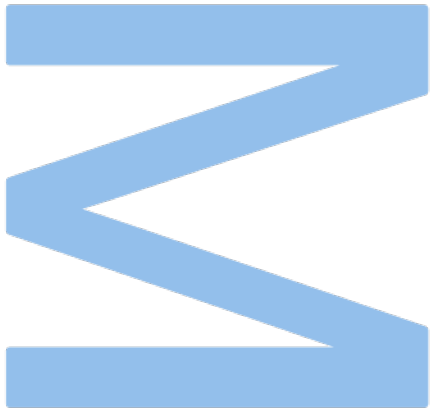
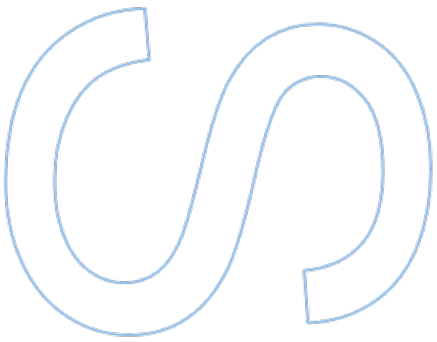
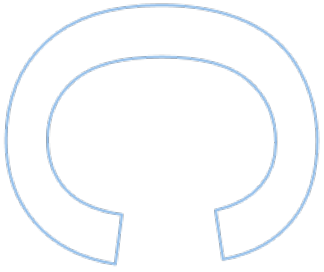
Visualization for a Smart-city Digital Twin for Autonomous Vehicles







António Gonçalo Bertelo
Mestrado em Ciência de Computadores
Departamento de Ciência de Computadores
2023

Supervisor
João Paulo da Conceição Soares, Doutorado Equiparado a Investigador Principal, Faculdade de Ciências da Universidade do Porto

Co-supervisor
Rolando da Silva Martins, Professor Auxiliar, Faculdade de Ciências da Universidade do Porto


U.PORTO
FACULDADE DE CIÊNCIAS
UNIVERSIDADE DO PORTO

Abstract

In this thesis, we introduce an approach to simulating and examining urban traffic dynamics via a comprehensive co-simulation framework. By harnessing the strengths of two renowned simulation platforms, Simulation of Urban Mobility (SUMO) and Car Learning to Act (CARLA), our system offers a vibrant environment that meticulously represents the interplay between vehicles, traffic signals, and diverse weather patterns.

Central to our approach is a custom-designed Application programming interface (API) that ensures fluid communication between the simulations. This facilitates real-time data exchange, synchronization, and integrates a suite of sensors for data validation. Such integration not only enhances the authenticity and relevance of our simulations but also enables the concurrent visualization and validation of both platforms.

To demonstrate the efficacy of our methodology, we undertake in-depth simulations encompassing LiDAR vehicle detection, intricate traffic management, and vehicle behavioral adaptations. Our assessments are rooted in several critical performance metrics, such as traffic throughput, mean journey duration, and vehicle queue durations.

The findings underline our co-simulation system's capability to yield invaluable insights for urban traffic governance and infrastructural design. Such insights bear significant ramifications for urban planners, policymakers, and technologists spearheading Advanced driver-assistance systems (ADAS) and autonomous driving innovations.

Ultimately, this research strengthens the chase of smart city solutions. By amalgamating diverse simulation components and presenting a rigorous analytical scaffold, our system paves the way for refining and strategizing the urban transport networks of the future.

Resumo

Nesta tese, apresentamos uma abordagem para simular e analisar a dinâmica do tráfego urbano usando uma framework abrangente de co-simulação. Utilizando os recursos de duas renomeadas plataformas de simulação, Simulation of Urban Mobility (SUMO) e Car Learning to Act (CARLA), o nosso sistema fornece um ambiente dinâmico que detalha precisamente a interação entre veículos, semáforos e diferentes condições climáticas.

O cerne da nossa abordagem é uma Application programming interface (API) especialmente desenvolvida que assegura uma comunicação eficaz entre as simulações. Isso facilita a troca de dados em tempo real, sincronização e ainda integra um conjunto de sensores para validação dos dados. Essa integração não apenas eleva a autenticidade e relevância das nossas simulações, como também possibilita a visualização e validação simultâneas em ambas as plataformas.

Para demonstrar a eficácia do nosso método, realizamos simulações aprofundadas, abordando a detecção de veículos com sensores Light Detection And Ranging (LIDAR), gestão de tráfego para otimizar o fluxo e testes ao comportamento dos veículos. As nossas avaliações baseiam se em diversas métricas de desempenho essenciais, como fluxo de tráfego, tempo médio de viagem e tempo de espera dos veículos.

Os resultados ressaltam a capacidade do nosso sistema de co-simulação em fornecer insights valiosos sobre o tráfego urbano. Tais insights são de grande relevância para planeadores urbanos, formuladores de políticas públicas e especialistas focados em inovações no Advanced driver-assistance systems (ADAS) e viagem autónoma.

Por fim, esta pesquisa reforça a busca por soluções em cidades inteligentes. Ao combinar diferentes componentes de simulação e apresentar uma robusta estrutura analítica, o nosso sistema aponta caminhos para aprimorar e planear as redes de transporte urbano do futuro.

Acknowledgments

"The best view comes after the hardest climb."

During this challenging period, the frame with this quote that I have above my computer often served as a reminder that challenges and difficulties are part and parcel of any journey worth undertaking.

I would like to express my deepest appreciation to all those who provided me with the possibility to complete this thesis.

I wish to thank my family for their unwavering support and encouragement throughout this challenging period as without them, this accomplishment would not have been possible. My gratitude also goes out to my friends and colleagues who were always there to listen, offer advice, and provide emotional support.

Contents

Abstract i

Resumo iii

Acknowledgments v

Contents xii

List of Tables xiii

List of Figures xvi

Listings xvii

Acronyms xix

1 Introduction 1

1.1 Motivation 1

1.2 Advancements in Simulation 2

1.3 Goals 2

1.4 Contribution 2

1.5 Outline of the Dissertation 2

2 Background 5

2.1 IoT and Sensor Networks in Smart Cities and Autonomous Vehicles 5

2.1.1 Importance of Sensor Data 6

2.1.2 Types of Sensors Used . . . 6
2.1.3 Data Aggregation Methods . . . 6
2.1.4 IoT Platforms . . . 6
2.1.5 Real-Time Monitoring and Decision Making . . . 6
2.1.6 Enhanced Safety Measures . . . 6
2.1.7 Synergy with Autonomous Vehicles . . . 7
2.2 Smart City . . . 7
2.3 Digital Twins . . . 8
2.3.1 Evolution and Fundamentals . . . 8
2.3.2 Applications in Diverse Domains . . . 8
2.3.3 Significance in Smart Cities . . . 9
2.4 Visualization Technologies for Smart City Digital Twins . . . 9
2.4.1 Virtual Reality (VR) . . . 9
2.4.2 Augmented Reality (AR) . . . 10
2.4.3 3D Modeling . . . 10
2.5 The Role of Autonomous Vehicles in Smart Cities . . . 10
2.5.1 Data Analytics and AI in Smart Cities and Autonomous Vehicles . . . 11
2.5.2 Real-world Case Studies of Digital Twins and Autonomous Vehicles in Smart Cities . . . 13

3 State of the Art **17**
3.1 CARLA . . . 17
3.1.1 OpenDRIVE and Environment Modelling . . . 18
3.1.2 Vehicular and Pedestrian Dynamics . . . 18
3.1.3 CARLA's Sensor Suite . . . 18
3.1.4 Scalability and Reproducibility . . . 20
3.1.5 Environmental Variability . . . 20
3.1.6 Integration and Extensibility . . . 21

3.1.7 Traffic Management and Benchmarks . . . 21
3.1.8 Future Potential . . . 21
3.2 SUMO . . . 21
3.2.1 XML-based Simulations . . . 21
3.2.2 Route Generation and Management . . . 21
3.2.3 Traffic Control Interface (TraCI) . . . 22
3.2.4 NETEDIT: A Graphical Network Editor for Simulation of Urban Mobility (SUMO) . . . 22
3.3 Co-simulation: Integrating Car Learning to Act (CARLA) and SUMO . . . 23
3.3.1 Map Compatibility . . . 23
3.3.2 Synchronization . . . 24
3.4 CARLA and SUMO in the Context of Smart Cities, Digital Twins, and Autonomous Vehicles . . . 24
3.5 mySMARTLife Project for Helsinki's 3D City Model . . . 25
3.5.1 Initial Outcomes . . . 25
3.5.2 Integration of Real-time Data . . . 25
3.5.3 Evolution of the 3D Model . . . 26
3.5.4 Digital Twin and User Engagement . . . 26
3.5.5 Enhancements to Sensor Data Collection . . . 26
3.5.6 Prioritizing Data Privacy . . . 26
3.6 An Integrated Traffic and Vehicle Co-Simulation Testing Framework for Connected and Autonomous Vehicles . . . 26
3.6.1 Emphasis on Real-world Data . . . 27
3.6.2 Trajectory Extraction: Easier Said Than Done . . . 27
3.6.3 Vehicle Detection and Tracking: Leaning on Advanced Techniques . . . 27
3.6.4 Flight Perspective: A New Angle . . . 28
3.6.5 Network Generation in SUMO . . . 28
3.6.6 Demand Modeling in SUMO . . . 28
3.6.7 Integration of SUMO and CARLA . . . 28

3.6.8 Case studies . . . 29
3.6.9 A Highway Scenario for CAVs . . . 31
3.7 A Novel Traffic Simulation Framework for Testing Autonomous Vehicles Using SUMO and CARLA . . . 32
3.7.1 Object Detection Algorithm . . . 32
3.7.2 Case Study and Results . . . 33
3.8 OpenCDA: A framework for Cooperative Driving Automation . . . 33
3.8.1 Key Features of OpenCDA . . . 34
3.8.2 Emphasis on Co-Simulation: SUMO and CARLA . . . 34
3.8.3 Practical Application: Platooning . . . 35
3.9 Framework for Interactive M3 Visualization of Microscopic Traffic Simulation . . . 36
3.9.1 Background: Paramics . . . 36
3.9.2 Virtual Environment and Signal Control . . . 36
3.9.3 Distributed Multi-User Visualization . . . 36
3.9.4 Versatility in User Perspective . . . 36
3.9.5 Innovative Multi-Mode Visualization . . . 37
3.10 Conclusion . . . 38

4 Objectives and System Architecture **41**

4.1 Objectives . . . 41
4.2 System Architecture Review . . . 42
4.2.1 CARLA: Realism in Vehicle Dynamics . . . 42
4.2.2 SUMO: Traffic Dynamics and Light Control . . . 42
4.2.3 NETEDIT: Configuring Demand Patterns . . . 42
4.2.4 TraCI: Synchronization and Interaction . . . 43
4.2.5 Dynamic Interplay . . . 43
4.2.6 Simulation Input Specifications . . . 43
4.2.7 Harmonizing Map Representations . . . 44

4.2.8 Co-Simulation Choreography . . . 44
4.2.9 System Overview . . . 45

5 Development **49**

5.1 The map . . . 49
5.1.1 Need for Realistic Environments . . . 49
5.1.2 Advantages of Predefined Environments . . . 49
5.1.3 Map Conversion to XML Format . . . 50
5.2 Demand . . . 52
5.2.1 Mapping CARLA's Vehicle Blueprints to SUMO's Vehicle Types . . . 52
5.2.2 Creating Demand Elements in Netedit . . . 52
5.3 Detailed Traffic Scenario . . . 54
5.3.1 Traffic Light Centric Testing . . . 54
5.4 API . . . 55
5.4.1 Weather Control . . . 57
5.4.2 Return live information . . . 58
5.4.3 Behavior Modifiers and Their Significance . . . 58
5.4.4 Traffic Light Management . . . 60
5.4.5 Sensors and Data Collection . . . 62
5.4.6 Vehicle Detection using Light Detection And Ranging (LIDAR) . . . 62
5.5 Resume . . . 65

6 Evaluation **67**

6.1 Objectives . . . 67
6.2 Methodology . . . 68
6.2.1 Simulation Test Procedure: . . . 68
6.2.2 Consistency and Reproducibility: . . . 68
6.2.3 Weather Dynamics . . . 69
6.2.4 Traffic Flow . . . 70

6.2.5 Driving Behaviors . . . 71
6.2.6 Sensor Testing . . . 71
6.3 Results and Discussion . . . 72
6.3.1 Consistency and Reproducibility . . . 72
6.3.2 Weather Dynamics . . . 73
6.3.3 Traffic Flow . . . 75
6.3.4 Queue in lanes analysis . . . 79
6.3.5 Emissions Analysis . . . 80
6.3.6 Driving Behaviors . . . 81
6.3.7 Sensor Testing . . . 82
6.3.8 Vehicle Detection using LIDAR . . . 82
6.3.9 Overall System Performance . . . 86
6.4 Conclusion . . . 86

7 Conclusions **87**

Bibliography **89**

List of Tables

6.1 Comparison of traffic metrics before and after the implementation of the traffic light algorithm. 77

6.2 Comparison of median trip duration for each predefined route. 79

6.3 Different scenarios generated based on Tau, Sigma, and Impatience. 81

List of Figures

2.1 The concept of Internet of Things (IoT) 5

2.2 The concept of a smart-city digital twin 7

2.3 Virtual Reality 9

2.4 Augmented Reality 10

2.5 3D Modeling (Image By freepik) 10

3.1 LIDAR visual representation 18

3.2 RADAR visual representation 19

3.3 RGB Camera output 20

3.4 mySMARTLife project overview 25

3.5 Overview of co-simulation utilized 30

3.6 The morning vs the night scenario 33

3.7 Platooning scenario testing under co-simulation settings[36] 35

3.8 Visualization system overview[37] 37

3.9 Multiple views created: (a) menu, (b) bird's-eye, (c) pedestrian, (d) car follow and (e) driver)[37] 37

4.1 System input overview 44

4.2 System architecture 46

5.1 Map overview 50

5.2 Road Network in Town 05 51

5.3 Translated Map on Simulation of Urban Mobility (SUMO) 51

5.4 Map in NETEDIT . . . 53
5.5 The process of defining routes using the available edges, denoted in green. . . . 53
5.6 Intersection overview . . . 54
5.7 Car Learning to Act (CARLA) and SUMO running side by side . . . 57
5.8 Sensors placement . . . 63
5.9 Light Detection And Ranging (LIDAR) Sensor output . . . 64

