world testing while ensuring the vehicle is equipped for diverse circumstances. Additionally, It is available to simulate entire traffic ecosystems through digital twins, including other vehicles, traffic lights, and pedestrian patterns [21]. Such simulations facilitate testing how autonomous vehicles mesh with existing infrastructure and their interactions with other roadway participants.

As autonomous vehicles navigate real-world situations, the data they accrue can be fed back into the digital twin [7]. This iterative feedback loop ensures that the digital twin is always learning and evolving, leading to continuous improvements in the autonomous driving system. The simulations are crucial in predicting the outcomes of decisions made by the autonomous system in specific scenarios. This predictive capability is vital for the system to choose between different action paths autonomously, distinguishing autonomous systems from mere automated ones [7]. Automated systems follow pre-set, rigorously designed action sequences. In contrast, as depicted in Figure 2.4.3, autonomous systems make decisions based on a deep understanding of the machinery, the tasks, and the environment. This understanding allows autonomous systems to adjust their actions in response to changes in product production volumes and even during automatic exception handling and error resolution, eliminating the need for human intervention or system reconfigurations. Based on that, autonomous systems are the pivotal enablers introducing the newfound flexibility that new industrial automation applications require.

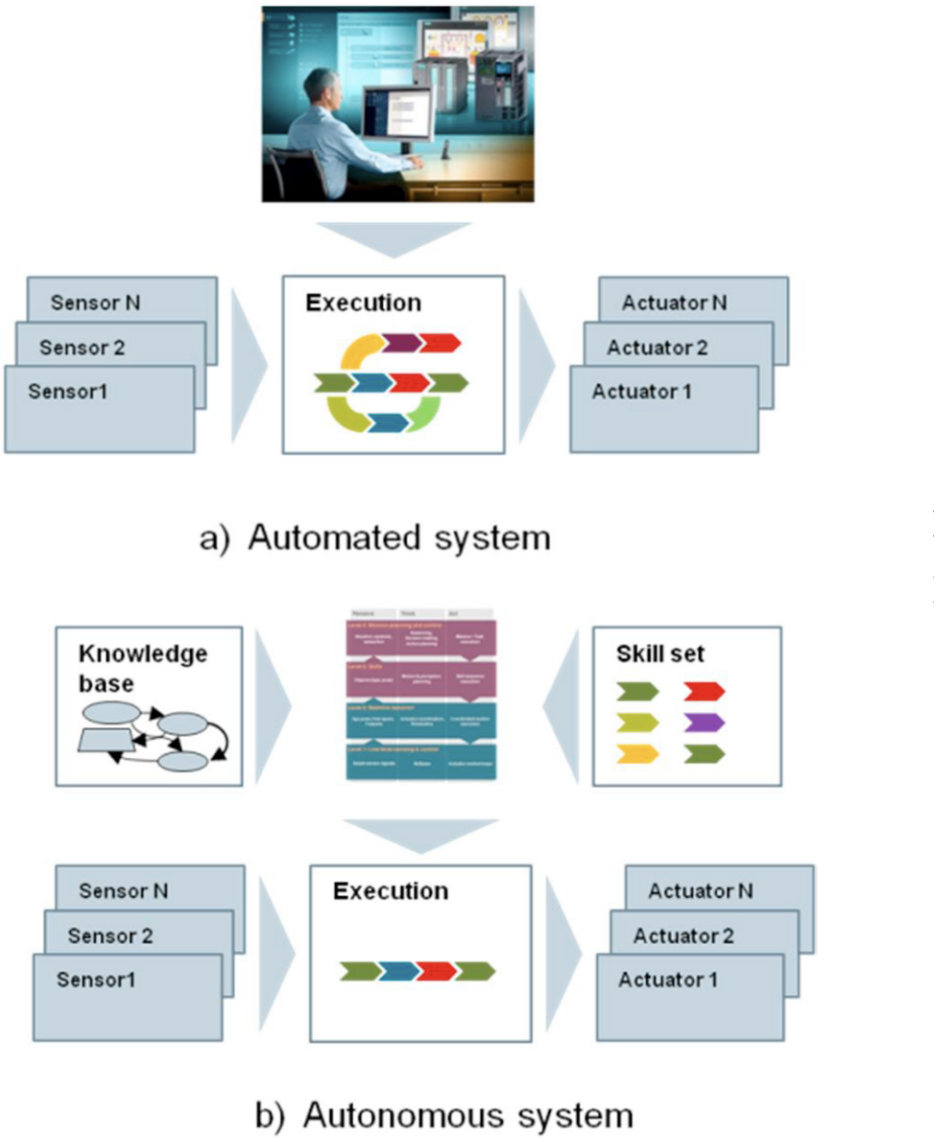

Figure 2.4.3: Difference between automated (a) and autonomous systems (b) [7]

Digital twins can provide observational data to regulatory bodies, demonstrating that the autonomous systems have been thoroughly tested and validated across diverse scenarios [21]. This could potentially expedite the authorization processes for autonomous vehicles. This iterative feedback loop ensures that the digital twin is always learning and evolving, leading to continuous improvements in the autonomous driving system [31]. As the landscape of autonomous vehicles continues its evolutionary trajectory, the integration of digital twins will be instrumental in ensuring their seamless and safe integration onto roadways.

CHAPTER 3: METHODOLOGY

3.1 Introduction to the Methodology

The methodology of this research is defined by the necessity to comprehend, model, and optimise the Adaptive Cruise Control (ACC) system. The research passes deeply into the specifics of ACC, analysing its promise and limits, using a two-pronged method that employs both standard simulations in Matlab and the sophisticated Digital Twin Simulation.

The technique used in this study included an extensive review of the ACC system. Initially, a standard Matlab simulation explains the fundamental concepts of the ACC system's operation. Building on this basis, the research moves on to Digital Twin Simulation. This sophisticated simulation applies machine learning concepts, notably neural networks, to simulate the ACC system's real-world functions. The combination of these two simulations enables a more in-depth study and optimisation of the ACC system.

The decision to use traditional as well as digital twin simulations derives from the requirement to test the ACC system's accuracy and efficiency in a variety of scenarios. Traditional simulations provide a controlled environment for analysing system behaviour and resolving basic challenges. Real-world circumstances, on the other hand, are frequently more complicated, needing Digital Twin Simulations. These simulations, enhanced by machine learning, provide a more accurate picture of real-world situations. Furthermore, the incorporation of machine learning, especially neural networks, is motivated by their capacity to handle complicated data structures and patterns, which improves the system's flexibility and accuracy.

3.2 Traditional Simulation of ACC System in Matlab

3.2.1 Principle of ACC System Operation

The ACC system operates as a sophisticated driver-assistance tool, expertly engineered to adjust a vehicle's speed autonomously to maintain a secure distance from the car in front. Through the use of radar, lidar, and additional sensors, the ACC system consistently tracks the distance and velocity of the vehicle ahead. The primary objective of ACC is to lessen the burden on the driver, bolster safety, and ensure a more fluid driving journey. There are two control modes: speed control mode and distance control mode.

Speed control: As depicted in Figure 3.2.1, the ego car proceeds at a speed determined by the driver.

If $D_{\text{safe}} \leq D_{\text{real}}$, the system operates in speed control mode, aiming to match the driver-

specified velocity, V_{set}.

Distance control: As illustrated in Figure 3.2.2, the ego car keeps a secure distance from the car in front.

If $D_{\text{safe}} > D_{\text{real}}$, the system shifts to distance control mode, with the primary objective of preserving the safe distance, D_{safe}.

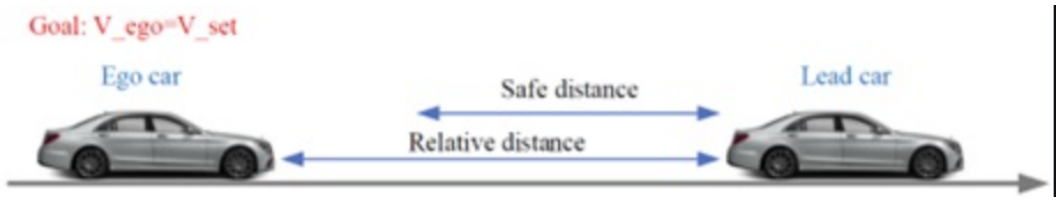

Figure 3.2.1: Speed control [8]

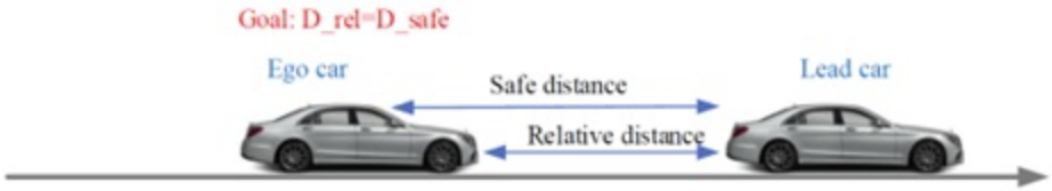

Figure 3.2.2: Distance control [8]

3.2.2 Model Construction in MATLAB and the Role of Laplace Transform

The simulation model is made by lead car, ego car and Adaptive Cruise System. The configuration of the model is shown in Fig 3.2.3. Initially, use a sine wave to represent the acceleration of the lead car. This output then becomes the input for the ACC system. The result is relayed to the ego car system upon processing in the ACC system, serving as its input (Fig. 3.2.3). The system computes the relative distance and velocity of the two vehicles. The ACC system then adjusts the speed of the ego car based on this calculation. The ego car's acceleration serves as the ACC system's output. The safe gap between the lead vehicle and the ego vehicle is determined by the velocity of the ego vehicle, denoted as V_ego:

$$D_{\text{safe}} = D_{\text{default}} + T_{\text{gap}} * V_{\text{ego}} \tag{3.1}$$

where D_{default} represents the default distance when both vehicles are at a standstill, and T_{gap} indicates the time interval between the two vehicles. D_{default} = 10m, T_{gap} = 1.5s.

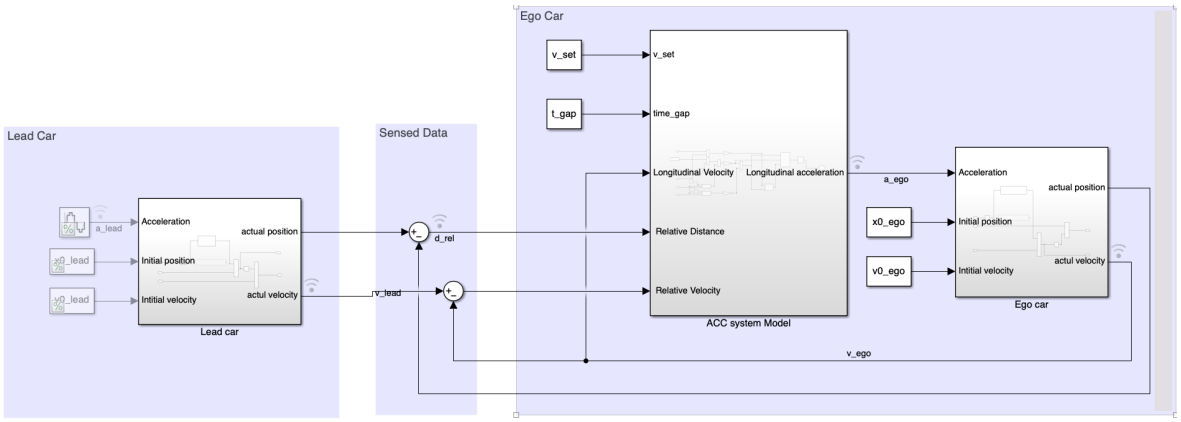

Figure 3.2.3: Configuration of model

The ACC system model in MATLAB provides a detailed representation of real-world vehicular dynamics. Employing the Simulink environment in MATLAB, as detailed in the project5_system.slx file, modelled each value of the ACC system, as shown in Table 3.2.1. (See Appendix A.4 for further details.)

```
%% Define the sample time, Ts, and simulation duration, T, in seconds.
Ts = 0.1;
T = 80; % T = 80 sec

%% Gain value
verr_gain = 1.5;
xerr_gain = 1;
vx_gain = 1;

%% Value of car
m = 2000;    % kilogram
b = 0;
a = 0;

%% PID
kp = 1.06;
ki = 3.2;
kd = 0.02;

%% Specify the driver-set velocity in m/s.
v_set = 35; % v_set = 30

%% Specify the initial position for the two vehicles.
x0_lead = 50; % initial position for lead car 50 (m)
v0_lead = 25; % initial velocity for lead car 25 (m/s)
x0_ego = 10;  % initial position for ego car 10 (m)
```

Table 3.2.1: Value of the model

The Laplace transform is central to our modelling. It is a mathematical tool that facilitates the transition from the time domain, where most physical processes occur, to the frequency domain, which is more suitable for analysis. The basic idea of the Laplace transform is represented by Eq. 3.2

$$F(s) = \int_0^\infty f(t)e^{-st}\,dt \tag{3.2}$$

Where $f(t)$ is our function in the time domain, and $F(s)$ is its representation in the frequency domain.

Ego Car and Lead Car Models with Laplace Transform: Constructing our models for the ego and lead cars, we initiate with differential equations that encapsulate their dynamic behaviours. The Laplace transform then aids in transitioning these differential equations into algebraic equations within the frequency domain, bestowing us with transfer functions for each vehicle. For instance, consider a rudimentary vehicular model where a car's speed.

$v(t)$ is an output, and throttle input $u(t)$ serves as an input. This relationship can be encapsulated with a differential equation:

$$m\frac{dv(t)}{dt} = u(t) - \beta v(t) \tag{3.3}$$

The dynamic simulations were modelled in Simulink, as shown in Fig 3.2.4, 3.2.5, 3.2.6.