6.1 Deviation throughout all simulations . . . 73
6.2 Different type of environmental conditions . . . 74
6.3 Traffic intersection overview . . . 75
6.4 Duration per route per lane . . . 76
6.5 Queue times per route per lane . . . 77
6.6 Duration per route before algorithm integration. . . . 78
6.7 Duration per route after algorithm integration. . . . 78
6.8 Total Queue time for each lane over time . . . 80
6.9 CO2 emissions without managing traffic . . . 80
6.10 CO2 emissions after managing traffic . . . 80
6.11 Impact of Driving Behavior on Emissions and Trip Duration . . . 81
6.12 Radar processing . . . 83
6.13 LiDAR placement . . . 83
6.14 LiDAR sensor scanning with no car . . . 84
6.15 LiDAR sensor scanning cars . . . 84
6.16 Overview of last simulation step . . . 85

Listings

5.1 SUMO vehicle type example 52
5.2 Main API initialization method 55
5.3 Initializing CARLA module 55
5.4 Initializing SUMO module 56
5.5 Connecting both Simulations into the synchronization module 56
5.6 Method to start the simulation 56
5.7 Method to set weather 58
5.8 Method to modify behaviours conditions 59
5.9 Example of SUMO vehicle type with the behavioral changes 59
5.10 Utility Function 61
5.11 Traffic light algorithm 61
5.12 Vehicle detection algorithm 65

Acronyms

AD Autonomous Driving

AI Artificial Intelligence

DT Digital Twin

GUI Graphical User Interface

IoT Internet of Things

SCDT Smart-City Digital Twin

VR Virtual Reality

SUMO Simulation of Urban Mobility

CARLA Car Learning to Act

ADAS Advanced driver-assistance systems

AR Augmented Reality

SM Smart-City

AVs Autonomous Vehicles

API Application programming interface

TraCI Traffic Control Interface

IP Internet Protocol

LIDAR Light Detection And Ranging

IMU Inertial Measurement Unit

GNSS Global Navigation Satellite System

Chapter 1

Introduction

The dawn of the 21st century has seen the relentless and rapid expansion of urban areas worldwide[38]. Forecasts indicate that this urbanization trend will continue, with cities swelling in both size and population[38]. This immense urban growth presents not only opportunities but also strains on resources.As cities continue to grow, there is an imminent need for innovative methods of management to ensure sustainability.

Ence the concept of "smart cities". The term, which has gained traction in recent years, represents a vision of urban development that harnesses the power of pervasive and mobile computing systems[14]. Thanks to the widespread availability of commodity low power sensors, smart devices like phones and tablets, and robust wireless networking infrastructures, cities are on the cusp of a revolution. Leveraging technologies such as Artificial Intelligence Artificial Intelligence (AI), the Internet of Things Internet of Things (IoT), and Big Data analytics, smart cities aim to address urban challenges ranging from traffic congestion and pollution to energy efficiency and parking management[38]. The promise is clear: cities that are not only smarter in their operation but also more sustainable and livable.

Parallel to this development is the rise of self-driving cars, which promises to redefine urban transportation. Autonomous vehicles, with their potential to improve traffic flow, reduce accidents, and transform urban landscapes, are increasingly becoming integral components of the smart city framework[31].

In essence, as the modern world grapples with the challenges and promises of rapid urbanization, smart cities emerge as a beacon of hope. They epitomize the potential of modern technology from AI to IoT in optimizing and enhancing the urban experience for all its inhabitants.

1.1 Motivation

Smart cities utilize an abundance of sensors and devices to collect and analyze extensive data. This data-driven approach aids in optimizing resource allocation and decision-making processes

for policymakers and citizens alike. Among the technological advances, the integration of autonomous vehicles promises a transformative impact on road safety, traffic management, and energy efficiency. Nevertheless, the complexity of integrating these vehicles into existing infrastructures demands rigorous simulation and testing.

1.2 Advancements in Simulation

Simulation platforms like Car Learning to Act (CARLA) and Simulation of Urban Mobility (SUMO) have emerged as indispensable tools for modeling real world scenarios in smart cities. CARLA is renowned for its accurate sensor models for autonomous vehicles, while SUMO excels at replicating realistic traffic behavior. Combining these platforms enables the comprehensive evaluation of various smart city scenarios, which is crucial for developing and refining advanced driving systems.

1.3 Goals

The primary objective of this thesis is to understand the intricacies of integrating CARLA and SUMO for simulating urban traffic scenarios,vehicle behaviours, and consequently analysing consequences on traffic flow, queue times and emissions. This understanding would serve as the foundation to evaluate the performance of autonomous vehicle perception systems in different simulated environments. We aim to derive insights on the importance of multi sensor integration and its potential impact on the future integration of autonomous vehicles in urban landscapes.

1.4 Contribution

This research, at its core, emphasizes the significance of a hybrid simulation approach by marrying the strengths of CARLA and SUMO. Through our experiments, we highlight the challenges and limitations of current perception models in diverse simulated scenarios. Our findings underscore the need for comprehensive multi-sensor strategies to enhance the robustness of autonomous vehicles, especially in suboptimal conditions. This work contributes to the broader discourse on ensuring safety and reliability in autonomous vehicular systems for future smart cities.

1.5 Outline of the Dissertation

This dissertation is systematically structured into six comprehensive chapters.

1. **Introduction (1)**: This initial chapter provides an overarching perspective on the research topic. It outlines the central problem, highlights the primary motivations, and offers a concise

roadmap about the contribution and goals.

2. **Background (2)**: This chapter delves into the foundational theories and prior works that inform the research. It aims to ground in the existing knowledge while pointing out gaps that this research addresses.

3. **State of the Art (3)**: Here, a detailed review of contemporary research and technologies pertinent to the topic is presented. This chapter delineates the current landscape, emphasizing the areas this project intends to innovate in.

4. **Objectives and System Architecture (4)**: We introduce our objectives in this section as well as a first approach of the architecture implemented.

5. **Development (5)**: This chapter describes the creation and refinement of the system under study. It dives into the methodologies, tools, and design principles adopted to architect and develop the proposed solution.

6. **Evaluation (6)**: In this section, rigorous tests and experiments conducted on the developed system are discussed. It elucidates the testing methodologies, outcomes, and the implications of these results on the research.At last a analysis of the results obtained are discussed and future solutions are given.

7. **Conclusion (7)**: This final chapter encapsulates the findings, contributions, and implications of the research. It also provides a perspective on potential future work and the broader impact of the study.

Chapter 2

Background

This introductory section serves as a primer for a comprehensive examination of the fundamental theories and core principles central to our research focus on the Smart-City Digital Twin (SCDT) and its interplay with Autonomous Driving (AD). It aims to equip with a detailed contextual background, laying a solid foundation for the in-depth discussions and analyses that will ensue in subsequent chapters. This overview is instrumental in fostering a more nuanced understanding of the study's domain.

2.1 IoT and Sensor Networks in Smart Cities and Autonomous Vehicles

The Internet of Things (IoT), at its essence, is a vast network of interconnected devices equipped with sensors, software, and communication capabilities. This network extends its tendrils across the landscape of smart cities and the operational landscape of autonomous vehicles, fostering an environment of constant data exchange and synergy.[4].



Figure 2.1: The concept of IoT

2.1.1 Importance of Sensor Data

Sensor data serves as the backbone of both smart cities and autonomous vehicles, facilitating real-time monitoring, control, and decision-making[17]. From environmental monitoring to traffic management, the data collected is integral for various applications within these systems[16].

2.1.2 Types of Sensors Used

Various types of sensors are deployed in smart cities and autonomous vehicles to capture a wide range of data. In smart cities, environmental sensors measure air quality and temperature, while traffic sensors collect data on vehicle speed and density[17]. For autonomous vehicles, **Light Detection And Ranging (LIDAR)**, **Radar**, and **Cameras** are commonly used to gather navigational and situational data[42].

2.1.3 Data Aggregation Methods

Data aggregation is a crucial step in sensor networks, primarily to reduce the amount of data to be sent to the servers, saving both energy and bandwidth[17]. Various methods, including simple averaging, weighted averaging, or even more complex machine learning models, are employed to aggregate data at the sensor or gateway level before sending it for further analytics[17].

2.1.4 IoT Platforms

IoT platforms often serve as the intermediary that manages sensor data, analytics algorithms, and user applications. These platforms are essential for coordinating the various components of smart cities and autonomous vehicles, ensuring that data flows smoothly from sensors to applications and decision-makers[22].

2.1.5 Real-Time Monitoring and Decision Making

Sensors provide real-time data, ranging from traffic conditions to environmental metrics such as air quality and temperature. This real-time monitoring enables both autonomous systems and city managers to make instant decisions, optimizing traffic flow or rerouting vehicles as necessary to improve urban mobility[20].

2.1.6 Enhanced Safety Measures

Sensors play a critical role in ensuring road safety in a smart city. Advanced sensor technologies in Autonomous Vehicles (AVs) can detect obstacles, assess road conditions, and even predict

pedestrian movements. In conjunction with traffic management systems, this enhances the overall safety of urban transportation networks [35].

2.1.7 Synergy with Autonomous Vehicles

In the realm of autonomous vehicles, sensors are pivotal for safe and efficient operation[20]. Data from city-wide sensors can be integrated into the decision-making algorithms of AVs, allowing for more harmonious interaction between autonomous and human-operated vehicles. This represents a future-forward step in integrated, smart urban mobility[20].

2.2 Smart City

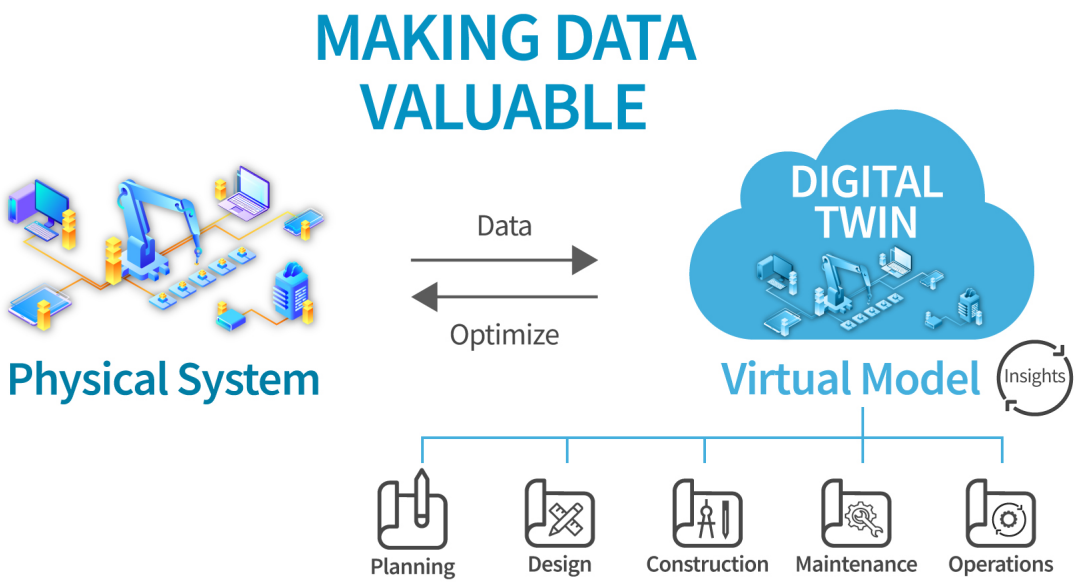


Figure 2.2: The concept of a smart-city digital twin

The concept of a Smart-City (SM) stands at the crossroads of urban planning, information technology, and sustainable development.Modern technology and data analytics are used by smart cities to raise the standard of living for its residents, streamline municipal processes, and pursue sustainability objectives[10]. For a comprehensive view of the smart city digital twin's functionality, see Figure 2.2. Urban ecosystems have advanced quickly as a result of the fusion of technologies like the IoT, Artificial Intelligence (AI), and Big Data analytics[34]. Researchers are particularly interested in how Smart Cities might use these technologies for better waste management, energy conservation, and traffic control, among other applications [24].

Central to the idea of smart cities is the potential for data to revolutionize urban living, but it's only through effective visualization that data can truly be leveraged. Visualization, in the realm of smart cities, is not merely about presenting data in graphical formats[29]. Instead, it's about crafting a narrative that aids decision-making, fosters citizen engagement, and facilitates collaborative urban development. Tools and platforms that provide immersive experiences, like Augmented Reality (AR) and Virtual Reality (VR), further accentuate the potential of visualization.

2.3 Digital Twins

Digital Twins represent a fusion of the physical and digital realms, offering a holistic view of products, processes, and systems. This synergy creates opportunities for performance optimization, predictive maintenance, and enhanced user experience [24].

2.3.1 Evolution and Fundamentals

The inception of Digital Twins can be traced back to the early 2000s, primarily in the realm of aerospace and defense[15]. These were complex systems that benefited immensely from accurate digital replicas for testing and simulation [27]. Over the years, advancements in Internet of Things (IoT) devices, machine learning algorithms, and computational capacity have further expanded the possibilities and applications of Digital Twins[7].

The fundamental pillars of a Digital Twin encompass[27]:

- **Physical Products/Systems**: Real-world entities that are under observation or analysis.
- **Digital Representation**: A virtual model capturing the physical system's essential aspects and behaviors.
- **Data Connection**: Bi-directional data flow between the physical and digital entities, ensuring real-time synchronicity .

2.3.2 Applications in Diverse Domains

Beyond their origins in aerospace and defense, Digital Twins are making inroads into various sectors, including healthcare (for patient monitoring), agriculture (for crop and livestock management), and even retail (for customer experience enhancement) [39].