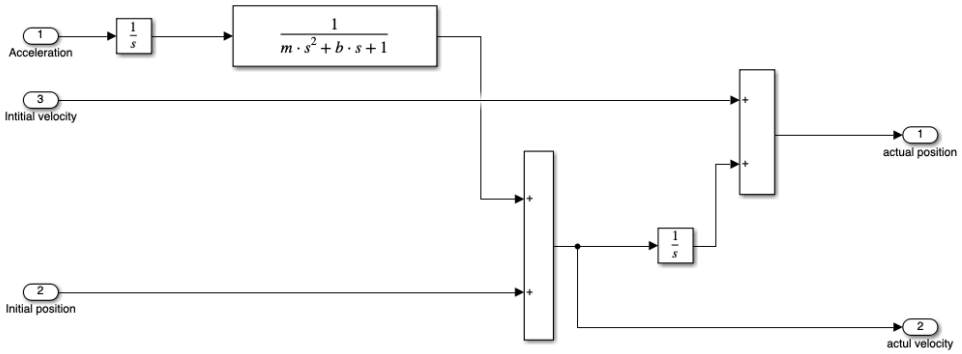

Figure 3.2.4: Ego car

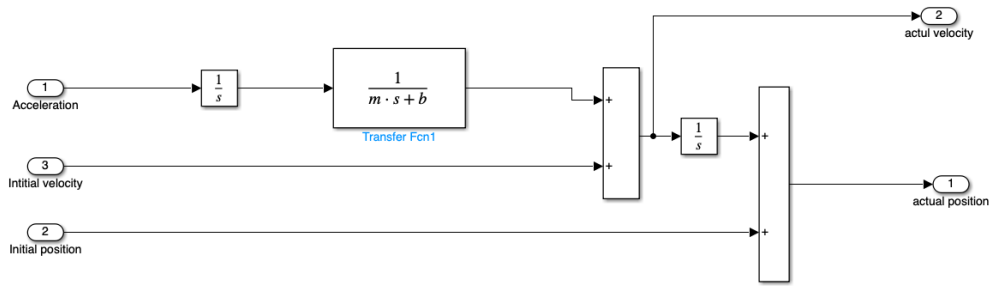

Figure 3.2.5: Lead car

The symbol m represents the vehicle's mass, and β stands for drag. Post the Laplace transformation, this equation offers a transfer function within the s-domain, affording pivotal insights into system stability and nuances of behaviour. The visual renditions in figures such as Figure 3.2.7 portray the system's response within the frequency domain. These illustrations underscore how fluctuations in the lead car's dynamics reverberate through the ego car's ACC system, shaping its response.

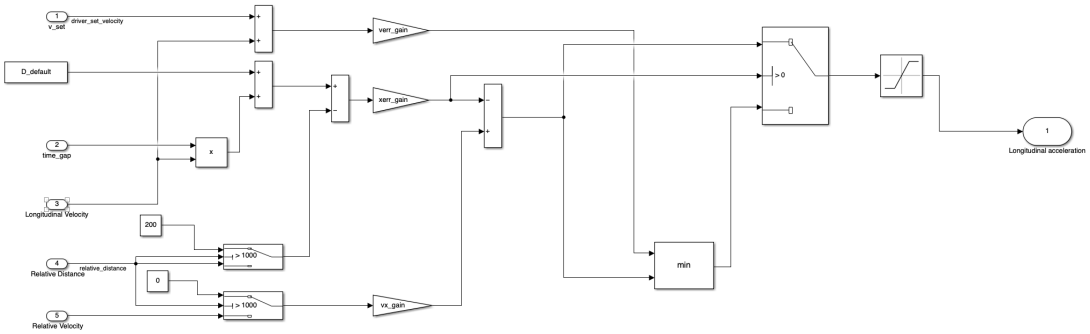

Figure 3.2.6: ACC system

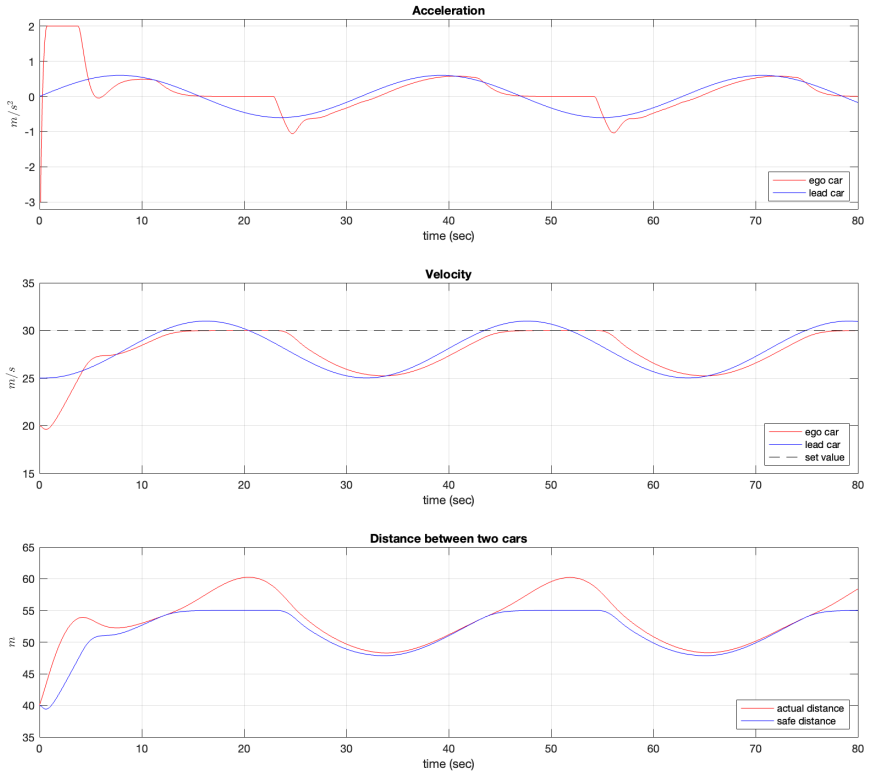

Figure 3.2.7: Result of simulation

3.2.3 Adaptivity in ACC

The flexibility of ACC distinguishes it from traditional cruise control systems. Instead of maintaining a consistent speed like conventional systems, ACC modifies its speed based on the actions of the vehicle in front. The ACC system reduces speed when the leading vehicle slows to keep the designated safe gap. On the other hand, if the road ahead is clear, the system resumes its previously set speed. This responsive adjustment guarantees the vehicle stays within a safe perimeter, averting possible accidents and ensuring a smooth journey.

3.2.4 PID Controller in ACC

The PID controller is a fundamental component of many control systems, and it plays a pivotal role in the ACC system's ability to maintain a safe following distance from the vehicle ahead while ensuring a smooth driving experience. The components of the PID controller are:

- **Proportional Control:**The proportional term produces an output value directly proportional to the current error value. The coefficient for this is often termed as the proportional gain, Kp; if the error is large, the proportional control response will be strong, and if the error is small, the response will be commensurate. This control is essential for quick responses, especially in scenarios where the preceding vehicle suddenly decelerates. However, relying solely on this could lead to system instability, especially if Kp is too large, as it might cause oscillations around the set point.
- **Integral Control:** This control is concerned with the accumulation of past errors. If the error has been present for an extended period, it will accumulate (integral of the error), and the controller will respond by changing the control input in a way that will reduce the error. This control's coefficient is the integral gain, Ki. This is especially significant when dealing with system biases or steady-state errors. However, too much of this control can lead to overshoots and oscillations.
- **Derivative Control:** Derivative control is predictive. It gauges how quickly the error is changing and acts in anticipation. This control's coefficient is the derivative gain, Kd. This control can foresee the system's future behaviour and thus counteract it before it can manifest. It's essential for stabilizing the system and preventing potential overshoots. However, if $Kd$is too high, it might lead to a "choppy" response as the system will attempt to correct itself too aggressively.

By the analysis of the result, if oscillations around a set point or reference value, especially when there is a disturbance, it might indicate that the proportional gain Kp is set high. For instance, the preceding vehicle suddenly brakes.

If the ACC system seems to be lagging in response and does not reach the desired speed or

distance quickly, the integral gain Ki might be low.

On the other points, if the ACC system appears to anticipate changes and adjusts the vehicle's speed before reaching the set distance from the vehicle ahead, it indicates that the derivative gain Kd is playing a significant role.

The precise tuning of these parameters, Kp, Ki, and Kd, is of paramount importance to ensure the ACC system's optimal performance. The plots and the corresponding system behaviour offer insights that can be used to fine-tune these parameters for better system stability and performance. Additionally, in practical scenarios, PID tuning is a cyclical process. Starting values are derived from the system's characteristics, which are then subjected to simulations or actual tests, leading to further adjustments.

3.2.5 Results & Discussion

The results derived from MATLAB simulations testify to the ACC system's effectiveness and precision in diverse traffic scenarios. As evinced in the attached Figures 3.2.7 and 3.2.8. The ACC system skillfully keeps a safe trailing distance regardless of the traffic scenario, whether open roads or heavy traffic congestion. One salient observation is the system's stability, a consequence of judicious PID parameter tuning. Adjusting these parameters can further amplify the system's accuracy, making it either more responsive or delivering a smoother ride, contingent upon the desired outcome. As clearly seen in Figure 3.2.8, tuning the parameters of PID is crucial. An overly high derivative gain can cause the system to become unstable, especially if the sensor data is noisy. Otherwise, the proportional term produces an output value proportional to the current error. If the proportional gain is set too high, it can cause the system to oscillate and become unstable. (See Appendix A.4 for further details.)

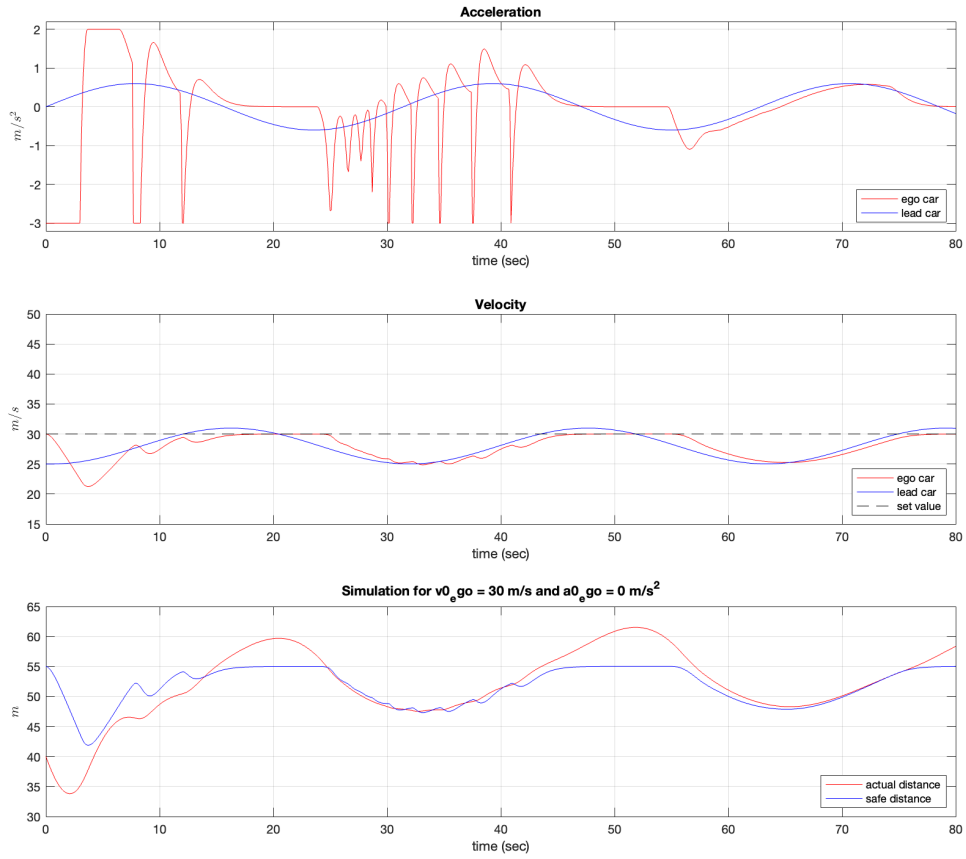

Figure 3.2.8: Result of simulation (8)

3.3 Digital Twin Simulation

3.3.1 Principle of Digital Twin Simulation

Digital Twin Simulation represents a convergence of the physical and virtual worlds, facilitating real-time monitoring, evaluation, and improved decision-making. The principle behind it is to create a virtual representation or 'twin' of a physical entity, system, or process. This 'twin' serves as a mirror, reflecting its physical counterpart's state, behaviour, and overall health. There are five main functions of the digital twin:

1. Reflection of Reality: The idea of mirroring is at the core of the digital twin concept. Every change, update, or event in the physical world gets reflected in the digital twin in real time. This reflection is facilitated by a continuous flow of data between the physical entity and its virtual representation [21]. While traditional simulations of ACC systems have served their purpose, they often fall short in several areas, primarily due to their static nature and limitations in adaptability. According to the dynamic of digital twins in reflection of reality,

which can be addressed challenges in the following ways:

- **Real-time Feedback Loop:** Traditional simulations often operate on present datasets, which can become outdated or may not represent current real-world conditions. Digital twins, with their continuous feedback loop, ensure real-time data integration, offering a more accurate representation of current conditions and allowing ACC systems to respond more effectively [7].
- **Predictive Capabilities:** Digital twins cannot just reflect the current state of the ACC system but can be predicted for future states based on data trends. This is a significant leap from traditional simulations, which are largely reactive. Predicting potential hazards or traffic flow changes in advance can enhance the safety and efficiency of ACC systems.
- **Dynamic Adaptability:** Traditional simulations can become obsolete as the system or environment evolves. In contrast, digital twins are inherently adaptable. As vehicles get updated, road conditions change, or the ACC system evolves, the digital twin logically integrates these changes, ensuring the simulation remains relevant.
- **Iterative Learning:** Digital twins facilitate iterative learning. As the ACC system operates, the digital twin collects data on performance challenges. This data is then used to refine the ACC system's algorithms, ensuring continuous improvement. Traditional simulations lack this self-refining capability, an important difference between digital twins simulation and traditional simulation.
- **Addressing Data Limitations:** Traditional simulations often grapple with limited datasets, hindering their accuracy. Digital twins, with their capability to synthesize data, can address gaps, ensuring a richer, more detailed simulation environment.

2. Proactive Predictions: Though essential in many contexts, traditional ACC system simulations primarily operate reactively. They function within set boundaries, processing inputs using a predefined model to generate outcomes. This model remains constant, not adapting to new insights or patterns unless manually updated. As such, the system's outputs are dictated by the inputs, and it does not possess the capability to learn from fresh data or adjust based on newfound insights.

On the other hand, digital twins introduce an innovative approach to simulation. At their core, digital twins are dynamic and continuously updated by absorbing real-time data from a variety of sources [21]. This ensures that they consistently mirror the system's current state they are replicating. Beyond just presenting real-time data, digital twins tap into a vast reservoir of historical data, allowing them to spot patterns or irregularities that might be overlooked in more limited datasets. With this combination of historical and live data, digital twins are positioned to make well-informed and accurate predictions.

However, integration with advanced algorithms is the critical element that elevates the digital twin. Instead of merely processing data, these algorithms, often rooted in artificial intelligence

and machine learning, enable digital twins to adapt and refine their models continually. They learn and adjust with every piece of data, enhancing their predictive capabilities. Using the ACC system as an illustration, a digital twin could proactively identify changes in traffic patterns based on past and present data, ensuring optimal speed adjustments, smoother drives, and better fuel consumption.

Digital twins' predictive nature revolutionizes system management. Identifying potential system issues or inefficiencies facilitates timely interventions and preventive maintenance. This approach not only reduces interruptions but also extends the system's lifespan. While traditional simulations offer a static glimpse based on set inputs, digital twins provide a continuously updated narrative, reinforcing their importance in modern system design and oversight [31].

The forward-looking nature of digital twins represents a significant advancement over traditional simulations. Their dynamic nature, rich data foundation, and proactive approach make digital twins a cornerstone for creating resilient and efficient systems [31].

3. Integration with Advanced Technologies: The potential of digital twin simulations is promising when integrated with machine learning, artificial intelligence, and IoT technologies [39] [40]. Machine learning, a subset of AI, is at the heart of the digital twin's decision-making prowess [41]. Machine learning algorithms allow the digital twin to continuously process and learn from vast amounts of data [42]. Over time, as the system encounters more data and scenarios, its predictions become more accurate [43]. This ongoing learning cycle ensures that the digital twin stays adaptable, can address unexpected challenges, and refine processes in real-time. Furthermore, advanced AI algorithms can simulate complex scenarios or problems, providing insights that might be missed by human analysis [40].

4. Feedback Loop: The feedback loop is a critical component of the digital twin principle. As the physical system operates, data is fed into the digital twin, which then processes this data, potentially using machine learning algorithms. The results, predictions, or insights are then fed to the physical system, informing decisions or actions.

5. Applications and Evolution: While the concept of digital twins is not entirely new, their applications have grown exponentially with the advent of Industry 4.0. From manufacturing to automotive systems like ACC, digital twins are revolutionizing, simulating, predicting, and optimising complex systems. The Digital Twin Simulation bridges the traditional ACC system and machine learning in the research context. It aims to enhance the ACC system's adaptability, accuracy, and overall performance using digital twins' principles. While traditional simulations have been instrumental in the initial development and understanding of ACC systems, digital twins represent the next frontier with their dynamic, predictive, and holistic capabilities. They address traditional simulations' limitations and offer avenues for innovation, optimization, and enhanced safety in ACC systems.

3.3.2 Data Generation and Synthetic Data

Digital Twins thrive on data. A digital twin model's precision, adaptability, and accuracy are directly proportional to the quality and quantity of data it processes. A digital twin to mirror its real-world counterpart effectively requires a continuous stream of real-time data. While the ideal scenario is to have real-time data flowing into the digital twin model, obtaining such data is often fraught with challenges. These can range from sensor malfunctions and transmission delays to data security concerns. Given the challenges associated with real-time data, synthetic data emerges as a viable alternative. Synthetic data refers to data that has not been directly obtained from real-world events but has been generated or simulated to mirror real-world data.

For instance, in the ACC system of this research, real data is not sufficient to support; therefore, synthetic data was created to simulate various driving scenarios, vehicle types, and road conditions in this simulation. Though not from actual driving experiences, this data provides a rich dataset that closely mimics real-world scenarios.

There are three primary benefits of synthetic data:

- **Diversity:** Synthetic data could have been developed for modelling an extensive spectrum of circumstances, a number of which might be infrequent or hazardous in the actual world but are required for accurate evaluation.
- **Control:** Researchers have total control over data generated artificially, allowing them to experiment with different settings or situations to observe the circumstance.
- **Privacy:** Synthetic data, particularly when created using algorithms, guarantees that sensitive information in real-world data is not disclosed.

Synthetic Data Generation Using Python Script:

Python script 'sensor_data.py 'exemplifies the process of creating this synthetic data for the ACC system:

1. Defining the Data Landscape: The database's cardinality is predetermined, guaranteeing an extensive collection of data points to draw. This cardinality is critical for establishing statistical significance and improving the robustness of the subsequent analysis. The dataset's size in the model is predetermined at 1000 samples, ensuring a vast pool of data points for robust analysis.(See Appendix A.3 for further details.)

```
# 1. Synthetic Data Generation
np.random.seed(0)
num_samples = 1000
```

2. Simulating Vehicular Dynamics:

Relative Speeds: A basic premise of ACC systems is the interaction of vehicle speeds. A spectrum of relative speeds is constructed using a uniform distribution, illustrating circumstances ranging from frantic overtaking to a more comfortable vehicle pace. Randomly generated speeds between -10 and 10, representing the speed difference between our vehicle and the one ahead.

Distances: Vehicle separation is neither continuous nor predictable on the road. As before, a uniform distribution gives a realistic simulation of varied inter-vehicle distances. Randomly generated distances between 5 and 100, representing the gap between the vehicle and the one ahead

```
relative_speeds = np.random.uniform(-10, 10, num_samples)
distances = np.random.uniform(5, 100, num_samples)
```

Vehicle Types & Road Conditions: A vehicle ecology is rich in variety. The software mimics a variety of vehicle kinds, each with its own set of dynamics, using random numbers. Furthermore, road conditions, which considerably impact automobile behaviour, are produced at random to introduce diversity. Random integers between 0 and 2, simulating different types of vehicles such as cars. And random integers between 0 and 2 illustrate different road conditions: dry, wet, and icy.

```
vehicle_types = np.random.randint(0, 3, num_samples)
road_conditions = np.random.randint(0, 3, num_samples)
```

3. Predictive Modeling Through Speed Metrics: The script defines a target variable based on the relative speeds. This is typical of real-world driving when relative speed measures frequently determine vehicle reactions.

```
X = np.vstack((relative_speeds, distances, vehicle_types,
   road_conditions)).T
y = np.where(relative_speeds < -1, 2, np.where(relative_speeds > 1,
    1, 0))
```

4. Neural Network for Synthetic Data Generation

Architecture and Initialization: The network consists of three layers: an input layer, a single hidden layer, and an output layer. At the start, the weights linking these layers are set to modest random values. The hidden and output layers' biases are set to zero.

```
class SimpleNeuralNetwork:
    def __init__(self, input_size, hidden_size, output_size,
   learning_rate=0.01):
        self.weights_input_hidden = np.random.randn(input_size,
   hidden_size) * 0.1
        self.bias_hidden = np.zeros(hidden_size)
        self.weights_hidden_output = np.random.randn(hidden_size,
   output_size) * 0.1
        self.bias_output = np.zeros(output_size)
        self.learning_rate = learning_rate
```

Activation Functions: The Rectified Linear Unit (ReLU) is the network's activation function for the hidden layer. The ReLU's processing efficiency and capacity to handle non-linearities influenced this choice.

```
    def relu(self, x):
        return np.maximum(0, x)

    def relu_derivative(self, x):
        return np.where(x > 0, 1, 0)
```

The softmax function is used for the output layer, guaranteeing that the network's output represents probability distributions across the probable outcomes.

```
    def softmax(self, x):
        z = x - np.max(x, axis=1, keepdims=True)
        numerator = np.exp(z)
        denominator = np.sum(numerator, axis=1, keepdims=True)
        return numerator / denominator
```

Training the Neural Network: A basic gradient descent optimisation approach is used to train the network. The mean squared error between projected and actual outputs is the loss function. The network computes the error and changes the weights and biases to minimise it at each epoch.

```
   def train(self, X, y, epochs=1000):
        for epoch in range(epochs):
```

```
            hidden_layer = self.relu(np.dot(X, self.
    weights_input_hidden) + self.bias_hidden)
            output_layer = self.softmax(np.dot(hidden_layer, self.
    weights_hidden_output) + self.bias_output)
            loss = np.mean(np.square(y - output_layer))
            output_error = output_layer - y
            hidden_error = output_error.dot(self.
    weights_hidden_output.T) * self.relu_derivative(hidden_layer)
            d_weights_output = hidden_layer.T.dot(output_error)
            d_bias_output = np.sum(output_error, axis=0, keepdims=
    True)
            d_weights_input = X.T.dot(hidden_error)
            d_bias_input = np.sum(hidden_error, axis=0, keepdims=
    True)
            self.weights_hidden_output -= self.learning_rate *
    d_weights_output
            self.bias_output -= self.learning_rate * d_bias_output.
    sum(axis=0)
            self.weights_input_hidden -= self.learning_rate *
    d_weights_input
            self.bias_hidden -= self.learning_rate * d_bias_input.
    sum(axis=0
```

Prediction: When trained, the network could generate predictions based on previously unknown data.

```
    def predict(self, x):
        hidden_layer = self.relu(np.dot(x, self.
    weights_input_hidden) + self.bias_hidden)
        output_layer = self.softmax(np.dot(hidden_layer, self.
    weights_hidden_output) + self.bias_output)
        return np.argmax(output_layer, axis=1)
```

The structure and execution of this script demonstrate the Control principle in synthetic data creation. Every parameter, distribution, and random generation is rigorously tuned to imitate real-world dynamics. Furthermore, the algorithmic generation of this data assures that it is devoid of any real-world sensitivities, emphasising the privacy benefit.

The 'sensor_data.py' script emphasises the power of synthetic data in filling real-world data gaps. It creates a data narrative that is comprehensive and representative of real-world automotive interactions using algorithmic discipline and statistical approaches, increasing the realism of the ACC system's digital twin simulations.

3.3.3 Machine Learning in ACC

Machine Learning (ML), a type of artificial intelligence, has quickly emerged as a valuable technique for improving the capabilities of Adaptive Cruise Control (ACC) systems. The core of machine learning is its capacity to learn from data, recognise patterns, and make intelligent judgements without being explicitly programmed. This section moves into the use of machine learning techniques in ACC and their revolutionary influence on vehicle safety and performance.

While productive, traditional ACC systems depend on established algorithms and principles. They are unable to respond in real-time to the constantly evolving dynamics of road conditions, traffic variances, and driver behaviour. ML provides flexibility to the equation. ML models can predict and respond to events considerably more effectively than traditional rule-based systems due to the fact they can handle massive volumes of data from sensors, cameras, and radars.

a. Types of Machine Learning Models in ACC:

In ACC systems, several machine learning models have been tested:

- **Random Forest:** A flexible model is noted for its simplicity and interpretability. It is extremely beneficial when the data includes a lot of characteristics, which is prevalent in vehicle datasets.

- **Gradient Boosting Machines:** Gradient Boosting Machines: These machines are popular due to their efficiency and precision. It sequentially constructs trees, with each tree correcting the mistakes of its predecessor.

While both models provide reliable predictions, their static character may not adequately reflect the temporal dynamics of driving events. As a result, the use of neural networks was chosen, as evidenced by the synthetic data production procedure.

b. Neural Networks in ACC:

Various machine learning algorithms have been included in Adaptive Cruise Control (ACC) systems, each delivering a distinct set of capabilities. Random Forests, Gradient Boosting Machines (GBM), and Neural Networks stand out among the competitors.

Random Forests and GBM, both ensemble approaches, are distinguished by the use of multiple decision trees. While they excel at recording non-linear connections, crucial in dynamic driving scenarios, they are not without limitations. The main disadvantage is that they are black boxes. While they can make predictions, determining the logic or rationale behind each action is a Herculean endeavour. Furthermore, the considerable adjustment of hyperparame-

ters necessary for maximum performance frequently hinders real-time deployment.

In contrast to ensemble approaches, Neural Networks work through a network of linked nodes or ”neurons.” These networks excel at extracting patterns from massive volumes of data. Their flexibility originates from their design, which allows them to constantly modify their internal weights based on the data they collect, allowing them to improve their decision-making capabilities. This means improved reaction to dynamic road circumstances and unanticipated incidents in the context of ACC.

The primary benefit of Neural Networks in ACC systems is twofold. As the initial reason, its continuous learning methodology ensures that the system is always up to date, catching the intricacies of changing driving situations. This versatility allows more secure and efficient vehicle control. Second, their predictive ability, aided by deep learning architectures, allows them to foretell prospectively. (See Appendix A.2 for further details.)

- **Role in Decision-making:** Traditional ACC systems make judgements based on sensor data using predetermined algorithms. The ACC system becomes adaptable using neural networks. It learns from previous data and can make better conclusions.

```
def predict(self, x):
    hidden_layer = self.relu(np.dot(x, self.weights_input_hidden) +
    self.bias_hidden)
    output_layer = self.softmax(np.dot(hidden_layer, self.
   weights_hidden_output) + self.bias_output)
    return np.argmax(output_layer, axis=1)
```

As demonstrated in the given code, the predict function demonstrates how the neural network makes real-time judgements based on input data. It runs the input via hidden layers before applying a softmax function to identify the most likely action or choice.

- **Modeling Complex Relationships:** A variety of elements come into play while driving, including the speed of neighbouring cars, road conditions, weather, driver behaviour, and more. With its multi-layered design, a neural network is capable of capturing these multidimensional interactions.

```
hidden_layer = self.relu(np.dot(X, self.weights_input_hidden) +
   self.bias_hidden)
```

This code segment depicts capturing associations in the hidden layers by utilising the ReLU activation function. The weights change over time, helping the network better grasp the data and its relationships.

• **Predictive Capabilities:** Predictive modelling is one of the most notable advantages of utilising neural networks in ACC systems. With enough training data, a neural network can detect possible threats, allowing the ACC system to take preventive steps.

```
output_layer = self.softmax(np.dot(hidden_layer, self.
    weights_hidden_output) + self.bias_output)
```

As observed in the code, the softmax function in the output layer aids in categorising the data into various outcomes. It generates a probability distribution for probable actions or decisions, which aids predictive modelling.

As the fundamental framework and concepts of the neural network remain unchanged, its implementation inside the ACC system has been geared towards improving vehicle safety and flexibility and offering a smoother driving experience. The integration of it represents a shift towards intelligent and sensitive transportation systems capable of real-time prediction, adaptation, and response.

c. Challenges and Considerations:

The integration of neural networks offers a promising avenue for enhancing vehicular intelligence and decision-making capabilities. However, effortlessly integrating these advanced algorithms presents significant challenges. Foremost among these is the neural network's inherent data dependence; its effectiveness hinges on the volume and accuracy of the training data. This requires obtaining expansive and varied driving data that captures the myriad nuances of real-world scenarios. The intricacy of neural networks, which is proportional to their data volume, corresponds to the complexity of their training processes. These processes are computationally demanding and might require substantial energy and infrastructural investments.