2.3.3 Significance in Smart Cities

Digital Twins, in the context of Smart Cities, are poised to revolutionize urban planning, infrastructure management, and public services. Some specific applications include:

- **Infrastructure Monitoring**: With sensors embedded in critical infrastructure components like bridges, tunnels, and roads, Digital Twins can predict wear and tear, optimizing maintenance schedules and preventing catastrophic failures [5].
- **Energy Management**: Digital replicas of power grids can facilitate optimal power distribution, reducing waste and ensuring uninterrupted power supply[25].
- **Traffic Flow Optimization**: By capturing real-time traffic data and simulating different scenarios, cities can proactively manage congestion and improve public transport services [32].

The multifaceted applications of Digital Twin (DT) underscore their significance in testing smart cities. As a key technological component, DT not only enable the proactive management of urban challenges but also contribute to the development of resilient, efficient, and sustainable urban ecosystems.

2.4 Visualization Technologies for Smart City Digital Twins

2.4.1 VR

Virtual Reality offers an immersive experience that enables users to interact with a simulated environment[41]. For smart city digital twins, VR has emerged as an indispensable tool for planners, engineers, and stakeholders to experience urban settings before they exist. They provide a simulated environment that closely resembles the actual physical spaces, thereby allowing city officials to understand the implications of their designs or decisions in a more tangible manner[41].



Figure 2.3: Virtual Reality

2.4.2 AR

While VR offers a fully immersive experience, Augmented Reality overlays digital information on the real world. This technology has been used for a variety of applications within smart cities, including augmented maintenance for infrastructure, real-time analytics display, and citizen engagement tools. For instance, on medicine “Recent advances in medical imaging have provided scientists and physicians a huge amount of data that could be profitable used to support several activities ” [26].



Figure 2.4: Augmented Reality

2.4.3 3D Modeling

3D models are at the core of any digital twin technology. They provide the necessary spatial geometry and can be linked with databases to provide real-time information about different city components[6]. These models are vital for the visualization of architectural elements, urban planning, and for running simulations to assess the impact of various environmental factors on the city’s infrastructure[6].



Figure 2.5: 3D Modeling (Image By freepik)

2.5 The Role of Autonomous Vehicles in Smart Cities

AVs, often known as self-driving cars, leverage an array of advanced sensors, such as LIDAR and Radar, along with machine learning algorithms to navigate and operate without the need for

human control[3]. Their emergent role in the urban transportation landscape is widely considered a potentially disruptive force that could redefine the paradigms of mobility and transportation [8].

One of the most immediate and visible impacts of AVs is their potential to alleviate traffic congestion. By utilizing real-time traffic data and sophisticated algorithms, AVs can adjust their routes dynamically, thus optimizing the flow of traffic across various parts of the city. This dynamic rerouting capability could significantly reduce the average time spent in traffic and improve overall traffic flow [3].

Beyond the realm of traffic management, AVs also offer environmental benefits. Given their optimized driving patterns and the possibility for electric operation, they stand as a cleaner, more sustainable alternative to traditional automobiles. Their precise driving capabilities allow for fuel-efficient travel, which in turn can help in lowering emissions of greenhouse gases and other pollutants. Thus, they contribute not only to a more efficient transportation network but also to a more sustainable urban environment [3].

In terms of road safety, the data-driven operation of AVs minimizes the risk of accidents caused by human error, which includes distracted driving, impaired driving, and other forms of recklessness. Although the technology is still evolving, initial studies indicate a potential for significantly safer roads as the market share of AVs increases [3].

The seamless integration of AVs into the digital twins of smart cities presents exciting possibilities. Digital twins, with their capability to simulate and analyze multiple urban scenarios in real time, can offer valuable insights into optimizing AV operations. For instance, how would a fleet of AVs respond to a sudden change in weather conditions, or how might they interact with public transport services? Such scenarios can be tested and optimized in the digital twin environment, allowing for a nuanced understanding of AVs' impact on urban dynamics [11].

The incorporation of AVs into the larger smart city ecosystem thus offers a multi-faceted range of benefits, from operational and environmental to safety-related aspects. However, this also necessitates comprehensive planning and regulation to ensure that the deployment of AVs aligns with the broader objectives of urban sustainability and social inclusivity [11].

2.5.1 Data Analytics and AI in Smart Cities and Autonomous Vehicles

2.5.1.1 Role of Data Analytics in Smart Cities

Data analytics is an essential component in the successful deployment of smart cities[18]. Advanced data analytics techniques process vast amounts of data generated from various sensors and devices in real time. These techniques are crucial for interpreting complex datasets, thereby aiding in urban planning, traffic management, and resource allocation. Data analytics can significantly improve the efficiency, sustainability, and quality of life in urban settings.“The application of big data in a smart city has many benefits and challenges, including the availability of large computational and storage facilities to process streams of data produced within a smart city

environment."[18]

2.5.1.2 Machine Learning in Smart Cities

Machine learning, a subset of artificial intelligence, is increasingly being adopted in smart cities for predictive analytics and decision-making. For instance, machine learning algorithms can predict traffic congestion and suggest alternate routes, thereby optimizing the flow of vehicles. Also, they can analyze past power consumption patterns to optimize energy production, thereby contributing to more sustainable cities[19].

2.5.1.3 Artificial Intelligence in Autonomous Vehicles

The integration of AI into autonomous vehicles is a breakthrough that has significantly advanced their development and implementation[1]. AI algorithms process data from various sensors in real-time, making instantaneous decisions that control the vehicle's movement, from speed adjustment to obstacle detection and avoidance. These algorithms are continually improved via machine learning, adapting to new data and scenarios to perform more reliably over time.

2.5.1.4 Real-time Decision-making and Predictive Models

Both smart cities and autonomous vehicles benefit from real-time decision-making capabilities powered by data analytics and AI[19]. In the context of a smart city, predictive models can be developed to simulate various scenarios, from natural disasters to peak energy consumption times[19]. For autonomous vehicles, real-time analytics and predictive modeling are critical for safe navigation and optimal route planning.

2.5.1.5 Synergy between Smart Cities and Autonomous Vehicles

The confluence of data analytics and AI not only serves smart cities or autonomous vehicles independently but also creates a symbiotic relationship between the two. Smart city platforms can integrate data from autonomous vehicles to enhance urban services and infrastructure. Similarly, autonomous vehicles can benefit from smart city data, such as real-time traffic conditions, to make more informed decisions[19].

2.5.2 Real-world Case Studies of Digital Twins and Autonomous Vehicles in Smart Cities

2.5.2.1 Implementation of Digital Twins in Urban Planning

There are numerous instances where digital twins have been effectively used in the urban planning phase. Cities like Singapore and Amsterdam have successfully implemented digital twins to visualize and predict various urban scenarios, thereby aiding in more efficient planning and management[28].

2.5.2.2 Autonomous Vehicles for Public Transport

Several cities have begun piloting autonomous vehicles in their public transport systems. For example, self-driving buses and taxis have been tested in various cities, providing insights into how autonomous vehicles can be integrated into existing public transit systems[40].

2.5.2.3 Traffic Management using Digital Twins

Digital twins also find applications in real-time traffic management. They are employed to simulate and analyze different traffic situations, helping traffic controllers make quick and informed decisions[9]. This has been practically applied in cities like Barcelona and Melbourne to reduce traffic congestion and improve road safety[28].

2.5.2.4 Digital Twins in Infrastructure Maintenance

Real-world implementations also demonstrate the usefulness of digital twins in the maintenance of urban infrastructure[12]. For example, they have been used to monitor the structural integrity of bridges and roads, aiding in timely maintenance and ensuring public safety[12].

2.5.2.5 Integration of Autonomous Vehicles and Digital Twins

Some pioneering smart city projects have successfully integrated autonomous vehicles with digital twins. These integrated systems allow for real-time simulation and analytics that help in better vehicle routing, avoiding traffic bottlenecks, and improving overall transportation efficiency[40].

2.5.2.6 Smart Grids and Energy Management

Digital twins have also been used in the creation and management of smart grids, aiding in more efficient energy usage[12]. Case studies demonstrate how energy consumption can be optimized in real-time, balancing supply and demand effectively[12].

AVs stand as a testament to the confluence of innovation, technology, and transportation. Their promise of safer, more efficient, and more sustainable urban transport aligns with the Smart City vision. For AVs to become an everyday reality, rigorous testing, and validation in varied scenarios are paramount. Here, tools like Car Learning to Act (CARLA), with its rich set of sensors and realistic rendering, play a pivotal role. Simulation of Urban Mobility (SUMO), on the other hand, offers a granular perspective on traffic management, allowing researchers to study AVs behavior in diverse traffic conditions, from congested city centers to free-flowing highways. In summary, as urban centers grow and technology continues its relentless march forward, tools like CARLA and SUMO will be at the heart of research and development. They serve as bridges between the digital and physical realms, ensuring that the cities of tomorrow are not just smarter but also safer, efficient, and sustainable.

As we conclude this chapter, it is essential to acknowledge the challenges and outline potential directions for future research in the intersection of SCDT and AD. While the integration of sensor networks, IoT, and autonomous vehicles holds immense promise, it is not without its complexities.

Firstly, issues of data privacy and security loom large. With the extensive collection of sensitive information for both smart cities and autonomous vehicles, safeguarding data against unauthorized access and malicious attacks becomes a critical concern. Addressing these challenges necessitates the development of robust encryption protocols, secure communication channels, and limited access controls.

Secondly, the scalability of these systems poses a significant challenge. As smart city infrastructure and autonomous vehicle fleets expand, ensuring that the underlying technologies can seamlessly scale to accommodate the growing data volume and network complexity becomes imperative. Scalability issues may hinder the full realization of the potential benefits these technologies promise.

Additionally, the ethical implications of deploying AD in urban environments require careful consideration. Decisions made by autonomous vehicles, especially in complex and ambiguous situations, raise questions about accountability, liability, and the ethical frameworks guiding their behavior. Balancing technological progress with ethical considerations is an ongoing challenge.

Looking forward, future research should delve into refining the integration of SCDT and AD by addressing these challenges. This includes the development of advanced machine learning algorithms for real-time decision-making in autonomous vehicles, enhanced data aggregation techniques for sensor networks, and the exploration of innovative solutions to privacy and security concerns.

Moreover, as technology continues to advance, exploring the potential synergies between emerging technologies such as 6G networks, edge computing, and blockchain with SCDT and AD could unlock new possibilities. These advancements may further enhance the efficiency, responsiveness, and sustainability of smart cities and autonomous transportation.

In conclusion, the intersection of SCDT and AD represents a dynamic and evolving field with the potential to redefine urban living. As we move forward, it is crucial to navigate the challenges and leverage emerging technologies to create smarter, safer, and more sustainable urban environments. This sets the stage for subsequent chapters, where we will delve deeper into some projects that explore this topic, methodologies, experiments, and findings that contribute to the advancement of knowledge in this interdisciplinary domain.

Chapter 3

State of the Art

The evolution of technology has reshaped urban landscapes, ushering in the era of Smart Cities (Smart-City (SM)) dedicated to enhancing efficiency, sustainability, and convenience for their residents. At the heart of this transformation lies the intricate interplay between Digital Twins (Digital Twin (DT)), Autonomous Vehicles (Autonomous Vehicles (AVs)), and complementary technologies such as visualization tools, data analytics, and the Internet of Things (Internet of Things (IoT)). This chapter undertakes a comprehensive review of existing literature in these domains, examining their present state, capabilities, and real-world applications.

The rise of smart cities and their visual representation has spurred a surge in innovations, particularly within the realm of traffic management. Globally, researchers and professionals are harnessing various sensor technologies to engineer advanced systems, optimizing urban transportation.

As we embark on the upcoming chapters, our journey delves into the intricacies of technology. We will explore the depths of available technologies, unraveling the complexities of simulation platforms like Car Learning to Act (CARLA) and Simulation of Urban Mobility (SUMO). Beyond mere features, our exploration will unveil the symbiotic relationship between these simulation platforms, highlighting the seamless integration that underpins our research. This technological voyage not only reveals the capabilities of CARLA and SUMO but also illuminates their transformative potential for the future of autonomous systems and urban mobility.

3.1 CARLA

CARLA is a highly versatile and open-source simulator designed to support the development and validation of autonomous driving systems[13]. Rooted in its deep integration with the Unreal Engine, CARLA provides a platform where both urban and highway driving scenarios can be simulated with a rich set of features including a variety of environmental conditions, pedestrian behaviors, and realistic traffic scenarios[13].

3.1.1 OpenDRIVE and Environment Modelling

At the core of CARLA's environment creation is the OpenDRIVE format. This standard provides a detailed representation of road networks, encompassing elements such as lanes, intersections, and road types. Complementing this, CARLA integrates complex 3D models, ensuring a visually rich and immersive simulation environment that closely mirrors real-world settings[13].

3.1.2 Vehicular and Pedestrian Dynamics

A salient feature of CARLA is its commitment to realistic simulations. This realism extends beyond mere visual fidelity. The simulator provides extensive controls over vehicular dynamics, from simple car movements to intricate controls mimicking real-world physics. Furthermore, pedestrian behaviors can be tailored to match real-world scenarios, enhancing the depth and authenticity of simulations[13].

3.1.3 CARLA's Sensor Suite

CARLA is equipped with a wide array of sensors tailored to closely emulate real-world counterparts, making it invaluable for testing autonomous driving algorithms.

3.1.3.1 LIDAR

Light Detection and Ranging (LIDAR) is a remote sensing method utilizing laser light to measure distances to objects, creating detailed 3D representations of the environment[42].

- **Working Principle**: A LIDAR unit sends out a laser pulse and measures the time it takes for the pulse to return after reflecting off objects, thereby gauging distance.
- **Output**: Produces dense point cloud data, capturing the shape, distance, and reflectivity of surrounding objects.
- **Applications**: Vital for object detection, road profiling, and creating high-definition maps for autonomous navigation.

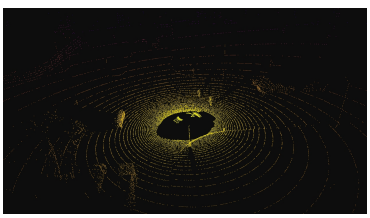


Figure 3.1: LIDAR visual representation

3.1.3.2 RADAR

RADAR (Radio Detection And Ranging) sensors employ radio waves to determine the distance, angle, and relative speed of objects.

- **Working Principle**: Sends out radio waves and measures the time taken for them to bounce back after hitting an object.
- **Output**: Returns data in polar coordinates, capturing the distance and velocity of detected objects.
- **Applications**: Commonly used for collision avoidance, adaptive cruise control, and blind-spot detection.



Figure 3.2: RADAR visual representation

3.1.3.3 Cameras

Cameras in CARLA are versatile tools, offering data in various formats, like RGB, depth, and semantic segmentation.

- **RGB Camera**: Captures scenes in red, green, and blue color channels, akin to human vision.
- **Depth Camera**: Measures the distance from the camera to objects for each pixel, aiding in gauging object proximity.
- **Semantic Segmentation Camera**: Provides a segmented view where each pixel is color-coded based on the object category (e.g., road, car, pedestrian).
- **Applications**: Essential for object recognition, lane detection, environment understanding, and depth perception.



Figure 3.3: RGB Camera output

3.1.3.4 Additional Sensors

Apart from the primary sensors, CARLA offers others like a lane invasion detector, Inertial Measurement Unit (IMU), and Global Navigation Satellite System (GNSS).

- **Lane invasion detector**: Registers an event each time its parent crosses a lane marking.
- **IMU (Inertial Measurement Unit)**: Offers data regarding vehicle acceleration, angular rate, and orientation.
- **GNSS (Global Navigation Satellite System)**: Similar to GPS but provides more global coverage, drawing data from multiple satellite systems.

3.1.4 Scalability and Reproducibility

CARLA stands out in its ability to run multiple simulations concurrently, allowing for extensive testing in a shorter time frame. Additionally, its focus on reproducibility ensures that developers can consistently recreate test scenarios, guaranteeing reliability in results.

3.1.5 Environmental Variability

Testing autonomous systems requires accounting for the unpredictability of real-world conditions. CARLA rises to this challenge by offering varied weather patterns (like rain or fog) and lighting conditions (like day or night), allowing for a holistic assessment of systems under various scenarios.

3.1.6 Integration and Extensibility

With its Python API, CARLA integrates seamlessly with leading machine learning libraries, promoting a fluid development process. Moreover, being open-source means developers can tweak its core to cater to specific requirements, be it adding sensors or creating new scenarios.

3.1.7 Traffic Management and Benchmarks

CARLA's Traffic Manager governs NPC movements, resulting in a dynamic, unpredictable yet realistic simulation. Additionally, the provided safety benchmarks serve as challenging test beds for autonomous systems, ensuring they can handle demanding situations.

3.1.8 Future Potential

Given its extensive features and continuous community-driven enhancements, CARLA is poised to remain at the forefront of autonomous system development. As more real-world driving challenges are identified, it's anticipated that the simulator will evolve, adding even richer features to its arsenal.

3.2 SUMO

SUMO is an esteemed open-source platform designed for simulating diverse traffic flow scenarios in urban environments. Renowned for its adaptability, scalability, and intricacy in traffic modeling, SUMO stands as a linchpin in urban mobility simulations[2].

3.2.1 XML-based Simulations

The core of SUMO's environment is rooted in XML files. These meticulously structured files offer a detailed description of elements such as road networks, vehicular types, traffic signals, and predefined vehicular routes. Their structured nature ensures that a wide range of traffic scenarios can be effectively modeled and simulated.

3.2.2 Route Generation and Management

One of SUMO's pivotal features is its robust route management system. It provides the flexibility of predefined paths or, if necessary, dynamic route generation. This flexibility guarantees that both generic and specialized traffic scenarios can be emulated with precision.

3.2.3 Traffic Control Interface (TraCI)

TraCI is an intrinsic tool when it comes to interacting and interfacing with running SUMO simulations. It operates as a TCP-based interface, allowing external programs to control, modify, and extract information from a running traffic simulation[2].

- **Real-Time Interaction**: One of the salient features of TraCI is its ability to facilitate real-time manipulation of the simulation. Users can dynamically introduce new vehicles, change traffic lights, or even adjust road attributes on the fly, offering a dynamic simulation experience.
- **Versatility in Programming**: TraCI supports various programming languages, making it a flexible tool for different projects. Whether using Python, Java, or even C++, developers can harness the power of TraCI to interact with SUMO.
- **Data Extraction**: Beyond controlling the simulation, TraCI is also instrumental in extracting data. Users can retrieve metrics like vehicle counts, speeds, waiting times, and many others, aiding in detailed analysis and optimization efforts.
- **Integration with External Systems**: Owing to its interface capabilities, TraCI can seamlessly integrate SUMO with other systems, making it particularly useful for complex projects that require communication between multiple platforms.

3.2.4 NETEDIT: A Graphical Network Editor for SUMO

NETEDIT stands as the official graphical network editor bundled with SUMO. It's designed to facilitate the creation, modification, and visualization of traffic networks.

- **User-Friendly Interface**: NETEDIT's intuitive Graphical User Interface (GUI) allows users to easily design and adjust road networks, junctions, traffic lights, and other network elements.
- **Validation Tools**: Integrated validation tools assist users in identifying and rectifying issues in the network, ensuring that simulations run smoothly.
- **Layer Management**: Users can seamlessly manage different layers, such as traffic signals, junctions, and pedestrian zones, allowing for intricate and detailed network designs.
- **Import and Export Capabilities**: NETEDIT supports importing various formats, like OpenStreetMap, and can export the designed networks into SUMO-compatible XML files.

3.2.4.1 Demand Creation in NETEDIT

NETEDIT is not only designed for network editing but also for demand modeling, a critical component of traffic simulations. This feature makes it particularly valuable for researchers and

professionals who wish to understand how traffic demand affects road networks under various scenarios.

- **Demand Mode**: One of the standout features of NETEDIT is its 'Demand Mode'. This mode allows users to define specific routes and traffic flows across the network.
- **Vehicle Flows**: Within Demand Mode, users can create individual vehicle routes or flows of multiple vehicles. These flows can be time-dependent, allowing for simulations that account for peak traffic hours, for instance.
- **Public Transport**: NETEDIT also caters to public transportation demand. Users can define bus and tram routes, stops, and schedules, offering a comprehensive view of how public transport interacts with private vehicle traffic.
- **Pedestrian and Bicyclist Flows**: Beyond motorized transport, NETEDIT also enables the creation of pedestrian and bicyclist flows, ensuring a holistic simulation of urban mobility.
- **Dynamic Demand Changes**: Another advanced feature of NETEDIT's demand modeling is the ability to introduce dynamic changes to traffic demand, such as sudden surges due to events or blockages.
- **Intuitive Interface**: NETEDIT's graphical interface allows users to visually create and adjust routes, making the process of demand modeling more intuitive. This visual approach is particularly useful when designing complex scenarios with multiple overlapping routes and flows.

In essence, NETEDIT's demand creation capabilities ensure that the generated traffic scenarios in SUMO are both realistic and aligned with the research or planning objectives. Given its centrality to our project, leveraging these features was critical to achieving meaningful and actionable simulation results.

3.3 Co-simulation: Integrating CARLA and SUMO

To harness the complementary capabilities of both CARLA and SUMO, integrating them seamlessly becomes an imperative task. While CARLA brings to the table a detailed 3D simulation environment with a plethora of sensors and real-world visualization, SUMO provides meticulous traffic simulations and a broad array of traffic modeling capabilities.

3.3.1 Map Compatibility

A primary concern in co-simulation is map compatibility. This involves the translation of CARLA's OpenDRIVE maps into SUMO-compatible XML files. Achieving this ensures that

the environment and road structures represented in both simulators are consistent, which is fundamental for a synchronized co-simulation.CARLA has been designed with a vision to be extensible and interoperable.

3.3.2 Synchronization

The essence of a successful co-simulation lies in its ability to seamlessly synchronize diverse simulation platforms, thereby ensuring data integrity, event coordination, and temporal alignment. Recognizing this pivotal aspect, CARLA presents a robust framework for native SUMO integration.

CARLA's integration framework is more than just a mere interface between the two simulators. It embeds a meticulous synchronization mechanism, which ensures that entities like vehicles, when spawned in SUMO, are correspondingly and coherently manifested in the CARLA environment. This bidirectional communication ensures that actions and events in one simulator are accurately reflected in the other.

This synchronization is underpinned by a dedicated set of Python classes that CARLA provides. These classes abstract away the complexities of data mapping, event coordination, and time-step matching. For the end-user, this translates to a streamlined experience wherein vehicles and traffic conditions set in SUMO can be visualized and interacted with in CARLA's rich 3D environment without any disjointed behavior.

Furthermore, this synchronization capability extends not just to vehicle spawning, but also to the intricate behaviors, trajectories, and interactions that vehicles exhibit within the simulation space. This ensures that traffic scenarios, from simple vehicle movements to complex multi-vehicle interactions, are faithfully represented across both simulation platforms.

3.4 CARLA and SUMO in the Context of Smart Cities, Digital Twins, and Autonomous Vehicles

The urban landscape is continuously evolving, with technology being at the forefront of this transformation. The vision of Smart Cities, where infrastructure and technology converge to create efficient, sustainable, and intelligent urban environments, has become central to urban development agendas. Within this paradigm, the role of digital twins and autonomous vehicles becomes indispensable.

CARLA and SUMO, as discussed throughout this work, can be perceived as tools to create such digital twins of urban environments. Their combined capabilities offer a comprehensive simulation environment that reflects real-world traffic scenarios and infrastructure with high fidelity.

We'll now instantiate some approaches made in order to explore the visualization of models and retrieve the key points that will serve as a reminder when starting our development phase.

3.5 mySMARTLife Project for Helsinki's 3D City Model

The **mySMARTLife** project in Helsinki encompasses the development of an Energy and Climate Atlas as part of a 3D city model[30]. The project is built upon a strong foundation of real and simulated data, geared towards enhancing the city's energy efficiency and citizen engagement[30] as we can observe in the image below 3.4 the project aims to make the city more environmental friendly reducing the emissions of CO2.



Figure 3.4: mySMARTLife project overview

3.5.1 Initial Outcomes

The Energy and Climate Atlas, launched in 2018, houses an array of critical information about Helsinki's buildings. This includes data on energy-efficiency upgrades, performance classifications, and energy sources. Its creation was backed by the Technical Research Centre of Finland VTT, ensuring accurate energy consumption estimations for various structures[30].

3.5.2 Integration of Real-time Data

One of the hallmark features of the atlas is its plan to incorporate real-time data for a vast number of buildings. The city's Kalasatama district stands out, mandating that structures have

APIs to facilitate this. The chosen protocol for energy data, CIM, is significant in the realm of smart grids, providing a comprehensive framework for automatic meter reading and demand management[30].

3.5.3 Evolution of the 3D Model

The 3D city model has two developmental trajectories: enhancing visual textures and integrating new data sources. This dual approach results in a product that is rich both in appearance and in content. Semantically, the use of the application domain extension (ADE) allows for the incorporation of novel objects and attributes, further expanding the model's utility.

3.5.4 Digital Twin and User Engagement

A subsequent developmental phase, termed the Digital Twin, zeroes in on the Kalasatama District. This rendition strives for a perfect blend of actual and proposed urban structures, inviting users to engage with and shape their city's future. The use of tools like CityPlanner by Agency9 facilitates this, offering a dynamic platform for collaborative city planning and visualization.

3.5.5 Enhancements to Sensor Data Collection

The project's initial stages revealed gaps in the SensorThings API[1] documentation, specifically in handling sensor calibration information. This insight underscores the necessity for a universally accepted standard in managing such data. Nevertheless, the groundwork laid by the project offers a promising foundation for future iterations.

3.5.6 Prioritizing Data Privacy

In an age of increasing data concerns, the project has embedded General Data Protection Regulation (GDPR) compliance at its core. The creation of a privacy proxy is notable, giving data owners greater control over their information. This foresight not only ensures regulatory compliance but also fosters trust among Helsinki's citizens.

3.6 An Integrated Traffic and Vehicle Co-Simulation Testing Framework for Connected and Autonomous Vehicles

In the presented work, a comprehensive co-simulation framework is introduced, leveraging both the SUMO and CARLA. This integration not only combines vehicle dynamics and AVs

[1]The SensorThings API is an open standard for connecting the Internet of Things (IoT) devices and data

models but also factors in advanced micro traffic simulations for a more intricate traffic flow analysis[33]. Essential to this process is the acquisition of genuine and representative data. Thus, the incorporation of real-world factors pertinent to the vehicles, their sensors, and the encompassing environment has been prioritized. To achieve this, a synthesis of deep learning algorithms and video data has been employed, culminating in a holistic input-model-output co-simulation testing system for autonomous driving. This article provides some really good base knowledge for our start to comprehend the basic concepts of how the co-simulation would work.

3.6.1 Emphasis on Real-world Data

"The usage of virtual data alone to explain real-world traffic scenarios is challenging because of the complexity of the traffic dynamics" [33]. Relying solely on virtual data to portray the multifaceted dynamics of real-world traffic is an overly ambitious approach. The autonomous driving system's testing, prioritizing the safety and efficiency of connected and autonomous vehicles (CAVs), begs for real-world scenarios. Harnessing video data, an underappreciated reservoir of genuine traffic information, for simulation input is an insightful approach, albeit with challenges to overcome[33].

3.6.2 Trajectory Extraction: Easier Said Than Done

While fixed-view monitoring is hailed for its efficiency in capturing traffic operations, it's not without pitfalls. The prerequisite of camera calibration, demanding the transformation between real-world and pixel coordinates, is cumbersome. Despite its promises, this method can be daunting given the complexity of incorporating various coordinate systems.

An even more pressing concern is the lens distortion and the lack of prominent reference objects. The novel idea of using the distances between lane markings is commendable, attempting to alleviate these challenges, though its effectiveness remains to be tested extensively.