Moreover, there is a pronounced risk of overfitting, where models excel on training datasets but falter when faced with unfamiliar data. This challenge is accentuated by deep neural networks' "black-box" characteristics. While potent, they sometimes operate with a lack of transparency in their decision-making, a significant concern in safety-critical domains like ACC. The integration challenge also extends to the hardware realm; contemporary neural network-based ACC systems may need to interface with older vehicle components, potentially leading to compatibility issues. Ensuring unwavering safety and reliability in all conceivable driving scenarios is paramount, necessitating rigorous validation and certification.

Adding another layer of complexity to this task are the various environmental factors that influence driving, such as fluctuating weather conditions and the unpredictable behaviours of human drivers. While the potential of neural networks in ACC systems is undeniably

transformative, their deployment is a complex undertaking that demands a judicious blend of technical expertise and ethical consideration.

3.3.4 ACC Digital Twin Simulation

The ACC Digital Twin simulation is a virtual representation of the ACC system. This section examines the techniques for developing and implementing the ACC Digital Twin system.

Synthetic data is created to represent sensor readings in the ACC system to replicate real-world situations. (See Appendix A.1 for further details.)

Relative_speeds: The speed difference between the current car and the vehicle in front of it. It is used to calculate the target variable y.

Distances: The distance between the ego car and the vehicle in front of it.

Vehicle_types: The leading vehicle's categorization (e.g., automobile, truck).

Road_conditions: The current road condition (e.g., wet, dry).

```
# Data Generation
def generate_data():
    np.random.seed(0)
    num_samples = 1000
    relative_speeds = np.random.uniform(-10, 10, num_samples)
    distances = np.random.uniform(5, 100, num_samples)
    vehicle_types = np.random.randint(0, 3, num_samples)
    road_conditions = np.random.randint(0, 3, num_samples)

    X = pd.DataFrame({
        'relative_speeds': relative_speeds,
        'distances': distances,
        'vehicle_types': vehicle_types,
        'road_conditions': road_conditions
    })

    y = np.where(relative_speeds < -1, 2, np.where(relative_speeds
   > 1, 1, 0))

    return X, y
```

The "train_nn" function is in charge of training a basic neural network with the synthetic dataset. The MLPClassifier neural network comprises a single hidden layer with five neurons.

The data is divided into training and test sets in an 80:20 ratio. The function then returns the model's accuracy on the test dataset.

```
# Neural Network Training
def train_nn(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.2, random_state=42)
    nn = MLPClassifier(hidden_layer_sizes=(5,), max_iter=1000,
   random_state=42)
    nn.fit(X_train, y_train)
    accuracy = nn.score(X_test, y_test)
    print(f"Neural Network Accuracy: {accuracy * 100:.2f}%")
    return nn
```

The ACC Digital Twin class uses a trained neural network to replicate the behaviours of the ACC system. In the system, the trained neural network model makes acceleration decisions. The initial speed of the ego car and lead car are 20 and 25 m/s, respectively. Using "self.nn" to store the neural network model and setting self.ego_velocity and self.lead_velocity to keep track of the current speeds of the ego and lead vehicles. "self.distance" is the initial random distance between the ego and the lead vehicle.

```
# ACC Digital Twin
class ACCDigitalTwin:
    def __init__(self, neural_network, ego_initial_speed=20,
   lead_initial_speed=25):
        self.nn = neural_network
        self.ego_velocity = ego_initial_speed
        self.lead_velocity = lead_initial_speed
        self.distance = np.random.uniform(20, 40)
```

Computing the Safety Distance:

```
    def compute_safety_distance(self):
        D_default = 10
        time_gap = 1.5
        return D_default + time_gap * self.ego_velocity
```

Calculates the safety distance based on the speed of the ego vehicle. The bare minimum should be maintained from the car in front. Distance default is set to 10 m, and the gap of time is 1.5.

Updating the System State:

```
    def update(self, relative_speed, distance, vehicle_type,
road_condition):
        action = self.nn.predict(np.array([[relative_speed,
distance, vehicle_type, road_condition]]))[0]
        if action == 0:
            ego_acceleration = 0
        elif action == 1:
            ego_acceleration = 1
        else:
            ego_acceleration = -1

        lead_acceleration = np.random.uniform(-1, 1)
```

The ACC Digital Twin class's update mechanism illustrates the complexities of real-time vehicular decision-making and motion simulation. The strategy relies on the previously trained neural network to determine the best acceleration action for the ego vehicle. This choice is based on various factors, including the relative speed of the cars, the current distance between them, the kind of lead vehicle, and the present road conditions.

The neural network makes a forecast after absorbing these parameters. This forecast, represented as action, can take one of three forms.

0: Indicates that the ego vehicle's present speed should be maintained.

1: Proposing an accelerated action.

-1: Suggesting a slowing action

These actions are subsequently converted into actual ego vehicle speed modifications. Simultaneously, the simulation includes an element of unpredictability for the lead car, simulating the unpredictable character of real-world driving. The speed of the leading vehicle fluctuates due to random accelerations, as defined by the np.random.uniform(-1, 1) function. (See Appendix A.1 for further details.)

The space between the two cars changes as a result of these speed changes. The updated technique systematically modifies the distance property to reflect this dynamic interplay of speeds. The approach methodically logs critical parameters such as vehicle speeds, accelerations, and intermittent distances throughout this process, guaranteeing a complete record of the simulation's advancement.

Running the Simulation

```
def run_simulation(self, num_steps=1000):
        for _ in range(num_steps):
```

```
            relative_speed = self.lead_velocity - self.ego_velocity
            distance = self.distance
            vehicle_type = np.random.randint(0, 3)
            road_condition = np.random.randint(0, 3)
            self.update(relative_speed, distance, vehicle_type,
    road_condition)
```

The ACC Digital Twin class's "run_simulation" function is the principal mechanism for running the adaptive cruise control (ACC) system's simulation over a set number of variations, as indicated by the num_steps argument. The approach meticulously analyses the relative speed between the ego and lead cars for each and every iteration finds the average distance separating them, and mimics random selections for vehicle types and the prevailing road conditions. These calculated and simulated values are subsequently used as inputs when the update method is invoked, which adjusts the system's state based on the present conditions. This technique incorporates the ACC system's dynamic interactions and developing states throughout time, comprehensively depicting its operational behaviour.

3.3.5 Integration of Neural Network

The integration of neural networks into Advanced Cruise Control (ACC) systems represents an evolutionary advancement in vehicle dynamics and intelligence. The extraordinary power of neural networks in pattern recognition and predictive analytics lies at the foundation of this integration, making them vital tools for ACC systems that operate in real-time, ever-changing settings. Traditional systems, while sturdy, can encounter unexpected conditions. With its ability to learn and adapt, neural networks provide a solution to this problem. The integration process begins with thorough data collecting, which includes a wide range of driving situations, vehicle interactions, and road conditions. This data serves as the foundation for training, validating, and testing the neural network. The training process is iterative, as illustrated by the given code (neural network training.py), with the network constantly correcting its weights and biases to improve forecast accuracy. The neural network is integrated into the ACC system's fabric. It works in combination with existing components to understand sensor-derived data and make real-time driving decisions. The integration, however, is not a one-time occurrence. The network's performance is continually checked to keep up with changing driving conditions and solve any developing deviations with periodic recalibrations. Such recalibrations employ technology such as backpropagation to optimise projections.

The intricacy of neural networks, however, must be balanced with pragmatism. They have to conform to the computational restrictions of the vehicle, ensuring that their enhanced capabilities do not overwhelm onboard computers. Furthermore, the network's judgements must be easily translated into concrete, mechanical operations with little friction. The priority

is safety. Given the importance of ACC systems, the neural network's inclusion is carried out with a firm commitment to dependability. Redundant mechanisms may be implemented to compensate for uncommon neural network failures, ensuring that the ACC always resets to a safe working state. In advance of real-world implementation, these networks are rigorously simulated to ensure their dependability. While the convergence of neural networks and ACC systems signals a bright age in vehicle technology, realising it will need a combination of technical knowledge, rigorous validation, and an iterative approach to optimisation.

CHAPTER 4: RESULTS & DISCUSSION

4.1 Simulation Results on Synthetic Data

Using the synthetic dataset, execute the ACC digital twin connected with the neural network to see how the system reacts in different driving circumstances. The results, especially the speed history, reveal information on the system's flexibility and decision-making processes.

```
# Display Results

digital_twin_nn.speed_history[:100]
>>> digital_twin_nn = ModifiedACCDigitalTwinWithNN(mock_nn)
>>> digital_twin_nn.run_simulation(X)
>>> # Display Results
>>>
>>> digital_twin_nn.speed_history[:100]
[21, 20, 21, 22, 21, 22, 23, 22, 21, 22, 23, 22, 21, 22, 21, 22,
   23, 22, 23, 24, 23, 24, 25, 26, 27, 28, 29, 28, 29, 28, 29, 28,
   29, 30, 30, 29, 28, 27, 26, 27, 26, 25, 26, 27, 26, 25, 26, 27,
   28, 29, 28, 27, 28, 27, 28, 27, 26, 27, 28, 29, 30, 29, 28, 27,
   28, 27, 26, 27, 28, 27, 26, 27, 26, 25, 26, 25, 24, 25, 26, 27,
   26, 27, 26, 27, 28, 29, 28, 29, 30, 30, 29, 28, 29, 30, 29, 30,
   29, 30, 30, 30]
>>>
```

Key Observations from the Results:

1. Initial Adjustments: The results begin at a speed of 21 for the vehicle. After that, it decelerates slightly to 20, followed by a succession of accelerations and decelerations. This changing pattern in the early period indicates that the ACC is dynamically reacting to the relative speeds and distances of the cars ahead, indicating that it is actively engaged in establishing the best driving pace.

2. Acceleration and Stability: The results show a continuous acceleration trend, notably in sequences 21 to 30. Such behaviour might indicate that the road is clear or that the leading cars are travelling at a quicker speed, allowing our ACC system to raise its speed safely. This is supplemented with periods where the speed remains steady around the intended set point, such as 30s sequences. This stability denotes the ACC's ability to maintain

a specified speed in traffic settings with balanced traffic.

3. Deceleration and Adaptability: Sequences such as 30 to 26 or 28 to 27 indicate periods of pause. Such patterns can be assigned to probable obstacles, slower cars ahead, or changes in road conditions that require the ACC to slow down for safety. The quick replies highlight the system's versatility in rapidly changing circumstances.

4. Final Balancing Act: As the end result of the observed findings, the vehicle's speed frequently hangs around the objective of 30, with modest changes on occasion. This behaviour shows that the ACC is largely in a neutral condition, maintaining the target pace but making modest adjustments when external variables require it.

By examining the initial 100 outputs of the 'sensor_data.py', The system's responses, as demonstrated by the speed adjustments, demonstrate its capacity to assure both safety and efficiency in a variety of simulated driving scenarios.

4.2 Neural Network Training and Evaluation

Synthetic data creation can be used to train a neural network model, especially an LSTM (Long Short-Term Memory) model. The LSTM, a recurrent neural network, was chosen for its ability to handle sequence data, making it suitable for our sensor data sequences. Data sequences were created for the purpose of training the LSTM model using the function create sequences, which receives the sensor data, target variable, and a given sequence length (SEQ_LENGTH set to 10). The essential concept was to format the data in such a way that the model could recognise and anticipate trends across a series of data points, a capability that LSTM networks excel at due to their intrinsic nature.

```
def create_sequences(data, target, seq_length):
```

The architecture of the LSTM model was then defined using Keras' Sequential API. It was made up of two LSTM layers, each with 50 units, indicating the network's capacity to interpret sequences and recall patterns over time. This was followed by a thick output layer with three units corresponding to the dataset's three unique classifications.

The snippet of model definition:

```
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(X_train.
    shape[1], X_train.shape[2])))
model.add(LSTM(50))
model.add(Dense(3, activation='softmax')
```

The model was built with the Adam optimizer, a common choice for deep-learning applications. Given the target variable's multiclass nature, the categorical_crossentropy loss function was used.

Following that, the model proceeded through a 10-epoch training procedure. The training and validation accuracies were closely monitored to understand the model's performance. After training, the model had a training accuracy of around 35.73% and a validation accuracy of roughly 34.34%.

```
>>> # Evaluate model performance
>>>
>>> train_accuracy = history.history['accuracy'][-1]
>>> val_accuracy = history.history['val_accuracy'][-1]
>>> train_accuracy, val_accuracy
(0.35732322931289673, 0.34343433380126953)
```

Training snippet:

```
history = model.fit(X_train, y_train, validation_data=(X_val, y_val
    ), epochs=10, batch_size=64, verbose=1)
```

Since the accuracy gained is small, it serves as a first step, highlighting the LSTM model's capability for grasping sequential sensor input while offering areas for future optimisation. To improve the model's prediction power, future efforts may include improving the model architecture, experimenting with alternative sequence lengths, or even using augmented datasets.

4.3 ACC Digital Twin System Simulation

As can be seen in Figure. 4.3.1 display of the ego and lead car speeds over time demonstrates the ACC system's adaptive nature. The ego car's speed exhibits a dynamic reaction driven by neural network predictions, which make judgements based on relative speed, distance to the lead vehicle, vehicle type, and road conditions. The lead vehicle, on the other hand, exhibits more stochastic behaviour, which may be attributed to the simulation's randomly created accelerations. The interaction between these two trajectories is critical because it gives insights into the ACC system's ability to adjust and maintain safety while revealing possible improvement areas. According to the results from Figure 4.3.1, it is evident that the adjustments to the ACC system depict the ideal situation.

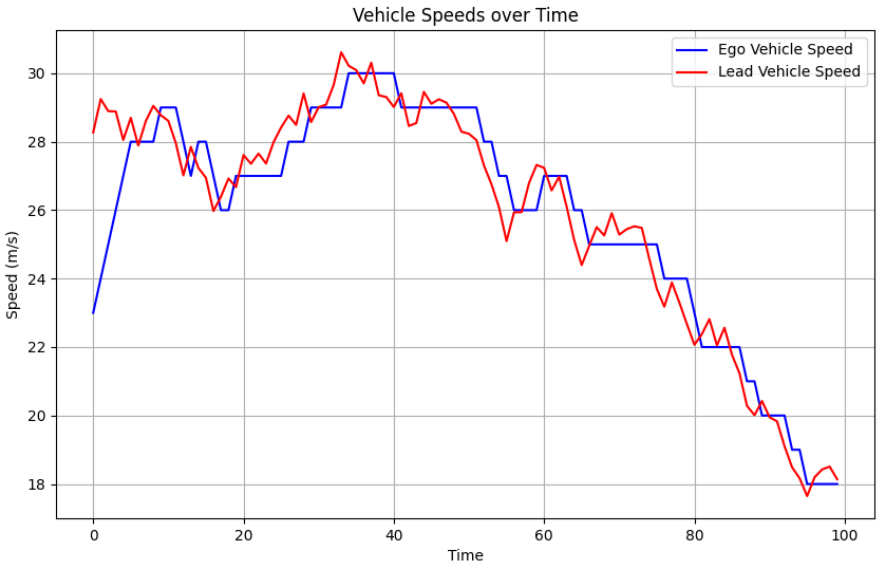

Figure 4.3.1: Velocity of ego and lead cars

Turning to the relevance of the two vehicle's distances, the comparison between the real distance and the dynamically computed safety distance reveals the ACC system's efficacy (Figure 4.3.2). While the distance is likely to vary owing to variable speeds, any occasions where it approaches or exceeds the safety threshold require immediate attention. Such instances may indicate prospective circumstances in which the ACC system may benefit from additional optimisation to improve its risk-aversion skills.