3.6.3 Vehicle Detection and Tracking: Leaning on Advanced Techniques

Opting for the YOLO[2] and DeepSORT algorithms showcase a step in the right direction, given their robustness in detection and tracking. The utilization of the calibrated model to convert pixel to world coordinates might seem straightforward but is a masterstroke, allowing speed calculations and trajectory enhancements[33].

[2]YOLO is an abbreviation for the term 'You Only Look Once'. This is an algorithm that detects and recognizes various objects in a picture (in real-time).[21]

3.6.4 Flight Perspective: A New Angle

The UAV's flight perspective offers an innovative way to detect vehicles. Yet, the reliance on the Canny edge detection technique, though accurate, might not be the most efficient solution. The correlation filter tracker, although prevalent, might not always guarantee precise tracking between frames[33].

The flight perspective's real challenge is the difficulty in camera calibration due to motion and insufficient reference objects. However, using traffic volume statistics as an ingenious workaround, by establishing a virtual loop vehicle detector, is noteworthy[33].

Traditional GPS or cross-sectional data-focused research might seem outdated when juxtaposed with this approach that centers on traffic surveillance video. The deep learning algorithms employed, combined with trajectory extraction using fixed-point videos, showcase innovation. The calibration using lane-line segment distances is a breakthrough, though its long-term feasibility is up for debate. Nevertheless, this novel framework for obtaining traffic flow information, especially from a flight perspective video, is a breath of fresh air in the domain of traffic analysis[33].

We are subsequently introduced to the amalgamation of SUMO and CARLA. This exploration underscores the strengths inherent to each tool, elucidating why their integration is pivotal for the co-simulation. Both tools, in their own right, offer unique capabilities. The coalescence of which promises to deliver a comprehensive simulation framework, ensuring that the critical aspects of each are not only maintained but also enhanced in the combined setting[33].

3.6.5 Network Generation in SUMO

The granularity of SUMO's road network structure and its reliance on Open Street Map (OSM) underline its intent to ensure accuracy in traffic simulations. The choice between automatic and manual road network generation methods signifies flexibility, catering to broad and specific research needs respectively.

3.6.6 Demand Modeling in SUMO

The multi-faceted approach to traffic demand modeling in SUMO showcases its adaptability in replicating diverse real-world scenarios. The pivotal role of TraCI, as an interface that bridges real-world data with the simulated environment, accentuates its significance in creating a genuine, dynamic simulation[33].

3.6.7 Integration of SUMO and CARLA

The integration is notable for its ambition to merge macroscopic traffic scenarios with intricate vehicle dynamics. The methodology, from transforming 2D road networks to 3D to syncing

SUMO and CARLA, reflects a commitment to detail and realism. The consistent use of TraCI is a testament to its versatility in managing interactions between different simulation paradigms.

3.6.7.1 Key Points

1. **Detailed Vehicle Models:** Vehicle dynamics simulations provide granular models that capture the kinematics of axles, steering models, and overall driving dynamics.
2. **CARLA:** An open-source driving simulator with sensor models offering extensibility and a flexible Python API for integration with SUMO.
3. **Road Network Construction:** 3D high-precision road networks are created from 2D simulation road networks using real-world traffic data. RoadRunner, a 3D scene editor, converts 2D to 3D based on elevation data and appends elements such as traffic signs, buildings, etc. The resultant OpenDRIVE and Filmbox files form the base model in CARLA[33].
4. **Synchronization:** Elemental information of SUMO and CARLA must align for effective co-simulation. CARLA's blueprints, which dictate elements like vehicle attributes and lidar sensor channels, are then matched with SUMO data.
5. **Communication:** The TraCI protocol ensures that CARLA and SUMO communicate effectively.
6. **Co-Simulation Execution:** SUMO handles the simulation of all vehicles, including the main CAV tested by CARLA. CARLA retrieves the speed, position, and steering angle of the test vehicle and updates these attributes in SUMO using TraCI. Consistent simulation step sizes across both tools are crucial.
7. **Realism and Interaction:** The vehicle dynamic model and sensor setup are virtual replicas of real test vehicles. Interactions among sensor models and CARLA's virtual AV environment occur dynamically. SUMO generates diverse traffic scenarios to test CAVs.
8. **Conclusion:** The framework introduces an integrated 3D vehicle dynamic simulator with realistic road environments and traditional traffic models. It also supports vehicle generation in both SUMO (traffic demand-based) and CARLA (with detailed dynamics).

3.6.8 Case studies

The presented work meticulously investigates the integration of traffic and vehicle dynamics simulations. The cornerstone of this endeavor is the co-simulation testing framework, uniquely capable of fabricating traffic scenes rooted in authentic data. This fusion not only enables the co-existence of CAVs with traffic elements but also fosters the emergence of spontaneous, yet realistic, traffic events like sudden lane changes and congestion[33].

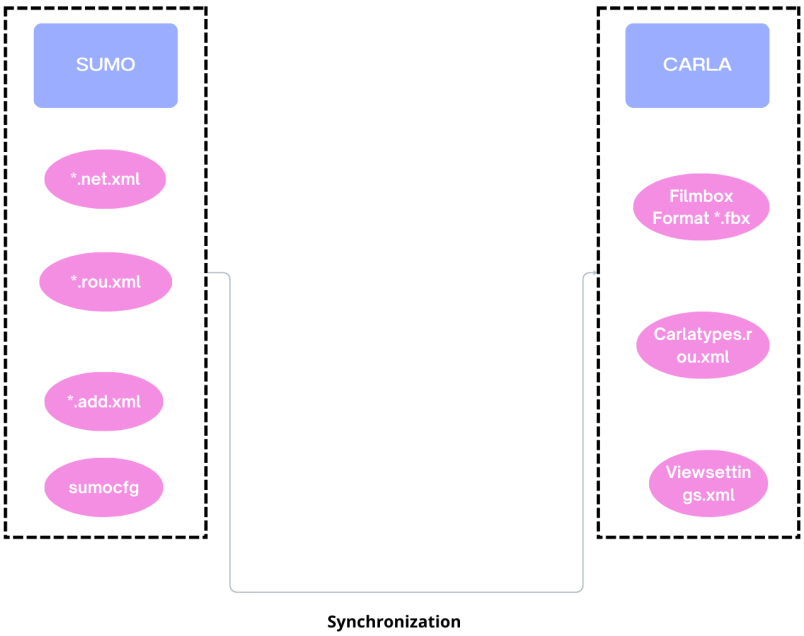


Figure 3.5: Overview of co-simulation utilized

Turning the lens to practical implementations, the study illuminates the usage of highway surveillance footage, specifically from the G25 segment of the Shanghai–Hangzhou Highway. Interestingly, while the videos showcase routine traffic, anomalies arise due to foreign objects, furthering the richness of the simulation. The utilization of deep learning via the YOLO network, paired with the DeepSort algorithm, plays a pivotal role in the detection and tracking of vehicles[33].

A paramount challenge is the calibration of camera parameters, given the absence of geo-referenced data in the highway's fixed-point footage. However, the study ingeniously exploits lane-line data, standardized in China, to circumvent this. The precision of this calibration technique is commendably validated with minimal discrepancies between computed and actual distances[33].

Post calibration, trajectory data undergoes transformation to the WGS84[3] coordinate system, adopting either the Bursa seven-parameter model for larger scale or the plane four-parameter model based on the spatial scale for a smaller scale. The concurrent display of SUMO and CARLA simulations then takes center stage, reflecting authentic vehicle movements and interactions with obstacles. Intriguingly, these simulations also pave the way for testing autonomous driving algorithms, presenting real-world challenges like abrupt halts due to obstacles.

The paper's thorough examination of the utility of video-based trajectory data, contrasted with varying lateral resolutions in SUMO, is commendable."Video data serves as a rich and widely

[3]The WGS84, or World Geodetic System 1984, is a standard used in cartography, geodesy, and satellite navigation including GPS (Global Positioning System)

available data source to obtain real-world traffic information that can be used as simulation input"[33].The analysis uncovers tangible differences between simulated and real-world vehicle reactions to obstacles. To quantify the fidelity of the simulation, micro-indicators like sharp decelerations and lane changes are distilled, affirming the relative accuracy of the co-simulation approach.

The researchers considered aerial videos taken from a UAV at an intersection in Wuxi, Jiangsu Province. Due to the repetitive nature of traffic at the intersection, a unique vehicle counting method was devised. The steps involved:

- Detecting vehicle edges using the Canny method and then consolidating those edges to depict a whole vehicle.
- Using a correlation filter tracker for vehicle movement tracking.
- Virtual loop vehicle detectors (akin to the in-ground sensors at intersections) are emulated to count vehicles.
- By tracking how often vehicles cross these virtual loops, they calculate traffic flow rates, which are then fed into a simulation.

Regarding the coordinate system, 24 coordinates were picked, achieving a fairly precise transformation with a minuscule margin of error.

By using SUMO, they could adjust various parameters like traffic signals to recreate different traffic scenarios. This approach not only models traffic behavior but also serves as a testbed for autonomous driving systems via CARLA[33].

3.6.9 A Highway Scenario for CAVs

The paper further explores the co-simulation framework's applicability in a highway scenario. An autonomous vehicle (AV) replaces a human-driven one in a test. This AV uses CARLA's automatic driving planning algorithm, sensing surrounding vehicles and plotting its course in real-time.

The experiment results depict the AV recognizing an obstacle and decelerating, but due to an immature algorithm, it halts instead of changing lanes. This showcased the framework's capacity to test and optimize AV driving algorithms.

Another experiment centered on information sharing between connected and autonomous vehicles (CAVs). Two CAVs on the highway shared information regarding their status and surrounding conditions. When the lead vehicle encounters an obstacle, the trailing CAV, informed of the situation, successfully changes lanes and overtakes, verifying the framework's suitability for CAVs.

This highlights the potential of vehicle-to-vehicle (V2V) communication, where vehicles can preemptively adjust based on information from other vehicles.

3.7 A Novel Traffic Simulation Framework for Testing Autonomous Vehicles Using SUMO and CARLA

Another really good system using CARLA and SUMO is the one introduced by Pei Li, Arpan Kusari and David J. LeBlanc from the University of Michigan Transportation Research Institute. The paper introduces an innovative traffic simulation framework that leverages both CARLA and SUMO for the rigorous testing and validation of AVs. This is particularly significant given the enormous variability in real-world driving scenarios that production-ready AVs may encounter. The paper does an excellent job addressing the limitations of current simulation software by integrating naturalistic data for a more realistic representation of background vehicle behaviors. Additionally, it presents a divide-and-conquer testing strategy which is valuable for developers to identify and rectify subsystem errors efficiently using OpenAI Gym[23].

1. **OpenAI Gym:** An open-source Python library designed for testing reinforcement learning algorithms in diverse simulation environments.
2. **SUMO (Simulation of Urban Mobility):** A microscopic simulator developed for modeling and predicting traffic flow.
3. **CARLA:** A high-fidelity simulator offering features like an in-built sensor suite and various weather conditions simulations.
4. **CARLA-SUMO-Gym:** This framework combines the strengths of the above platforms, aiming to deliver a holistic solution for AV simulation.

3.7.1 Object Detection Algorithm

The benchmark object detection model used in the study is based on Facebook's detectron2.Detectron2 is a powerful library that offers a variety of state-of-the-art models for different image processing tasks, including object detection[23]. The chosen model was pretrained on the COCO dataset, consisting of 330K images across 80 object categories. This pretrained model provides a solid foundation for detecting objects in various scenarios[23].

3.7.1.1 Simulation Setup

The object detection algorithm was tested under two simulated weather conditions: morning and night. The scenarios were designed in CARLA, incorporating realistic lighting conditions to mimic the challenges autonomous vehicles might face in the real world.



Figure 3.6: The morning vs the night scenario

3.7.1.2 Evaluation Metrics

The performance of the object detection model was evaluated using precision and recall metrics. These metrics were computed based on Intersection over Union (IoU) values, which measure the overlap between predicted bounding boxes and ground truth.

3.7.2 Case Study and Results

A case study was performed to assess a benchmark object detection model under two simulated weather conditions using CARLA: morning and night. The results were analyzed using precision and recall metrics, which are determined based on Intersection over Union (IoU) values of detected objects.

The findings include:

- **Morning Scenario:**
  - Precision: 0.82 (IoU 0.5) to 0.47 (IoU 0.8)
  - Recall: 0.41 (IoU 0.5) to 0.28 (IoU 0.8)
- **Night Scenario:**
  - Precision: 0.81 (IoU 0.5) to 0.37 (IoU 0.8)
  - Recall: 0.23 (IoU 0.5) to 0.12 (IoU 0.8)

3.8 OpenCDA: A framework for Cooperative Driving Automation

OpenCDA has emerged as a pioneering solution in the domain of Cooperative Driving Automation (CDA) research and development. Addressing the existing gaps, it stands out as an intuitive platform that boasts a full-stack CDA software, encompassing critical components like perception, communication, planning, and control.

3.8.1 Key Features of OpenCDA

OpenCDA is characterized by its six foundational features[36]:

1. **Connectivity and Cooperation**: This feature ensures seamless interaction between connected automated vehicles, promoting cooperative behaviors. This can potentially lead to enhanced road safety and traffic efficiency.
2. **Integration**: Through the integration of CARLA and SUMO, OpenCDA offers a co-simulated environment. CARLA provides high-fidelity visualization and sensor simulations, while SUMO furnishes granular traffic scenarios. This merger creates a holistic and realistic testing ground for CDA algorithms.
3. **Full-stack System**: OpenCDA's comprehensive system covers all essential domains of CDA research – from perception and communication to planning and communication to planning and control. This feature ensures a streamlined workflow, where each segment is closely interlinked and operates harmoniously.
4. **Modularity**: One of OpenCDA's primary strengths is its modular design. Researchers can easily plug in their own algorithms or swap out individual components without disrupting the entire system. This facilitates iterative development and testing, allowing for quicker adaptations and innovations.
5. **Benchmark**: OpenCDA comes equipped with benchmark scenarios and data sets. These standardized tests enable researchers to compare and evaluate the performance of different CDA algorithms objectively. Such benchmarks provide clarity on algorithm robustness, efficiency, and safety.