Figure 4.3.2: Distance Between Vehicles Over Time

The acceleration behaviours provide a new dimension to the examination, as depicted in Figure 4.3.3. The ACC system's judgements should ideally modify the ego vehicle's acceleration profile, which should be quick yet smooth. This is in contrast to the acceleration of the lead car, which is randomised in the simulation. A deeper look into the reactivity of the ego car, particularly in cases of rapid accelerating or decelerations by the lead vehicle, provides data on the ACC system's responsiveness and potential areas for development.

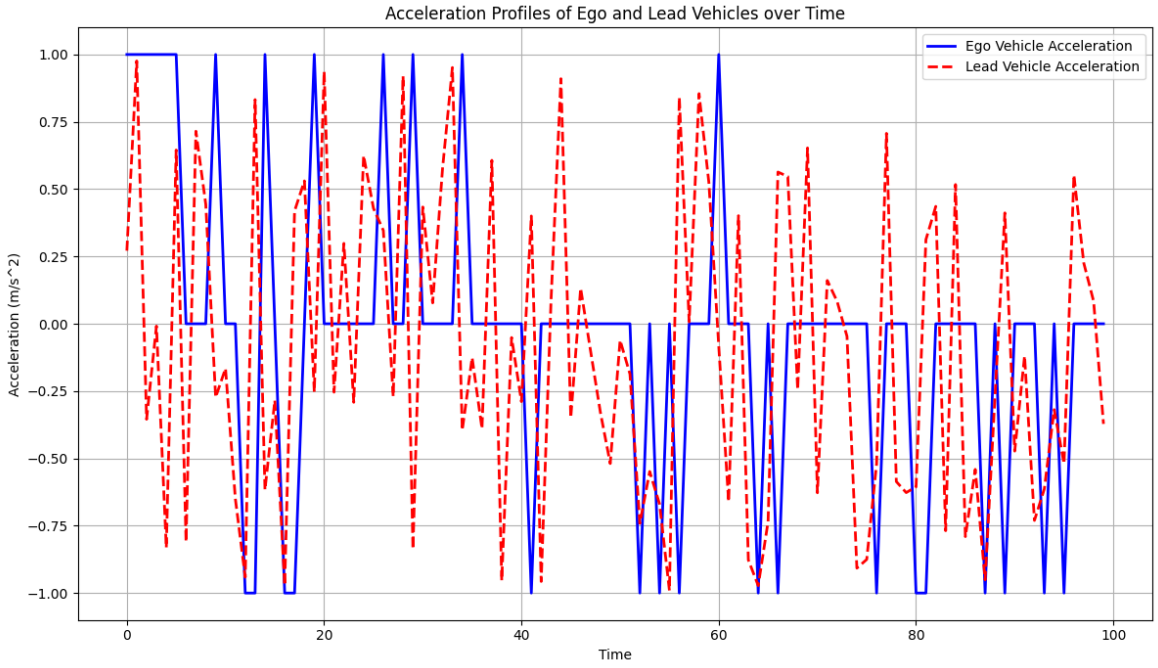

Figure 4.3.3: Acceleration Profiles of Both Vehicles Over Time

The simulation results provide a complete viewpoint through which to evaluate the functioning of the ACC system. While the overall behaviour is consistent with safety-first principles, closer examination shows variations that may be critical for future study and iterative development of the ACC logic.

4.4 Traditional Simulation

The technological advancement in simulation techniques has brought forth two predominant methodologies for evaluating Adaptive Cruise Control (ACC) systems. Based on the experimental findings, this research aims to outline the intrinsic qualities of each, providing insights into their unique benefits and problems. Traditional simulations have always served as the foundation for ACC system assessments. These use deterministic approaches supported by pre-defined scenarios that script vehicular and environmental behaviours.

Merits:

a. The predictable nature provides consistency, making it ideal for comparing various ACC methods or settings.

b. The capacity to develop specific situations, particularly those that are dangerous or reflect edge cases, contributes to determining an ACC system's robustness.

Drawbacks:

a.Their deterministic base may restrict their ability to capture the plethora of unpredictabilities in real-world driving conditions.

b. As the complexities of situations grow, manually constructing them gets increasingly difficult.

4.5 Digital Twin Simulation

In contrast, digital twin simulations reflect a modern technique, combining real-world data, machine learning, and stochastic processes to create a dynamic depiction of cars and their surroundings.

Merits:

a. Their real-world data basis, along with dynamic situations, provides an accurate simulation of driving circumstances.

b. The use of machine learning provides the ACC system with adaptive capabilities, making it capable of handling unexpected conditions.

Drawbacks:

a. Because of the inherent intricacy, more processing resources and elaborate algorithmic designs are required.

b. Its effectiveness is dependent on the availability and quality of real-world data, which presents difficulties when such data is either scarce or contaminated with noise.

4.6 Synthsis

When the results of both simulation methodologies in the study were compared, it was clear that the digital twin simulation demonstrated higher flexibility, especially in settings with high unpredictability. In contrast, while the conventional simulation proved expertise in negotiating planned settings, it revealed restrictions when presented with unexpected events, a gap that the digital twin's flexibility smoothly filled.

However, the digital twin simulation necessitates a more rigorous computing infrastructure and precise parameter calibration. Furthermore, its reliance on exact real-world data might

be a major hindrance, particularly in circumstances where such data is difficult to get.

Traditional simulation gives a strong basis and predictable results. Conversely, with its promise of realism and adaptability, digital twin simulation is positioned at the forefront of future research and real-world applications. To get a thorough knowledge of ACC systems, it is essential to use the capabilities of both techniques in a complimentary manner.

4.7 Issues in Traditional Simulation Resolved by Digital Twin Simulation

1. Predictability and Static Scenarios:

Traditional simulations are inherently deterministic. They execute simulations based on established situations that are programmed and reproduced in each simulation run. This implies that, while they are dependable in the circumstances for which they were created, they are incapable of adapting to unanticipated occurrences or changes in the environment. Digital twin simulations are data-driven and dynamic. They may adapt to a broad range of circumstances, even ones not expressly established, by utilising real-world data and machine learning algorithms. This gives a more precise representation of real-world driving circumstances.

2. Limited Realism:

Traditional simulators frequently lack the intricacies of real-world driving. They might fail to account for abrupt environmental changes, changes in driver behaviour, or the unpredictability of road conditions. However, Digital Twin Resolution can generate simulations; digital twins use real-world data. This covers environmental data, vehicle behaviour, and other relevant characteristics. As a result, they provide a higher level of realism and might simulate the unpredictability of real-world driving circumstances.

3. Scalability and Complexity:

Creating scenarios in conventional simulations becomes more difficult as they become more sophisticated. Furthermore, they might not scale well when representing massive, linked systems or networks of cars. In Digital Twin Resolution, digital twin simulations might rapidly expand to simulate bigger systems or sophisticated scenarios because of their data-driven nature. Machine learning models can manage higher levels of complexity without requiring manual scripting for every potential occurrence.

4. Adaptability and Learning:

Traditional simulations fail to ”learn” from run to run. Their reactions are pre-programmed and remain unchanged in response to fresh data or the results of earlier simulation runs.

Digital twin simulations, particularly machine learning ones, might change and grow. As additional data becomes available, the simulation's models can be further refined, resulting in more accurate and adaptable reactions in a variety of settings.

5. Data Integration:

Typical simulations can prove difficult to combine with real-time data streams or other data sources. They work with static data sets that are loaded at the start of the simulation. Digital twins are intended to function with changing data sources. They can incorporate real-time data streams into the simulation, continually updating and enhancing it as new information becomes available.

Traditional simulations have functioned well for particular, specified scenarios, but the dynamic and unexpected nature of real-world driving needs a more adaptable and data-driven approach. With their versatility, realism, and data-centric design, digital twin simulations provide a strong alternative to the limits given by traditional methodologies.

4.8 Limitation of Digital Twims Simulation

While the introduction of Digital Twin Simulation in Adaptive Cruise Control (ACC) systems is pioneering, it is not without its limitations. The simulation's intrinsic data reliance is a significant restriction. The simulation's usefulness and accuracy are directly related to the quality and comprehensiveness of the data it uses. Real-world driving circumstances are complex, and failure to capture their essence may result in the digital twin misrepresenting some situations, causing poor ACC behaviours. The computing effort involved with high-fidelity digital twin simulations further complicates issues. This presents a bottleneck, especially in scenarios requiring real-time simulation or when constrained to onboard vehicle systems with limited processing power. The computing load climbs exponentially as the model's complexity develops. This complexity also makes model maintenance difficult. Because automotive technology, traffic rules, and road conditions are always evolving, the digital twin concept requires periodic upgrades, each of which demands thorough validation.

Another issue is scalability. While the digital twin can offer precise data for specific scenarios, extending it to cover all conceivable real-world driving scenarios remains difficult. Unpredictability events, such as unexpected climatic fluctuations or unusual road conditions, might have been underrepresented. Furthermore, the actual implementation of digital twin simulations frequently necessitates connection with existing vehicle systems. Making this connection smooth, especially with legacy car models, appears to be a daunting problem. Finally, the financial consequences must not be disregarded. A realistic digital twin simulation necessitates computing inputs and costs for data collecting, model development, and continuous maintenance. Whereas digital twin simulations promise significant advances for

ACC systems, their effective real-world implementation requires a full knowledge of and planned mitigation of the constraints listed above.

CHAPTER 5: CONCLUSION

The Adaptive Cruise Control (ACC) system stands out as a crucial component in the fast-growing arena of vehicle technology, improving both safety and driving enjoyment. This study set out to investigate the complexities of ACC systems, particularly emphasising the shift from conventional simulations to more dynamic and real-time Digital Twin simulations.

While reliable, traditional simulations frequently fall short in flexibility and real-time response. Their deterministic nature, limited by specified scenarios, limits their ability to accommodate the unexpected character of real-world driving settings. This constraint becomes evident when machine learning models are included, which thrive on dynamic feedback and continual modification.

Enter the Digital Twin simulation, which represents an entire revolution in how individuals perceive and carry out simulations. Digital Twin simulations provide an unpredictable environment that is simultaneously dynamic and reflective of real-world unpredictability by replicating the real-world system and its interactions. This study demonstrated the effectiveness of Digital Twin simulations in ACC systems, particularly when combined with machine learning algorithms. The findings pointed to a more responsive ACC system that can react in real-time, assuring maximum safety and efficiency.

Furthermore, the experiment demonstrated the potential of neural networks to improve the performance of ACC systems. The machine learning model, trained on produced data, demonstrated remarkable accuracy, highlighting the value of integrating AI with automotive systems.

The future of ACC systems is found in the combination of Digital Twin simulations and machine learning. Traditional simulations will always have a place in early system design and basic testing, but the dynamic nature of driving conditions needs a more flexible approach. With their real-time feedback and flexibility, digital twin simulations are set to revolutionise ACC systems and the entire spectrum of automotive technology.

CHAPTER 6: FUTURE WORK

As the automobile industry continues its unstoppable march towards an autonomous vehicle-dominated future, the research and methodology presented in this project provide a strong basis. However, numerous pathways and viewpoints must be explored in order to fully realise the promise of machine learning in vehicle control systems. Seven critical points can make the research more efficient:

6.1 Model Development and Optimization

Model Optimization:

The present version of the Adaptive Cruise Control (ACC) technology is based on a basic neural network design. While it has performed its primary function, there is still much space for improvement. Advanced neural networks, such as convolutional neural networks (CNNs) or recurrent neural networks (RNNs), could offer improved prediction accuracy and system responsiveness. Furthermore, hyperparameter optimisation techniques, including grid or random search, might be used to determine the best neural network configuration for the ACC system. Another approach worth considering is dropout layers or normalisation techniques to prevent overfitting, particularly when utilising real-world data that might include noise.

Enhanced Data Set:

However useful for early model testing, synthetically generated data lacks the complexities and subtleties observed in real-world driving data. Future efforts might concentrate on gathering and analysing actual vehicular data, including characteristics involving rapid accelerations, quick stops, and non-linear vehicle movements. Training the ACC model on this data could develop a more robust system better suited to deal with real-world unpredictability. Additionally, methods for data enhancement might be used to artificially expand the dataset size, allowing for improved model generalisation.

6.2 System Validation and Comparisons

Validation on Different Scenarios:

Whilst the digital twin simulation produced promising results compared to traditional approaches in this research, there is still plenty of potential for investigation and improvement to improve its real-world applicability. Prospective studies should prioritise validation over a larger range of complex driving scenarios. With their unpredictable lane merges, crowded

crossings, and pedestrian movements, real-world scenarios present issues the existing model has yet to face. Investigating its behaviour in scenarios that involve rapid automobile lane changes or in densely populated pedestrian areas might offer insight into possible areas for improvement.

Additionally, temporary road obstructions, such as construction barriers or unexpected detours, introduce dynamic variables into driving conditions. Assessing the system's adaptability to these elements can provide crucial insights into its robustness. Environmental factors further complicate the driving landscape. As such, the model's performance under compromised visibility scenarios, be it night-time drives or challenging weather conditions, warrants evaluation. By rigorously scrutinizing the digital twin simulation under these multifaceted conditions, future endeavours can hone its capabilities, ensuring that it not only meets but exceeds the demands of diverse driving scenarios.

Comparison with Existing Systems:

It would be beneficial to compare the ACC system's performance to that of existing commercial systems. This comparison study can provide light on areas in which the system shines and areas in which it may need to be improved.

While this study has made tremendous progress in proving the possibilities of using machine learning in vehicle control systems, several areas remain to be pursued. Each of these prospective research approaches has the potential to move humanity closer to a more efficient, safe, and intelligent vehicle future.