These pillars form a comprehensive system, underpinned by a robust architecture, well-structured simulation flow, diverse testing scenarios, and a coherent software design.

3.8.2 Emphasis on Co-Simulation: SUMO and CARLA

Co-simulation is at the heart of OpenCDA's versatile design. Integrating both **SUMO** and **CARLA** tools, OpenCDA offers a hybrid environment for rigorous testing and development.

3.8.2.1 SUMO (Simulation of Urban Mobility)

They introduce the key points about sumo like:

- Provides detailed network importing and traffic demand modeling.
- Models a wide range of vehicle types, including passenger cars, buses, and pedestrians.

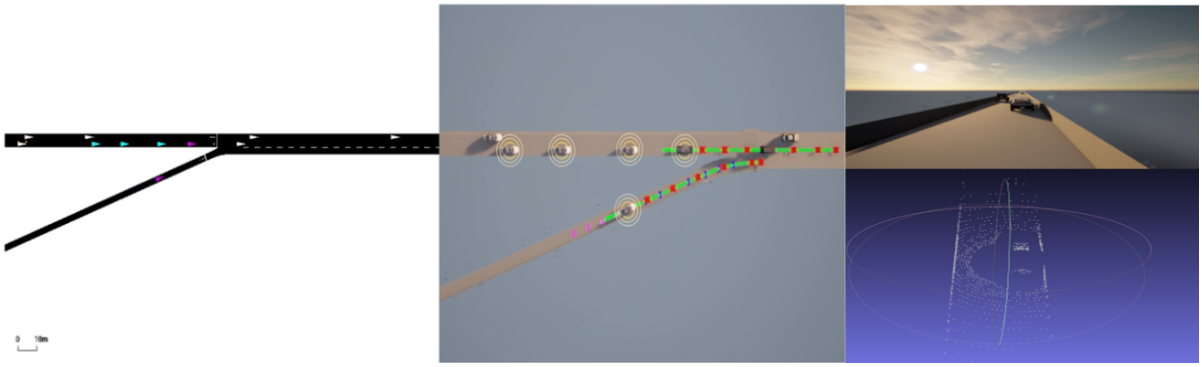


Figure 3.7: Platooning scenario testing under co-simulation settings[36]

- Supports traffic-light and public transport scheduling.

Within OpenCDA, SUMO plays a pivotal role in generating realistic traffic patterns and vehicular interactions, providing a credible backdrop against which CDA algorithms can be tested.

3.8.2.2 CARLA

CARLA stands out as a top-tier open-source simulator for autonomous driving research. The core strengths include:

- Rendering realistic urban environments with dynamic weather conditions.
- Simulating a plethora of sensors, including cameras, lidar, and radar.
- Providing APIs for custom scenario definitions and autonomous driving algorithms.

In the context of OpenCDA, CARLA is instrumental in offering a rich visual and sensory environment, allowing CDA algorithms to be tested in conditions mirroring real-world scenarios.

By synergizing SUMO's traffic simulation capabilities with CARLA's environment rendering, OpenCDA achieves high-fidelity co-simulations, paving the way for comprehensive CDA algorithm testing.

3.8.3 Practical Application: Platooning

OpenCDA's prowess is evident in its application to platooning scenarios. The modular pipeline efficiently manages CDA applications, with the system's inherent flexibility making it a fit for various custom solutions[36].

3.9 Framework for Interactive M3 Visualization of Microscopic Traffic Simulation

Visualization is an essential aspect of traffic simulations, helping professionals analyze microscopic traffic simulation models. The following project discusses how a 3D virtual environment enhances these visualization efforts[37].

The realm of traffic simulation has witnessed significant technological advancements, including the integration of visualization tools. This analysis dives into a project that aims to enhance the visualization of microscopic traffic simulation models (MTS) in a 3D virtual environment.

3.9.1 Background: Paramics

Paramics is a traffic simulation software, providing detailed models of traffic flows. In the context of this project, it plays a crucial role as it integrates with the virtual environment (VE) to provide real-time traffic signal updates. Understanding Paramics is essential because it forms the backbone of the simulated traffic data.

3.9.2 Virtual Environment and Signal Control

In real-world traffic systems, the flow is majorly regulated by traffic signals. Translating this into a VE, signal timings from Paramics are linked to control traffic lights. This ensures a realistic traffic flow, adhering strictly to the signal regulations in the VE.

3.9.3 Distributed Multi-User Visualization

A highlight of the project is its distributed client-server framework as ilustrated in 3.8. The system was tested by transforming Paramics into an MTS server, operating with TCP/IP protocols. This approach allows multiple clients (users) to connect to a centralized simulation server, each exploring different traffic scenarios. Such a distributed system has broad implications, especially for collaborative projects or public outreach where multiple stakeholders need to simultaneously view or interact with traffic models.

3.9.4 Versatility in User Perspective

Beyond the conventional zoom and pan, users can explore traffic from varied perspectives: bird's-eye view, pedestrian view, car follower view, and driver view. Each offers a unique vantage point, granting users a comprehensive understanding of traffic dynamics as shown in 3.9.

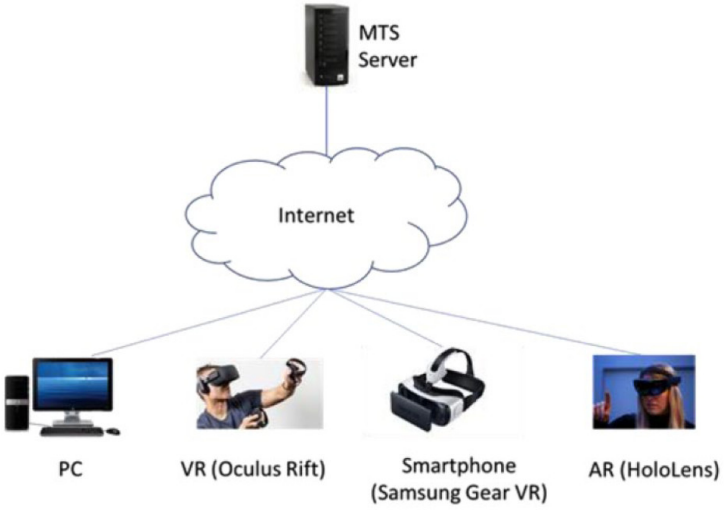


Figure 3.8: Visualization system overview[37]



Figure 3.9: Multiple views created: (a) menu, (b) bird's-eye, (c) pedestrian, (d) car follow and (e) driver)[37]

3.9.5 Innovative Multi-Mode Visualization

One groundbreaking achievement is the adaptation of the visualization for both Virtual Reality (VR) and Augmented Reality (AR) platforms. This means users can immerse themselves in the traffic scenarios, experiencing them firsthand. However, this adaptation wasn't without challenges, especially regarding user-interface changes, given the inherent differences between desktop and VR/AR interactions.

3.10 Conclusion

The projects discussed showcase the diverse applications and innovations in the field of smart cities, autonomous vehicles, cooperative driving automation, and traffic simulation visualization. Each project contributes significantly to its respective domain, addressing challenges and pushing the boundaries of technology. The mySMARTLife project in Helsinki sets a gold standard for smart city endeavors by leveraging cutting-edge technology for urban development while prioritizing citizen engagement and data privacy and introducing the importance of decreasing C02 emissions that we will take into note in our project. The Integrated Traffic and Vehicle Co-Simulation Testing Framework demonstrates the importance of real-world data in testing autonomous vehicles, emphasizing the need for a comprehensive co-simulation approach introducing for the first time for us a SUMO and CARLA co-simulation Different scenarios were crafted to validate the flexibility and scalability of the proposed framework. The paper emphasizes the nuanced differences between purely model-driven scenarios and those built upon real-world data. Particularly, it accentuates the value of periodic, signal-controlled intersections for autonomous algorithm testing. The case studies presented in the paper reaffirm the capability of the proposed framework to offer a comprehensive testing methodology for CAV algorithms[33].The article introduces a robust and adaptable co-simulation framework, leveraging aerial video data in tandem with established simulation tools such as SUMO and CARLA. By replicating real-world situations and integrating autonomous driving elements, it offers an essential foundation for enhancing and evaluating autonomous and connected vehicle algorithms. This research serves as a pivotal reference, highlighting the intricacies of how these two simulators interact, and offers vital insights for the development of our project. The Novel Traffic Simulation Framework for Testing Autonomous Vehicles using SUMO and CARLA introduces a valuable testing methodology, incorporating deep learning algorithms and video data for realistic simulation scenarios. The benchmark object detection model's performance under varying conditions emphasized the significance of incorporating multiple sensors, especially in challenging weather conditions as we will explore too more into depth on about how CARLA different scenarious affect or not SUMO and their compatibilities. OpenCDA stands out as a comprehensive framework for Cooperative Driving Automation, providing a full-stack system with modularity and benchmarking capabilities. Finally, the Framework for Interactive M3 Visualization of Microscopic Traffic Simulation enhances the visualization of traffic models in a 3D virtual environment, offering distributed multi-user visualization and innovative multi-mode perspectives which introduces us to a more user interactive project with the user and how important it is the immersive experience.

These projects collectively contribute to the ongoing evolution of smart technologies, paving the way for more sustainable, efficient, and connected urban environments. As technology continues to advance, the integration of these innovative solutions will play a crucial role in shaping the future of transportation and urban living. he analysis of various projects reveals the diverse applications and innovations within the realm of smart cities, autonomous vehicles, cooperative driving automation, and traffic simulation. Each project brings unique insights and advancements to its respective domain.

In reviewing the state of the art, we've explored a range of methodologies, tools, and innovations that are instrumental in the domain of autonomous vehicle simulations, traffic enhancements, and different types of visualization. While many of these approaches have brought significant advancements, gaps, and opportunities for further innovation remain evident. The comprehension of these existing methods not only provides a solid foundation but also underpins the motivation for the research presented in this work. As we transition from examining what has been accomplished to what can be achieved, the subsequent development chapter will detail the design, implementation, and evaluation of our proposed solution, building upon the insights gained from this review.

Chapter 4

Objectives and System Architecture

Urban traffic simulation is a complex endeavor that demands a holistic approach, considering dynamic interactions between vehicles and traffic infrastructure. This chapter explores the foundational objectives and intricacies of our co-simulation environment, merging the capabilities of Car Learning to Act (CARLA) and Simulation of Urban Mobility (SUMO). Our focus has been on creating a versatile and immersive platform for studying real-world traffic scenarios, enhancing the realism of simulations through vehicle detection mechanisms, adaptive traffic light management, dynamic weather conditions, and user-friendly modifiability.

The chapter unfolds with a detailed examination of our key objectives, illustrating the specific achievements and milestones expected to reach in each area. From there, we navigate the intricacies of the system architecture, dissecting the components and interactions that synergize to create a dynamic and responsive simulation environment. The discussion extends to future directions, exploring avenues for integration with machine learning, autonomous vehicle scenarios, scalability improvements, enhanced visualization, and community engagement.

Through this exploration, we aim to provide not only a comprehensive understanding of our co-simulation environment but also insights into its potential applications, adaptability, and the exciting possibilities it opens for future research and development.

4.1 Objectives

- Create a co-simulation environment that mirrors real-world traffic scenarios merging both SUMO and CARLA and explore into their compatabilities and what can possible be done. Considering factors such as traffic demand, diverse vehicle behaviors with a look into their impact in Co2 emissions and overall traffic congestion.
- Implement a robust vehicle detection mechanism using Light Detection And Ranging (LIDAR) sensors to enhance the realism of the simulation and provide accurate data for analysis.

- Explore into adaptive traffic light management that responds to real-time traffic conditions, reducing congestion and improving overall traffic throughput.
- Enable the simulation of diverse weather conditions to study the impact of weather on autonomous driving systems and traffic behavior.
- Create a user-friendly interaction with the API, allowing researchers to modify simulation parameters and behavioral conditions on the fly.

4.2 System Architecture Review

In this section, we delve into a comprehensive review of the system architecture, dissecting the pivotal components that synergize to create a seamless co-simulation environment. The integration of CARLA, SUMO, NETEDIT, and Traffic Control Interface (TraCI) forms the backbone of our system, each playing a crucial role in achieving our overarching objectives.

4.2.1 CARLA: Realism in Vehicle Dynamics

At the core of our co-simulation environment lies CARLA, a powerful simulator renowned for its realistic rendering of vehicle dynamics and sensor interactions. CARLA serves as the visual representation of the traffic scenarios, offering a high-fidelity environment for testing and validating autonomous vehicle algorithms. Vehicles, pedestrians, and environmental conditions are meticulously simulated, contributing to the immersive nature of our platform. Sensor such as LIDAR and RADAR are as well are retrieving information and are a key factor to our validation step.

4.2.2 SUMO: Traffic Dynamics and Light Control

Complementing CARLA, SUMO takes the helm in simulating traffic dynamics, generating vehicular flow, and managing traffic behaviors. It acts as the traffic backend, handling the generation of vehicles, defining traffic patterns, and managing traffic light states. Additionally, SUMO communicates with CARLA through the Traffic Control Interface (TraCI), ensuring synchronization between the visual and behavioral aspects of the simulation.

4.2.3 NETEDIT: Configuring Demand Patterns

NETEDIT plays a pivotal role in shaping the demand patterns within our co-simulation environment. This tool allows us to configure and fine-tune the network definition, influencing traffic generation and flow that guides traffic dynamics during the simulation.

4.2.4 TraCI: Synchronization and Interaction

The Traffic Control Interface (TraCI) acts as the bridge between CARLA and SUMO, facilitating seamless communication and synchronization between the visual representation and the underlying traffic dynamics. TraCI ensures that changes in traffic states or light control made in SUMO are reflected in CARLA, and vice versa. This bidirectional interaction is fundamental to the cohesive functioning of our co-simulation environment.