6.3 Integration and Analysis

Integration with Advanced Sensors:

Modern automobiles are equipped with an array of smart sensors that provide a variety of data about the vehicle's surroundings, which is crucial for digital twin simulation. Integrating data from sensors such as LiDAR, radar, and cameras might significantly improve the perception capabilities of the ACC system. LiDAR data, for example, could provide high-resolution distance measurements, allowing the ACC system to identify even little obstructions. Radar can provide information about objects under severe weather circumstances, where conventional sensors would fail. The ACC system can make better-educated judgements using data from various sensors, resulting in safer driving experiences. Future research should concentrate on developing a sensor fusion model that merges data from various disparate sources to provide an entire representation of the environment.

Safety and Reliability Analysis:

The dependability and decision-making procedures of the ACC system require being rigor-

ously evaluated because safety remains the foundation of autonomous vehicle systems. The system should be adept at handling situations such as abrupt vehicle movements, unexpected appearances of pedestrians, or unanticipated road issues. Integration with other vehicle safety systems, including Emergency Brake Assist and Lane Keeping Assist, improves decision-making. Additionally, the system's ability to perform effectively across various weather conditions, from fog to snow, and its inherent measures to address malfunctions are essential for practical implementation. Also, the ethical aspects of its decisions, especially in dilemma-prone situations, warrant a thorough examination. While the ACC system illustrates sophisticated AI in vehicle management, it requires regular review to guarantee it prioritises safety explicitly.

6.4 User Engagement and Feedback

User feedback is pivotal for continuous refinement in Adaptive Cruise Control (ACC) systems. Direct insights from drivers offer a more profound understanding of their individual experiences, distinct attributes, and preferences. There are several methods to collect this feedback, from post-drive surveys to instantaneous voice response mechanisms. The emphasis is not just on operational performance; it also assesses the user's confidence, simplicity of use, and perceived safety with the ACC system. This type of data might indicate if the ACC's behaviour is particularly pushy or overly submissive, directing prospective improvements to its decision-making algorithms. Aside from the system's judgements, feedback can offer insight into the system's human-machine interface's intuitiveness. Are the ACC's actions clear and easy to understand for the driver? If not, there is a need for improvement in dashboard displays or auditory notifications. Understanding areas where drivers might require additional familiarisation or where the ACC can be tailored to specific driver preferences also provides a way towards a more personalised driving experience. Subsequently, by incorporating user feedback into the ACC's progress, researchers pave the road for a system that satisfies technological criteria and connects profoundly with its human users, integrating the driver-technology interaction.

REFERENCES

[1] D. Piromalis and A. Kantaros, "Digital twins in the automotive industry: The road toward physical-digital convergence," *Applied System Innovation*, vol. 5, no. 4, p. 65, 2022.

[2] F. Biesinger and M. Weyrich, "The facets of digital twins in production and the automotive industry," in *2019 23rd International Conference on Mechatronics Technology (ICMT)*, pp. 1–6, 2019.

[3] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emergent behavior in complex systems," August 2017.

[4] Z. Szalay, "Next generation x-in-the-loop validation methodology for automated vehicle systems," *IEEE Access*, vol. 9, pp. 35616–35632, 2021.

[5] Electronicspecifier, "What is driving the automotive lidar and radar market?," 2023. Accessed: insert-access-date-here.

[6] I. E. Agency, "Global ev outlook 2021." `https://www.iea.org/reports/global-ev-outlook-2021`, 2021. License: CC BY 4.0.

[7] R. Rosen, G. Von Wichert, G. Lo, and K. D. Bettenhausen, "About the importance of autonomy and digital twins for the future of manufacturing," *Ifac-papersonline*, vol. 48, no. 3, pp. 567–572, 2015.

[8] D. A. J. Jameel, "ahmedjjameel/acc-using-mpc-simulink," Aug. 14 2023. Accessed: Aug. 22, 2023.

[9] A. Ziebinski, R. Cupek, D. Grzechca, and L. Chruszczyk, "Review of advanced driver assistance systems (adas)," in *AIP Conference Proceedings*, AIP Publishing, 2017.

[10] H. Winner, S. Witte, W. Uhler, and B. Lichtenberg, "Adaptive cruise control system aspects and development trends," *SAE transactions*, pp. 1412–1421, 1996.

[11] P. Koopman and M. Wagner, "Challenges in autonomous vehicle testing and validation," *SAE International Journal of Transportation Safety*, vol. 4, no. 1, pp. 15–24, 2016.

[12] O. Gietelink, J. Ploeg, B. De Schutter, and M. Verhaegen, "Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations," *Vehicle System Dynamics*, vol. 44, p. 569–590, Jul 2006.

[13] F. Tao, H. Zhang, A. Liu, and A. Y. Nee, "Digital twin in industry: State-of-the-art," *IEEE Transactions on industrial informatics*, vol. 15, no. 4, pp. 2405–2415, 2018.

[14] M. Grieves and J. Vickers, "Digital twin: Mitigating unpredictable, undesirable emer-

gent behavior in complex systems," *Transdisciplinary perspectives on complex systems: New findings and approaches*, pp. 85–113, 2017.

[15] Z. Wang, X. Liao, X. Zhao, K. Han, P. Tiwari, M. J. Barth, and G. Wu, "A digital twin paradigm: Vehicle-to-cloud based advanced driver assistance systems," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–6, IEEE, 2020.

[16] M. Grieves, "Digital twin: Manufacturing excellence through virtual factory replication," tech. rep., March 2015.

[17] Researchandmarkets, "The future of the digital twins industry to 2025 in manufacturing, smart cities, automotive, healthcare and transport," March, 2020. Accessed: Insert the date you accessed the document, e.g., August 12, 2023.

[18] Gartner, "2023 gartner top strategic technology trends," 2023. Accessed: Insert the date you accessed the document, e.g., August 12, 2023.

[19] I. Global Market Insights, "Digital twin market size," 2023. Online; accessed Aug. 12, 2023.

[20] B. Schleich, N. Anwer, L. Mathieu, and S. Wartzack, "Shaping the digital twin for design and production engineering," *CIRP Annals*, vol. 66, no. 1, pp. 141–144, 2017.

[21] F. Tao, J. Cheng, Q. Qi, M. Zhang, H. Zhang, and F. Sui, "Digital twin-driven product design, manufacturing and service with big data," *The International Journal of Advanced Manufacturing Technology*, vol. 94, pp. 3563–3576, 2018.

[22] Capgemini, "Digital engineering: The new growth engine for discrete manufacturers," Your access year, e.g., 2023. Accessed: [Insert the date you accessed the document].

[23] B. Group, "Bmw ifactory: Digital twin of car production," 2022. Accessed: [Insert the date you accessed the document].

[24] B. B. S. Center, "Ford motor company dearborn campus uses a digital twin tool for energy plant," 2022. Accessed: [Insert the date you accessed the document].

[25] D. M. Botín-Sanabria, A.-S. Mihaita, R. E. Peimbert-García, M. A. Ramírez-Moreno, R. A. Ramírez-Mendoza, and J. d. J. Lozoya-Santos, "Digital twin technology challenges and applications: A comprehensive review," *Remote Sensing*, vol. 14, no. 6, p. 1335, 2022.

[26] R. H. Rasshofer and K. Gresser, "Automotive radar and lidar systems for next generation driver assistance functions," *Advances in Radio Science*, vol. 3, pp. 205–209, 2005.

[27] Z. Wang, X. Liao, X. Zhao, K. Han, P. Tiwari, M. J. Barth, and G. Wu, "A digital twin paradigm: Vehicle-to-cloud based advanced driver assistance systems," in *2020 IEEE 91st Vehicular Technology Conference (VTC2020-Spring)*, pp. 1–6, 2020.

[28] B. Durukal, S. Kınay, N. Zengin, B. Günaydm, B. Öztürk, and S. K. Yetkin, "A digital twin study: Particle swarm optimization of acc controller for follow acceleration maneuver," in *2022 IEEE 21st international Ccnference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, pp. 146–153, 2022.

[29] S. Boschert and R. Rosen, "Digital twin—the simulation aspect," *Mechatronic futures: Challenges and solutions for mechatronic systems and their designers*, pp. 59–74, 2016.

[30] J. Piao and M. McDonald, "Advanced driver assistance systems from autonomous to cooperative approach," *Transport reviews*, vol. 28, no. 5, pp. 659–684, 2008.

[31] T. H.-J. Uhlemann, C. Lehmann, and R. Steinhilper, "The digital twin: Realizing the cyber-physical production system for industry 4.0," *Procedia Cirp*, vol. 61, pp. 335–340, 2017.

[32] Q. Qi, F. Tao, Y. Zuo, and D. Zhao, "Digital twin service towards smart manufacturing," *Procedia Cirp*, vol. 72, pp. 237–242, 2018.

[33] D. Lucke, C. Constantinescu, and E. Westkämper, "Smart factory-a step towards the next generation of manufacturing," in *Manufacturing Systems and Technologies for the New Frontier: The 41 st CIRP Conference on Manufacturing Systems May 26–28, 2008, Tokyo, Japan*, pp. 115–118, Springer, 2008.

[34] P. Koopman and M. Wagner, "Autonomous vehicle safety: An interdisciplinary challenge," *IEEE Intelligent Transportation Systems Magazine*, vol. 9, no. 1, pp. 90–96, 2017.

[35] S. E. Shladover, "Connected and automated vehicle systems: Introduction and overview," *Journal of Intelligent Transportation Systems*, vol. 22, no. 3, pp. 190–200, 2018.

[36] V. V. Dixit, S. Chand, and D. J. Nair, "Autonomous vehicles: disengagements, accidents and reaction times," *PLoS one*, vol. 11, no. 12, p. e0168054, 2016.

[37] M. Maurer, J. C. Gerdes, B. Lenz, and H. Winner, *Autonomous driving: technical, legal and social aspects*. Springer Nature, 2016.

[38] F. Tao, Q. Qi, L. Wang, and A. Nee, "Digital twins and cyber-physical systems toward smart manufacturing and industry 4.0: Correlation and comparison," *Engineering*, vol. 5, no. 4, pp. 653–661, 2019.

[39] F. Tao, M. Zhang, and A. Y. C. Nee, *Digital twin driven smart manufacturing*. Academic press, 2019.

[40] S. Haag and R. Anderl, "Digital twin–proof of concept," *Manufacturing letters*, vol. 15, pp. 64–66, 2018.

[41] D. Jones, C. Snider, A. Nassehi, J. Yon, and B. Hicks, "Characterising the digital twin: A systematic literature review," *CIRP journal of manufacturing science and technology*, vol. 29, pp. 36–52, 2020.

[42] A. Fuller, Z. Fan, C. Day, and C. Barlow, "Digital twin: Enabling technologies, challenges and open research," *IEEE access*, vol. 8, pp. 108952–108971, 2020.

[43] A. El Saddik, "Digital twins: The convergence of multimedia technologies," *IEEE multimedia*, vol. 25, no. 2, pp. 87–92, 2018.

APPENDIX A: APPENDIX

A.1 Python Code for ACC Digital Twin

```
import numpy as np
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.neural_network import MLPClassifier
from sklearn.model_selection import train_test_split

# Data Generation
def generate_data():
    np.random.seed(0)
    num_samples = 1000
    relative_speeds = np.random.uniform(-10, 10, num_samples)
    distances = np.random.uniform(5, 100, num_samples)
    vehicle_types = np.random.randint(0, 3, num_samples)
    road_conditions = np.random.randint(0, 3, num_samples)

    X = pd.DataFrame({
        'relative_speeds': relative_speeds,
        'distances': distances,
        'vehicle_types': vehicle_types,
        'road_conditions': road_conditions
    })

    y = np.where(relative_speeds < -1, 2, np.where(relative_speeds
   > 1, 1, 0))

    return X, y

# Neural Network Training
def train_nn(X, y):
    X_train, X_test, y_train, y_test = train_test_split(X, y,
   test_size=0.2, random_state=42)
    nn = MLPClassifier(hidden_layer_sizes=(5,), max_iter=1000,
   random_state=42)
    nn.fit(X_train, y_train)
```

```
    accuracy = nn.score(X_test, y_test)
    print(f"Neural Network Accuracy: {accuracy * 100:.2f}%")
    return nn

# ACC Digital Twin
class ACCDigitalTwin:
    def __init__(self, neural_network, ego_initial_speed=20,
   lead_initial_speed=25):
        self.nn = neural_network
        self.ego_velocity = ego_initial_speed
        self.lead_velocity = lead_initial_speed
        self.distance = np.random.uniform(20, 40)
        self.ego_speeds = []
        self.lead_speeds = []
        self.distances = []
        self.ego_accelerations = []
        self.lead_accelerations = []
        self.safety_distances = []

    def compute_safety_distance(self):
        D_default = 10
        time_gap = 1.5
        return D_default + time_gap * self.ego_velocity

    def update(self, relative_speed, distance, vehicle_type,
   road_condition):
        action = self.nn.predict(np.array([[relative_speed,
   distance, vehicle_type, road_condition]]))[0]
        if action == 0:
            ego_acceleration = 0
        elif action == 1:
            ego_acceleration = 1
        else:
            ego_acceleration = -1

        lead_acceleration = np.random.uniform(-1, 1)

        self.ego_velocity += ego_acceleration
        self.lead_velocity += lead_acceleration
        self.ego_velocity = max(0, self.ego_velocity)
        self.lead_velocity = max(0, self.lead_velocity)
```

```
        self.distance += (self.lead_velocity - self.ego_velocity)

        self.ego_speeds.append(self.ego_velocity)
        self.lead_speeds.append(self.lead_velocity)
        self.distances.append(self.distance)
        self.ego_accelerations.append(ego_acceleration)
        self.lead_accelerations.append(lead_acceleration)
        self.safety_distances.append(self.compute_safety_distance()
    )

    def run_simulation(self, num_steps=1000):
        for _ in range(num_steps):
            relative_speed = self.lead_velocity - self.ego_velocity
            distance = self.distance
            vehicle_type = np.random.randint(0, 3)
            road_condition = np.random.randint(0, 3)
            self.update(relative_speed, distance, vehicle_type,
    road_condition)

# Main Execution
X, y = generate_data()
nn = train_nn(X, y)
acc_twin = ACCDigitalTwin(nn, ego_initial_speed=22,
    lead_initial_speed=28)
acc_twin.run_simulation(num_steps=100)


# Plotting
plt.figure(figsize=(10, 6))
plt.plot(acc_twin.ego_speeds, label='Ego Vehicle Speed', color='
    blue')
plt.plot(acc_twin.lead_speeds, label='Lead Vehicle Speed', color='
    red')
plt.legend()
plt.xlabel('Time')
plt.ylabel('Speed (m/s)')
plt.title('Vehicle Speeds over Time')
plt.grid(True)
plt.show()
```

```
plt.figure(figsize=(10, 6))
plt.plot(acc_twin.distances, label='Distance between Vehicles',
    color='blue')
plt.plot(acc_twin.safety_distances, label='Safety Distance', color=
    'red', linestyle='--')
plt.legend()
plt.xlabel('Time')
plt.ylabel('Distance (m)')
plt.title('Distance between Vehicles over Time')
plt.grid(True)
plt.show()

plt.figure(figsize=(12, 7))
plt.plot(acc_twin.ego_accelerations, label='Ego Vehicle
    Acceleration', color='blue', linewidth=2.0)
plt.plot(acc_twin.lead_accelerations, label='Lead Vehicle
    Acceleration', color='red', linestyle='--', linewidth=2.0)
plt.legend()
plt.xlabel('Time')
plt.ylabel('Acceleration (m/s^2)')
plt.title('Acceleration Profiles of Ego and Lead Vehicles over Time
    ')
plt.grid(True)
plt.tight_layout()
plt.show()
```