4.2.5 Dynamic Interplay

The dynamic interplay between these components forms the crux of our architecture. CARLA provides the immersive visual canvas and sensors data, SUMO governs traffic behaviors, NETEDIT configures demand patterns, and TraCI ensures synchronization. Together, they create a holistic environment that mirrors real-world traffic scenarios, setting the stage for our comprehensive set of objectives.

The collaboration between CARLA and SUMO for co-simulation demands an intimate grasp of the respective input requisites and configurations of both systems. This section delineates pivotal considerations imperative for achieving an integrative, accurate, and cohesive simulation experience.

4.2.6 Simulation Input Specifications

4.2.6.1 CARLA: Painting Realism with 3D

CARLA employs the OpenDRIVE format (.xodr) to sculpt intricate road architectures. Additionally, CARLA utilizes the `viewsettings.xml` for visual configurations.

1. **Map Selection:** The simulation's foundation rests on the chosen map, serving as its canvas. The map must echo reality for vehicular challenges, be visually harmonized, and critically compatible with both simulation environments.

2. **Dynamism through Entities:** Infrastructure sets the ambiance, but it's the vehicles and pedestrians that instill life. CARLA's rich repository furnishes an array of entity behaviors and aesthetics, mirroring the vibrant and unpredictable nature of real-world traffic.

4.2.6.2 SUMO: Precision with XML Blueprints

SUMO's essence lies in its dependency on XML configurations. These XML constructs are holistic, encompassing everything from the minutiae of road networks to the dynamic narratives of vehicle routes.

1. **XML Network Files:** These are SUMO's backbone, meticulously laying out the roadmap: each segment, junction, and signal. It's the infrastructure blueprint for SUMO's symphony.
2. **Guiding with Route Files:** These files are the choreographers, setting the stage and directing each simulation entity, ensuring interactions are orchestrated.
3. **Configuration with sumocfg:** SUMO uses the `sumocfg` for overall configuration, integrating various input files and defining simulation parameters.
4. **Additional Settings with add.xml:** An optional file, the `add.xml` allows for fine-tuning and integration of supplementary features or adjustments.

A general overview of the system input is described in the image below 4.1

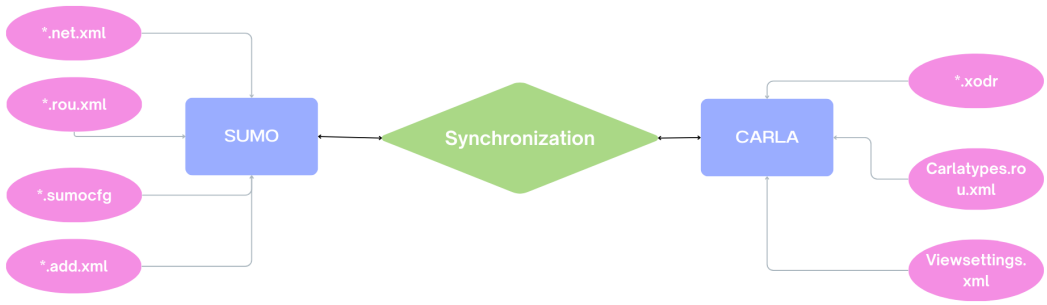


Figure 4.1: System input overview

4.2.7 Harmonizing Map Representations

The fusion of CARLA's OpenDRIVE maps with SUMO's XML preference isn't a mere format translation. It's an intricate transformation ensuring that every element retains its meaning and function across both simulative terrains.

4.2.8 Co-Simulation Choreography

1. **Synchronization:** The linchpin of co-simulation. This isn't just about time-step synchrony but about ensuring that concurrent actions in both simulations resonate in harmony.

CARLA already provides a co-simulation module, which serves as our synchronization foundation.

2. **Behavioral Uniformity:** Behavioral congruence across both simulations isn't just desirable; it's imperative. It ensures consistent simulation outcomes and is achieved via meticulous parameter tuning in both CARLA and SUMO.

3. **Demand Genesis:** Realistic vehicular flow demand is birthed through a collaborative effort between CARLA and SUMO, utilizing the unique demand-generation attributes of both tools.

4.2.9 System Overview

The intricate dance between CARLA and SUMO unfolds in a harmonious symphony, orchestrated by TraCI. The system's input specifications, harmonizing map representations, co-simulation choreography, and output integration culminate in a dynamic and responsive simulation environment. This architectural overview sets the stage for a detailed exploration of our objectives, where we dissect how this collaborative ecosystem addresses real-world traffic scenarios and contributes to advancements in autonomous driving research.

The Application programming interface (API) developed for the integration of CARLA and SUMO is implemented in Python and uses Flask for web server functionalities and TraCI for interacting with the SUMO simulation and the CARLA Python API to comunicate with CARLA.

The API serves as our central hub connecting CARLA and SUMO, bridging their respective capabilities to achieve the desired co-simulation. Its primary role is to facilitate real-time communication, coordinate state updates, and manage synchronization between the two simulation environments.

To be up to the excelence of the dynamic nature of simulations, our API is designed to handle concurrent access, which is vital for ensuring simultaneous requests from simulataneous http requests that are processed efficiently and without conflict. We achieved this by incorporating a thread lock mechanism. This locking system ensures that at any given moment, there isnt a call to the TraCI that would kill the simulation.

Moreover, the real-time aspect of our API allows for live requests. This means that during the simulation, various parameters can be adjusted, data can be retrieved, and new scenarios can be introduced on the fly without pausing or restarting the simulation.

In essence, our API isn't just a bridge between CARLA and SUMO, it's a dynamic tool that provides the flexibility and reliability needed for intricate and real-time co-simulations that serve as well for documented tests that we along the way performed to the numerous functionalities tested.

4.2.9.1 Python

Python was chosen as the primary programming language due to its readability and extensive library support, which are ideal for rapid development and prototyping. Furthermore, since both CARLA and SUMO offer Python-based API, using Python for our development was a logical choice.

4.2.9.2 Flask

Flask, a lightweight and extensible web framework, was chosen to run our web server, primarily due to its simplicity and robustness. The server functions as an intermediary, processing API requests and coordinating the actions between CARLA and SUMO. One of the major advantages of using Flask is its built-in support for RESTful APIs, which simplifies the task of sending and receiving data between the two simulation environments. Moreover, Flask's flexibility and modularity enable us to easily extend functionalities and implement custom logic as required. This is particularly useful for handling various types of requests, ranging from starting and stopping simulations to retrieving or updating simulation parameters in real time. By using Flask as our web server, we create a seamless integration between CARLA and SUMO, thereby enabling more dynamic and realistic traffic simulations.

TraCI is used to interact with SUMO for tasks like retrieving providential real-time information about vehicles, managing traffic lights, and vehicle spawning. By leveraging CARLA Python API, we have gained the ability to interact programmatically with the simulation, giving us a finer level of control over a multitude of variables. For example, we can specify sensor configurations, and even introduce obstacles or unexpected events, all of which are essential for creating a realistic and comprehensive simulation experience.

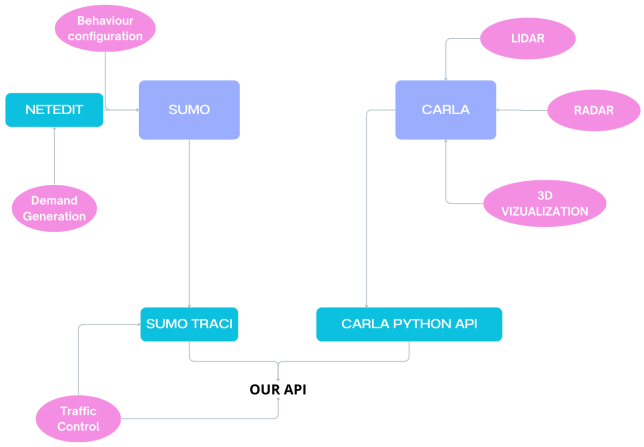


Figure 4.2: System architecture

The API's versatility also facilitates integration with other tools and frameworks, allowing us to create a rich, multi-faceted simulation setup. This is particularly beneficial when combined

with SUMO for traffic modeling, as the Python API ensures smooth interoperability between the two platforms. The sketch of the system is described in the picture below.

As we end this chapter and have defined the objectives and described the system architecture we now will start to introduce the development process and how we managed to get to the predefined objectives and what we used to achieve that.

Chapter 5

Development

This chapter outlines the methodology of integrating Car Learning to Act (CARLA), an open-source platform for autonomous driving research, with Simulation of Urban Mobility (SUMO), a widely-used traffic simulation software. The synergy of these two platforms creates a robust environment for testing and developing various aspects of smart city technology.

5.1 The map

The foundational step in creating a simulation environment for our transportation and traffic management study was the selection of an appropriate map. This choice significantly influences multiple facets of the research, from environmental realism to the complexity of road networks and the scope for including specific features of interest.

5.1.1 Need for Realistic Environments

The map, in many ways, is the canvas for our simulation masterpiece. Selecting an appropriate map is a balancing act as it must provide realistic challenges for vehicular simulations, be visually coherent, and crucially, be compatible with both CARLA and SUMO. In initial deliberations, a real map, such as the one representing the city of Porto, was considered due to its authentic portrayal of roadways, intersections, and landmarks. However, using such maps introduces challenges in obtaining accurate data, modeling intricate road geometries, and modeling the 3D environment in the most realistic way which was not possible without certain tools such as **roadrunner** as we observed in our state of art analysis.

5.1.2 Advantages of Predefined Environments

Given the complexities of using real-world maps, we turned our attention towards predefined environments that simulation platforms like CARLA offer. Among these, we chose the **Town 05**

map for its balanced attributes.



Figure 5.1: Map overview

5.1.2.1 Why Town 05?

The Town 05 map was selected based on several key benefits:

- **Rich 3D Models:** The map comes with pre-designed 3D models for buildings, roads, and other infrastructural elements, thereby heightening the environmental realism.
- **User-Friendliness:** Using a predefined map sidesteps the laborious process of manually modeling road geometries and landmarks, thus expediting the simulation setup.
- **Varied Scenarios:** With its grid layout, cross-junctions, and multiple lanes, Town 05 supports a range of traffic scenarios and interactions. This makes it versatile enough for exploring a variety of traffic management strategies.

According to CARLA official documentation, Town 05 is described as a "Squared-grid town with cross junctions and a bridge. It has multiple lanes per direction"[13], and that's exactly what we are looking for in this project. This characterization emphasizes the map's utility for investigating an array of traffic behaviors and interactions, aligning closely with the objectives of our simulation study.

5.1.3 Map Conversion to XML Format

Our development process commenced with the translation of the Town05 map, originally in CARLA's native format, an OpenDRIVE (.xodr), into a format compatible with both SUMO and Netedit. To achieve this, we employed the Python function **netconvert_carla** provided by CARLA. This function accepts an OpenDRIVE (.xodr) file and outputs a compliant XML file

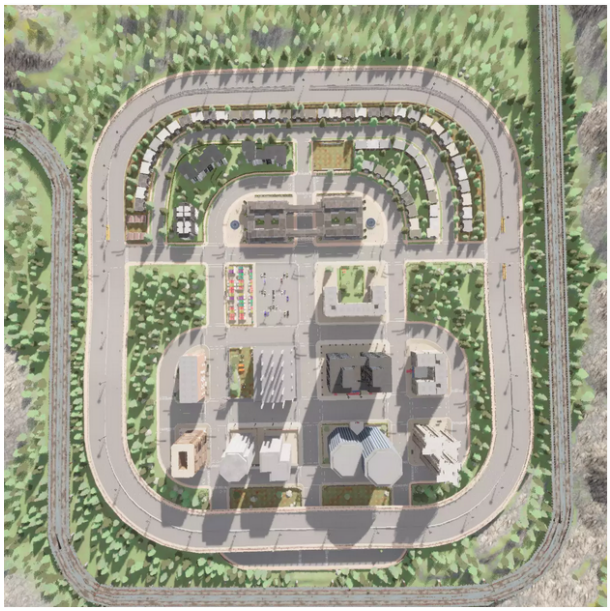


Figure 5.2: Road Network in Town 05

suitable for SUMO and Netedit. This stage was critical and necessitated meticulous attention, as it was imperative to ensure that the resulting network was accurate—complete with correctly connected edges and well-defined traffic signals.

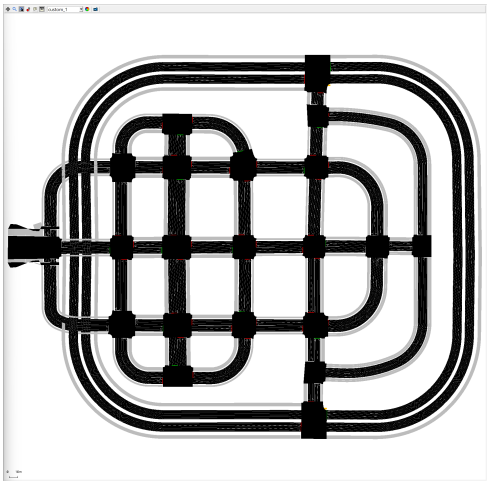


Figure 5.3: Translated Map on SUMO

5.2 Demand

5.2.1 Mapping CARLA's Vehicle Blueprints to SUMO's Vehicle Types

To seamlessly integrate vehicle types from CARLA into the SUMO simulation environment, it's vital to establish a clear correspondence between them. To facilitate this translation, we utilized the `create_sumo_vtypes.py` Python script, graciously provided by the CARLA development team. This script ingeniously maps every vehicle blueprint from CARLA to a tailor-made vehicle type (`vTypes`) in SUMO. This meticulous translation ensures that vehicle specifications, such as dimensions, defined in CARLA blueprints find their accurate representation in SUMO simulations. This precision is pivotal, especially when we delve into the intricacies of vehicle navigation in upcoming chapters.

As an illustration, here are a few examples from the output file:

Listing 5.1: SUMO vehicle type example

```
<vType id="vehicle.toyota.prius" length="4.54" vClass="evehicle"
       width="2.00" height="1.54" />
<vType id="vehicle.volkswagen.t2" length="4.47" vClass="passenger"
       guiShape="passenger/van" width="1.81" height="2.05" />
<vType id="vehicle.yamaha.yzf" length="2.19" vClass="motorcycle"
       width="0.77" height="1.63" />
```

With the CARLA vehicles now harmoniously mapped to their respective SUMO counterparts, we're poised to delve into the nuances of demand creation.