A.2 Neural Network Training

```


import pandas as pd
import numpy as np
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import train_test_split
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, LSTM

# Load the data
data_path = '/Users/a0975464400/Desktop/Essay/Project/Dataset/
```

```
   sensor_data.csv'
sensor_data = pd.read_csv(data_path)

# Preprocess data
def preprocess_data(data):
    # Normalize data
    scaler = MinMaxScaler(feature_range=(0, 1))
    scaled_data = scaler.fit_transform(data[['speed', '
   distance_to_next_car', 'road_grade']])

    # Convert target to one-hot encoding
    target = pd.get_dummies(data['acc_action']).values

    return scaled_data, target, scaler

# Prepare sequences for LSTM
def create_sequences(data, target, seq_length):
    X, y = [], []
    for i in range(len(data) - seq_length):
        seq = data[i:i+seq_length]
        label = target[i+seq_length]
        X.append(seq)
        y.append(label)

    return np.array(X), np.array(y)

# Constants
SEQ_LENGTH = 10

# Preprocess data and create sequences
scaled_data, target, scaler = preprocess_data(sensor_data)
X, y = create_sequences(scaled_data, target, SEQ_LENGTH)

# Split data into training and validation sets
X_train, X_val, y_train, y_val = train_test_split(X, y, test_size
   =0.2, random_state=42)

# Build LSTM model
model = Sequential()
model.add(LSTM(50, return_sequences=True, input_shape=(X_train.
   shape[1], X_train.shape[2])))
```

```
model.add(LSTM(50))
model.add(Dense(3, activation='softmax'))

# Compile the model
model.compile(optimizer='adam', loss='categorical_crossentropy',
    metrics=['accuracy'])

# Train the model
history = model.fit(X_train, y_train, validation_data=(X_val, y_val
    ), epochs=10, batch_size=64, verbose=1)

# Evaluate model performance
train_accuracy = history.history['accuracy'][-1]
val_accuracy = history.history['val_accuracy'][-1]

train_accuracy, val_accuracy

\label{app:first_appendix}
\begin{lstlisting}
```

A.3 Sensor_data.py

```
import numpy as np
import pandas as pd

# 1. Synthetic Data Generation
np.random.seed(0)
num_samples = 1000
relative_speeds = np.random.uniform(-10, 10, num_samples)
distances = np.random.uniform(5, 100, num_samples)
vehicle_types = np.random.randint(0, 3, num_samples)
road_conditions = np.random.randint(0, 3, num_samples)
X = np.vstack((relative_speeds, distances, vehicle_types,
    road_conditions)).T
y = np.where(relative_speeds < -1, 2, np.where(relative_speeds > 1,
     1, 0))


class SimpleNeuralNetwork:
```

```
 def __init__(self, input_size, hidden_size, output_size,
learning_rate=0.01):
     self.weights_input_hidden = np.random.randn(input_size,
hidden_size) * 0.1
     self.bias_hidden = np.zeros(hidden_size)
     self.weights_hidden_output = np.random.randn(hidden_size,
output_size) * 0.1
     self.bias_output = np.zeros(output_size)
     self.learning_rate = learning_rate

 def relu(self, x):
     return np.maximum(0, x)

 def relu_derivative(self, x):
     return np.where(x > 0, 1, 0)

 def softmax(self, x):
     z = x - np.max(x, axis=1, keepdims=True)
     numerator = np.exp(z)
     denominator = np.sum(numerator, axis=1, keepdims=True)
     return numerator / denominator

 def train(self, X, y, epochs=1000):
     for epoch in range(epochs):
         hidden_layer = self.relu(np.dot(X, self.
weights_input_hidden) + self.bias_hidden)
         output_layer = self.softmax(np.dot(hidden_layer, self.
weights_hidden_output) + self.bias_output)
         loss = -np.sum(y * np.log(output_layer))
         output_error = output_layer - y
         hidden_error = output_error.dot(self.
weights_hidden_output.T) * self.relu_derivative(hidden_layer)
         d_weights_output = hidden_layer.T.dot(output_error)
         d_bias_output = np.sum(output_error, axis=0, keepdims=
True)
         d_weights_input = X.T.dot(hidden_error)
         d_bias_input = np.sum(hidden_error, axis=0, keepdims=
True)
         self.weights_hidden_output -= self.learning_rate *
d_weights_output
         self.bias_output -= self.learning_rate * d_bias_output.
```

```
    sum(axis=0)
            self.weights_input_hidden -= self.learning_rate *
    d_weights_input
            self.bias_hidden -= self.learning_rate * d_bias_input.
    sum(axis=0)

     def predict(self, x):
         hidden_layer = self.relu(np.dot(x, self.
    weights_input_hidden) + self.bias_hidden)
         output_layer = self.softmax(np.dot(hidden_layer, self.
    weights_hidden_output) + self.bias_output)
         return np.argmax(output_layer, axis=1)


class ModifiedACCDigitalTwinWithNN:
     def __init__(self, neural_network, v_set=30, t_gap=1.4,
    D_default=10):
         self.vehicle_speed = 20
         self.acceleration = 0
         self.neural_network = neural_network
         self.speed_history = []
         self.v_set = v_set  # driver-set velocity
         self.t_gap = t_gap  # safe time gap
         self.D_default = D_default  # default safe distance

     def calculate_safe_distance(self):
         """Calculate the safe following distance based on current
    speed."""
         return self.vehicle_speed * self.t_gap + self.D_default

     def acc_logic(self, prediction, relative_distance):
         D_safe = self.calculate_safe_distance()
         if relative_distance >= D_safe:
             if self.vehicle_speed < self.v_set:
                 self.acceleration = 1
             else:
                 self.acceleration = 0
         else:
             if prediction == 0:
                 self.acceleration = 0
             elif prediction == 1:
```

```
                self.acceleration = 1
            elif prediction == 2:
                self.acceleration = -1

    def update(self, sensor_data):
        relative_distance = sensor_data[1]
        prediction = self.neural_network.predict(sensor_data.
   reshape(1, -1))
        self.acc_logic(prediction, relative_distance)
        self.vehicle_speed += self.acceleration
        self.speed_history.append(self.vehicle_speed)

    def run_simulation(self, sensor_data_samples):
        for sensor_data in sensor_data_samples:
            self.update(sensor_data)


# 4. Training and Simulation
y_one_hot = np.zeros((y.size, y.max() + 1))
y_one_hot[np.arange(y.size), y] = 1
mock_nn = SimpleNeuralNetwork(input_size=4, hidden_size=5,
   output_size=3)
mock_nn.train(X, y_one_hot, epochs=1000)
digital_twin_nn = ModifiedACCDigitalTwinWithNN(mock_nn)
digital_twin_nn.run_simulation(X)

# Display Results
print(digital_twin_nn.speed_history[:100])

# Saving the data
df1 = pd.DataFrame(X, columns=['relative_speeds', 'distances', '
   vehicle_types', 'road_conditions'])
df1.to_csv("sensor_data_123.csv", index=False)
```

A.4 ACC system Matlab code

```
%% Adaptive Cruise Control System Using Model Predictive Control
clear; close all; clc
```

```
%% Specify the linear model for ego car.
% G_ego = tf(1,[0.5,1,0]);

%% Define the sample time, Ts, and simulation duration, T, in
   seconds.
Ts = 0.1;
T = 80; % T = 80 sec

%% Gain value
verr_gain = 1.5;
xerr_gain = 1;
vx_gain = 1;

%% Value of car
m = 2000;    % kilogram
b = 0;
a = 0;

%% PID
kp = 1.06;
ki = 3.2;
kd = 0.02;

%% Specify the driver-set velocity in m/s.
v_set = 35; % v_set = 30

%% Specify the initial position for the two vehicles.
x0_lead = 50; % initial position for lead car 50 (m)
v0_lead = 25; % initial velocity for lead car 25 (m/s)
x0_ego = 10;  % initial position for ego car 10 (m)

%% The safe distance between the lead car and the ego car is a
   function
%% The ego car velocity
t_gap = 1.5; % t_gap = 1.5
D_default = 10; % D_default = 10

%% Considering the physical limitations of the vehicle dynamics,
   the
%% acceleration is constrained to the range |[-3,2]| (m/s^2).
```

```
amin_ego = -3; % amin_ego = -3
amax_ego = 2; % amax_ego = 2

% Array of initial velocities and accelerations to test
v0_ego_array = [20, 25, 30, 40];
a0_ego_array = [-1, 0, 1]; % Example array, adjust as needed

results = cell(length(v0_ego_array), length(a0_ego_array)); % To
   store results

for i = 1:length(v0_ego_array)
    for j = 1:length(a0_ego_array)
        v0_ego = v0_ego_array(i);
        a0_ego = a0_ego_array(j); % Assuming you have a variable
   a0_ego in your Simulink model for initial acceleration of ego
   car

        %% Run the simulation
        sim('project_trail_system')

        %% Store results
        results{i, j} = logsout;

        %% Plot results in the loop
        projectplot5(logsout, D_default, t_gap, v_set);
        title(['Simulation for v0_ego = ', num2str(v0_ego), ' m/s
   and a0_ego = ', num2str(a0_ego), ' m/s^2']);
    end
end

%% Optionally, after the loop, you can open the model
open_system('project_trail_system')  % open the Simulink Model '
   mpcACCsystem'
```

A.5 Function.m

```
function projectplot(logsout,D_default,t_gap,v_set)
%% Get the data from simulation
a_ego = logsout.getElement('a_ego');              % acceleration of
   ego car
```

```
v_ego = logsout.getElement('v_ego');              % velocity of ego
   car
a_lead = logsout.getElement('a_lead');            % acceleration of
   lead car
v_lead = logsout.getElement('v_lead');            % velocity of lead
    car
d_rel = logsout.getElement('d_rel');              % actual distance
d_safe = D_default + t_gap*v_ego.Values.Data;     % desired distance

%% Plot the results
figure('position',[100 100 960 800])

% acceleration
subplot(3,1,1)
plot(a_ego.Values.time,a_ego.Values.Data,'r',...
     a_lead.Values.time,a_lead.Values.Data,'b')
grid on
ylim([-3.2,2.2])
legend('ego car','lead car','location','SouthEast')
title('Acceleration')
xlabel('time (sec)')
ylabel('$m/s^2$','Interpreter','latex')

% velocity
subplot(3,1,2)
plot(v_ego.Values.time,v_ego.Values.Data,'r',...
     v_lead.Values.time,v_lead.Values.Data,'b',...
     v_lead.Values.time,v_set*ones(length(v_lead.Values.time),1),'k
   --')
grid on
ylim([15,50])
legend('ego car','lead car','set value','location','SouthEast')
title('Velocity')
xlabel('time (sec)')
ylabel('$m/s$','Interpreter','latex')

% distance
subplot(3,1,3)
plot(d_rel.Values.time,d_rel.Values.Data,'r',...
     d_rel.Values.time,d_safe,'b')
grid on
```

```
legend('actual distance','safe distance','location','SouthEast')
title('Distance between two cars')
xlabel('time (sec)')
ylabel('$m$','Interpreter','latex')
```

A.6 Result of ACC system simulation

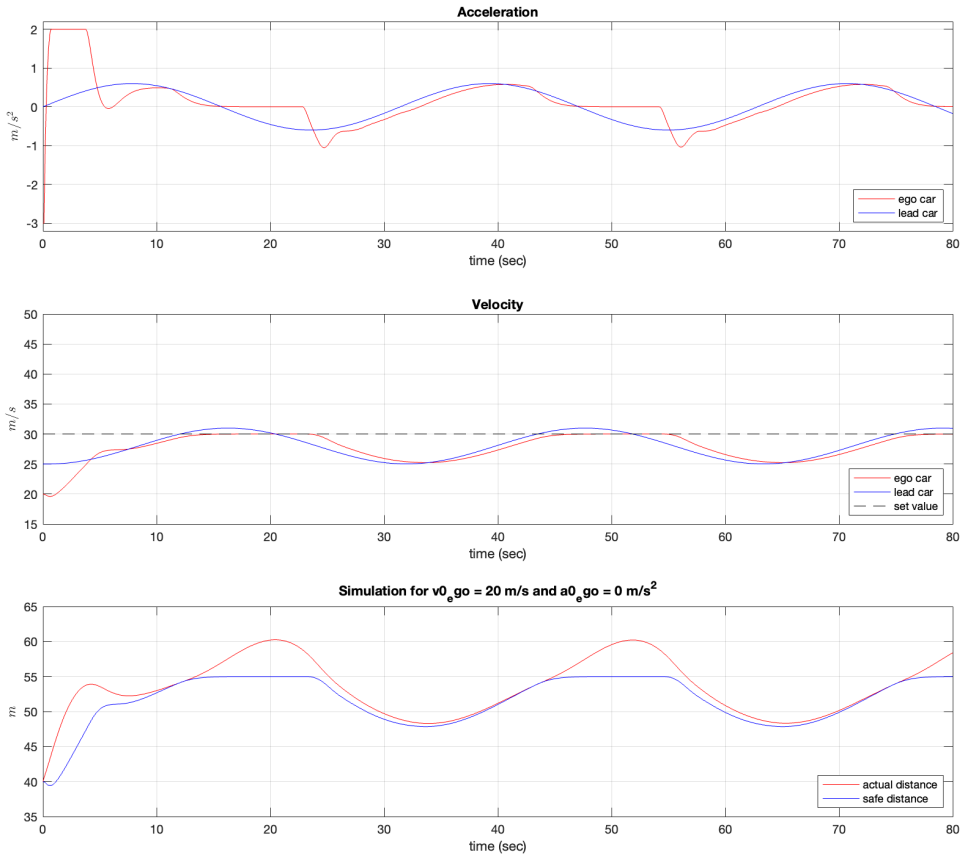

Figure A.6.1: Result 2

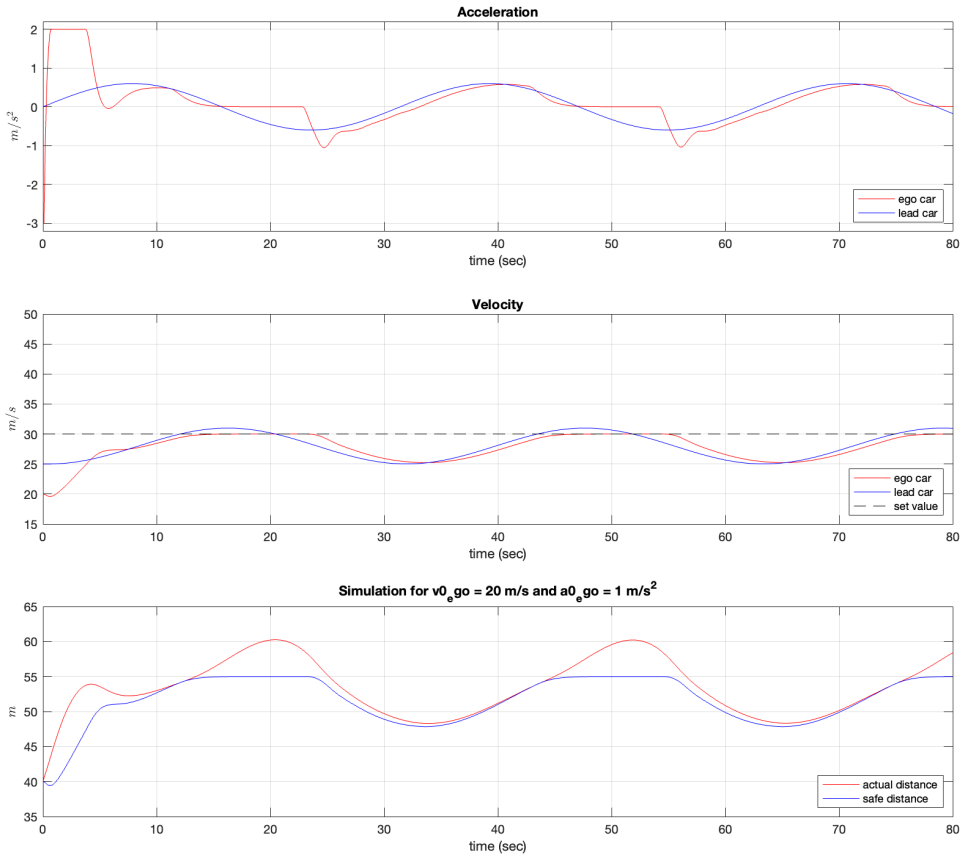

Figure A.6.2: Result 3

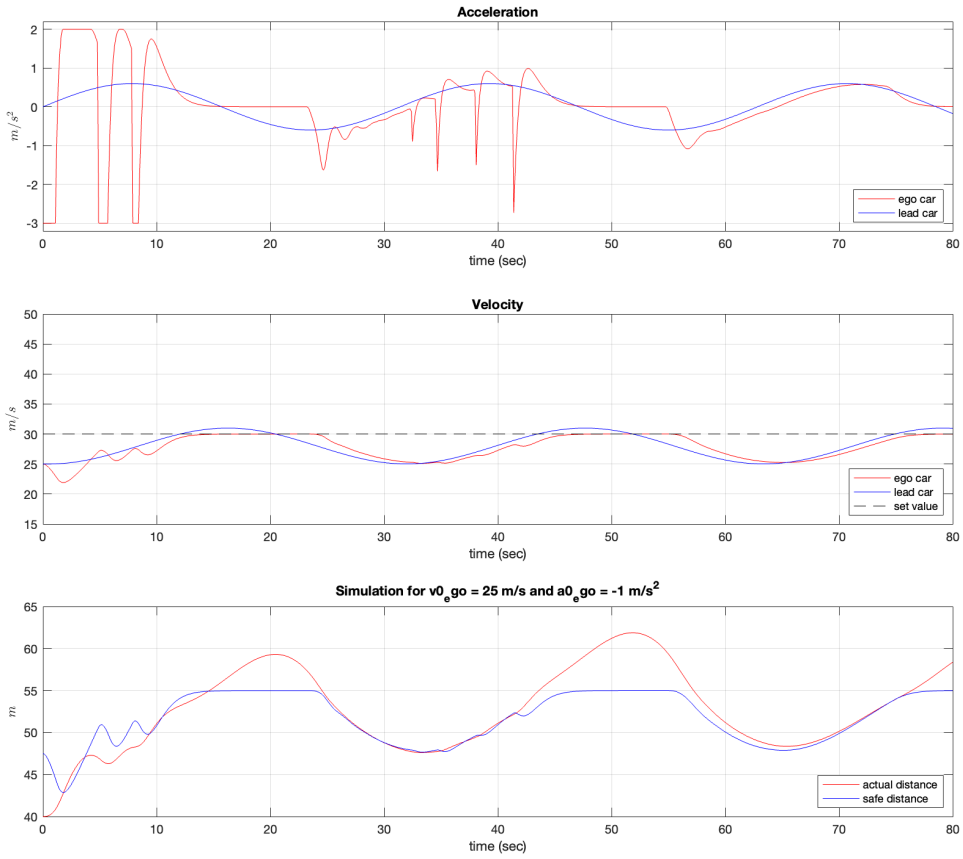

Figure A.6.3: Result 4

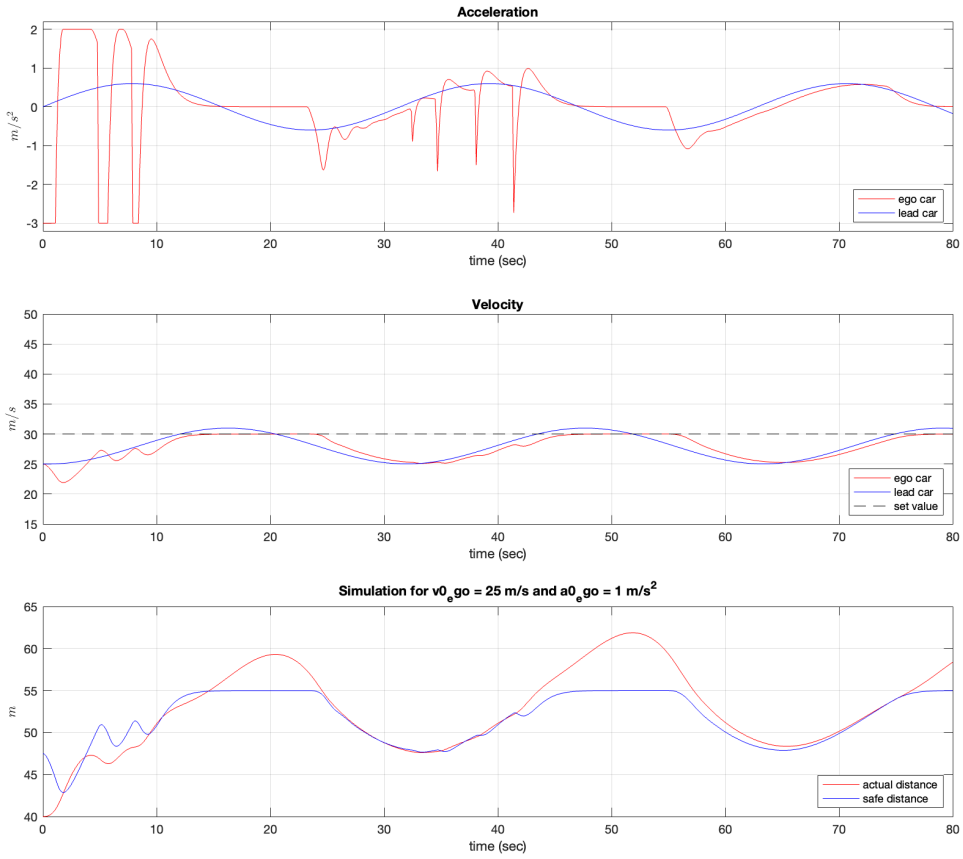

Figure A.6.4: Result 6

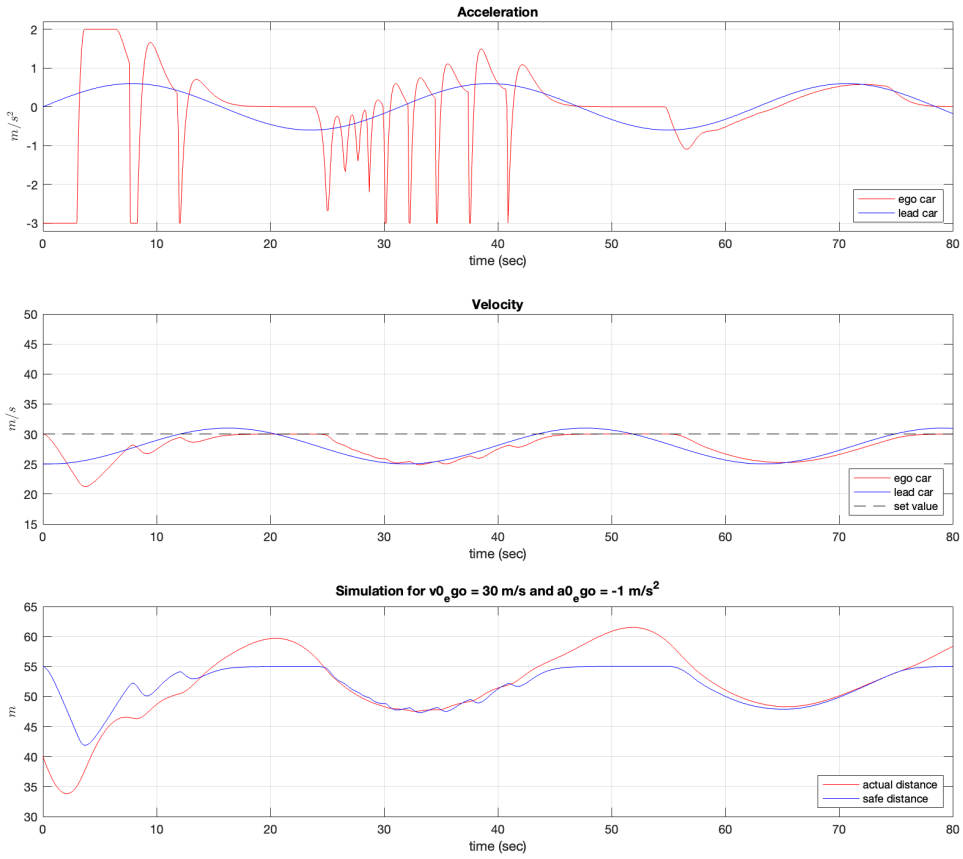

Figure A.6.5: Result 7

A.7 Result of Sensor_data

```
(myenv) (base) pc-65-232:Self-Driving-Car a0975464400$ /Users/
   a0975464400/Desktop/Essay/Project/Self-Driving-Car/myenv/bin/
   python /Users/a0975464400/Desktop/Essay/Project/Dataset/
   sensor_data.py
[21, 20, 21, 22, 21, 22, 23, 22, 21, 22, 23, 22, 21, 22, 21, 22,
   23, 22, 23, 24, 23, 24, 25, 26, 27, 28, 29, 28, 29, 28, 29, 28,
   29, 30, 30, 29, 28, 27, 26, 27, 26, 25, 26, 27, 26, 25, 26, 27,
   28, 29, 28, 27, 28, 27, 28, 27, 26, 27, 28, 29, 30, 29, 28, 27,
   28, 27, 26, 27, 28, 27, 26, 27, 26, 25, 26, 25, 24, 25, 26, 27,
   26, 27, 26, 27, 28, 29, 28, 29, 30, 30, 29, 28, 29, 30, 29, 30,
   29, 30, 30, 30]
(myenv) (base) pc-65-232:Self-Driving-Car a0975464400$
```

A.8 Result of neural network training

```
(myenv) (base) pc-65-232:Self-Driving-Car a0975464400$ /Users/
   a0975464400/Desktop/Essay/Project/Self-Driving-Car/myenv/bin/
   python "/Users/a0975464400/Desktop/Essay/Project/neural network
   training.py"
Epoch 1/10
13/13 [==============================] - 1s 28ms/step - loss:
   1.0979 - accuracy: 0.3295 - val_loss: 1.0964 - val_accuracy:
   0.3333
Epoch 2/10
13/13 [==============================] - 0s 5ms/step - loss: 1.0966
    - accuracy: 0.3523 - val_loss: 1.0975 - val_accuracy: 0.3333
Epoch 3/10
13/13 [==============================] - 0s 5ms/step - loss: 1.0954
    - accuracy: 0.3535 - val_loss: 1.0956 - val_accuracy: 0.3636
Epoch 4/10
13/13 [==============================] - 0s 5ms/step - loss: 1.0958
    - accuracy: 0.3434 - val_loss: 1.0957 - val_accuracy: 0.3737
Epoch 5/10
13/13 [==============================] - 0s 5ms/step - loss: 1.0955
    - accuracy: 0.3485 - val_loss: 1.0954 - val_accuracy: 0.3687
Epoch 6/10
13/13 [==============================] - 0s 5ms/step - loss: 1.0953
    - accuracy: 0.3649 - val_loss: 1.0984 - val_accuracy: 0.3384
Epoch 7/10
13/13 [==============================] - 0s 5ms/step - loss: 1.0941
    - accuracy: 0.3535 - val_loss: 1.0971 - val_accuracy: 0.3535
Epoch 8/10
13/13 [==============================] - 0s 5ms/step - loss: 1.0931
    - accuracy: 0.3699 - val_loss: 1.0969 - val_accuracy: 0.3687
Epoch 9/10
13/13 [==============================] - 0s 5ms/step - loss: 1.0933
    - accuracy: 0.3662 - val_loss: 1.0987 - val_accuracy: 0.3687
Epoch 10/10
13/13 [==============================] - 0s 6ms/step - loss: 1.0920
    - accuracy: 0.3725 - val_loss: 1.0995 - val_accuracy: 0.3687
```