5.2.2 Creating Demand Elements in Netedit

Following these conversions of both map and vehicles types, we imported the map into Netedit using its native import feature and selecting the freshly converted XML file. SUMO provides an array of methods for traffic generation, one of the most prominent being Netedit. As a holistic platform, Netedit equips users with a comprehensive toolkit to sculpt various demand elements. This includes the likes of vehicles and pedestrians, enabling the creation of simulations that adeptly reflect real-world traffic scenarios.

5.2.2.1 Importing Custom Vehicles

A pivotal step in the process involves importing the vehicle types that we translated as detailed in Section 5.2.1. It's imperative for SUMO to be cognizant of these vehicle types as they determine the characteristics of vehicles when instantiated within the simulation.

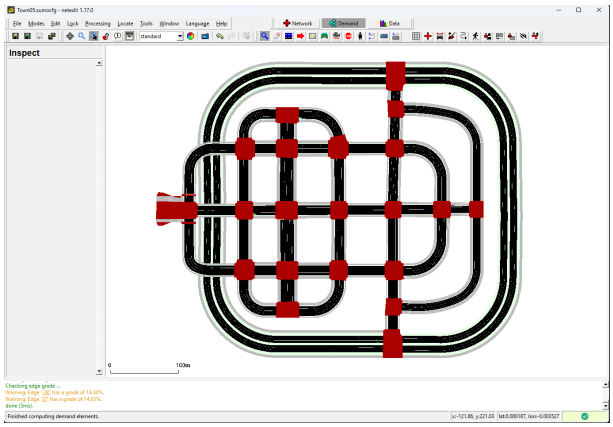


Figure 5.4: Map in NETEDIT

5.2.2.2 Creating Vehicle Routes

Netedit is characterized by its multifaceted `Demand Mode`. Within this mode, users can revel in a myriad of options tailor-made for defining intricate traffic demands. The user-friendly interface extends the liberty to either meticulously chart out individual vehicle routes or to designate a broader set of edges, leaving the route specifics to be determined dynamically by the simulation. This granularity ensures that unique vehicular behaviors and interactions can be captured with precision.

Furthermore, a standout feature of Netedit is its capability to orchestrate a sustained flow of vehicles over a user-defined duration. This dynamism is invaluable, enabling researchers and developers to emulate diverse traffic patterns, ranging from peak rush hour congestion to late-night sparse traffic.

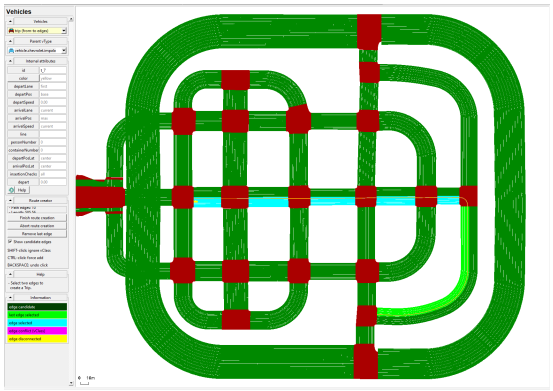


Figure 5.5: The process of defining routes using the available edges, denoted in green.

The way we defined the different routes depended significantly on the specific objectives of each test case we were going to explore. Factors such as desired traffic density, points of interest, interaction between different types of vehicles, and existing road conditions influenced our route selection. This flexibility allowed us to emulate real-world conditions as closely as possible, thereby enhancing the validity and relevance of our simulation outcomes.

5.3 Detailed Traffic Scenario

For our testing purposes we were primarily focusing on a specific traffic light junction with four intersections. The motivation behind this was to understand and analyze the vehicle behaviors, flow patterns, and potential congestion points at a multi-directional junction.

5.3.1 Traffic Light Centric Testing

The chosen junction that we observe in figure 5.6 is emblematic of a standard four-way intersection, regulated by a set of traffic lights. It plays a critical role in our simulations due to its complexity and the variety of potential traffic scenarios it can present.



Figure 5.6: Intersection overview

5.3.1.1 Flow Creation

To faithfully replicate the challenges and dynamics of genuine traffic conditions, we tailored our simulation around a flow that directly navigates through the highlighted intersection. The method employed to generate traffic at this junction was as follows:

- **Peak Hour Flow:** Our primary focus was on simulating the vehicular volume typical of peak morning hours. This entailed a dominant influx of vehicles gravitating towards commercial zones. To heighten the realism of our simulation, it was paramount for vehicles to approach the intersection at typical cruising speeds. As such, vehicles were spawned not directly at the intersection, but several intersections prior, ensuring they entered the main junction with momentum akin to genuine traffic patterns.

The scenario provided insights into various challenges faced by vehicles at multi-directional junctions, such as:

- Waiting times at the traffic light during traffic jams.
- Interaction between vehicles.

- The efficiency of traffic light timings in handling different traffic volumes as the volume increases over time.

5.4 API

The Application programming interface (API) main methods start by setting up environment variables for SUMO and initializing the CARLA client, connected to a specific port and Internet Protocol (IP). It also sets up a timeout that sets the maximum time a network call is allowed before blocking it and raising a timeout exceeded error, world settings, and simulation parameters for both SUMO and CARLA.

Listing 5.2: Main API initialization method

```
if __name__ == "__main__":
    #check environmental variables
    #initialize connection
    client = carla.Client('127.0.0.1', port)
    client.set_timeout(60.0)
    ...
```

CARLA provides a SUMO integration folder that contains a carla and sumo modules as well as a synchronization module that allows a providential role in our API as it ticks both simulations and takes care of the vehicle and other simulation synchronization aspects.

5.4.0.1 CARLA Simulation

The CARLA simulation is initialized using the module we just introduced, and settings like map, world, etc., are set up. Also sets fixed delta seconds for the simulation time-step, we defined 0.1 seconds to be ideal since after some experiments it allowed a seamless synchrony between both simulations running.

Listing 5.3: Initializing CARLA module

```
carla_simulation = CarlaSimulation('127.0.0.1', port, step_length)
```

5.4.0.2 SUMO Simulation

The SUMO simulation starts by loading the previously generated XML file, which we discussed in Section 5.2. Alongside this, the routes file and other relevant simulation parameters are configured. Notably, the **step length** used in the SUMO simulation is synchronized to match the value we set in the **Carla_simulation**. The **host** and **port** are initially set to `None`, allowing for the creation of a new SUMO server rather than connecting to an existing one. While our

preliminary experiments involved running a SUMO simulation first and then connecting our API, we ultimately chose to initialize everything through the API in the final version. This approach eliminates the need for the SUMO graphical interface and sidesteps compatibility issues, making the system more robust.Of course during the testing phase we did test with the SUMO Graphical User Interface (GUI) set to **True**, but then as SUMO would only serve as a backend to generate the vehicles and traffic light management we ultimately set it to **False**

Listing 5.4: Initializing SUMO module

```
sumo_simulation = SumoSimulation(configFile, step_length, host=None,
port=None, sumo_gui=False, client_order=1)
```

5.4.0.3 Synchronization

A synchronization module is put in place to coordinate both the state and time-step updates between the CARLA and SUMO simulations. To set this up, the previously initialized `sumo_simulation` and `carla_simulation` instances are inputted into the `SimulationSynchronization` method provided by CARLA and we named it the synchronization module. Additionally, the traffic light manager must be specified, which can either be controlled by SUMO or CARLA. In our case, we opted for SUMO to manage the traffic lights because it offers more precise and customizable control over traffic signal timing and phases, thereby enhancing the realism and validity of our simulation scenarios.

Listing 5.5: Connecting both Simulations into the synchronization module

```
synchronization = SimulationSynchronization(sumo_simulation,
carla_simulation, 'sumo', False, True)
```

5.4.0.4 Start Application server

Finally we just run our local application development server and now our main instances are now initialized and we can start defining several endpoints that will continue with the enrichment of our API.

5.4.0.5 Main simulation endpoint

Having the map loaded to the simulation and initialized all instances now we just need to create a cycle that will take care of each simulation step. The simulation loop is initiated by a Flask API endpoint called `/startSimulation` in which we upfront tick the synchronization module, which by itself tells CARLA and SUMO that a new simulation step is ready to be processed.

Listing 5.6: Method to start the simulation

```
@app.route('/startSimulation', methods=['POST'])
```

```
def start():
    while simulation_running and step_counter < max_steps:
    # Main simulation loop
    ...
```

This method takes care of the synchronization, handles the simulation tick, and manages traffic lights, and sensors for each vehicle in the simulation until the predefined maximum number of steps.

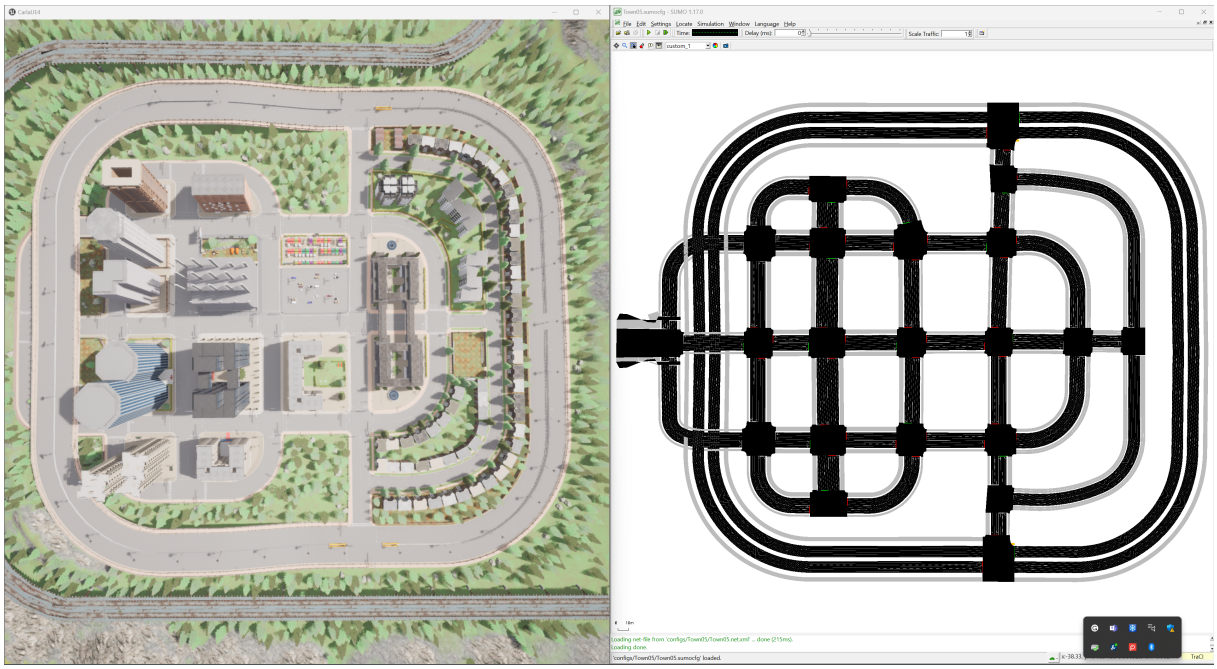


Figure 5.7: CARLA and SUMO running side by side

5.4.1 Weather Control

As our co-simulation environment matured, we began to push the boundaries of what could be achieved, diving deeper into the features that both CARLA and SUMO provide. A primary objective was to explore real-world scenarios, and the control of environmental factors, particularly the weather, was a significant area of interest.

Weather conditions can significantly impact autonomous driving systems, affecting sensor readings and vehicle behavior. By integrating advanced weather control into our API, we provided a mechanism for simulating various weather conditions, allowing for rigorous testing under diverse scenarios.

The capability of setting weather attributes dynamically was implemented in our API. The route `/set_weather` was designed to accept a POST request, where the body of the request would specify the desired weather parameters, such as cloud cover, humidity, precipitation levels, and wind intensity, among other important attributes.

A snippet of the API endpoint:

Listing 5.7: Method to set weather

```
@app.route('/set_weather', methods=["POST"])
def setWeather():
    data = request.get_json()
    try:
        world = client.get_world()
        weather = world.get_weather()
        weather.cloudiness = data['cloudcover']
        weather.wetness = data['humidity']
        weather.precipitation = data['precipitation']
        weather.precipitation_deposits = data['precipitation_deposits']
        weather.wind_intensity = data['wind_intensity']
        weather.fog_density = data['fog_density']
        weather.sun_azimuth_angle = data['sun_azimuth_angle']
        weather.sun_altitude_angle = data['sun_altitude_angle']
        weather.dust_storm = data['dust_storm']
        world.set_weather(weather)
```

With this, we could instantaneously change the environmental conditions within the CARLA simulation, making it an invaluable tool for assessing how different weather conditions influence autonomous driving algorithms and decision-making processes.

5.4.2 Return live information

We also integrated some endpoints that would return live information about the state of the simulation, such as the available blueprints from CARLA, the state of each vehicle in the simulation, such as **lane** ,**type** ,**postion** ,**speed**.

5.4.3 Behavior Modifiers and Their Significance

We identified critical behavioral parameters—namely ‘tau‘ (reaction time), ‘sigma‘ (driver imperfection), and ‘impatience‘—that can significantly influence traffic patterns[2].

Tau (Reaction time): Represents the time it takes for a driver to react to an event. A higher value indicates a slower reaction, while a lower value denotes a quick reaction. **Default value: 1.0**

Sigma (Driver imperfection): Captures the degree to which a driver deviates from their intended speed. A higher value suggests frequent speed changes, while a lower value implies consistent speed maintenance. **Default value: 0.5**

Impatience: Measures the likelihood of a driver becoming impatient, especially when stuck behind slower vehicles. A higher value indicates more aggressive drivers, whereas a lower value points to patient driving behavior. This value increases whenever the driver has to stop unintentionally. **Default value: 0.5**

5.4.3.1 Simulated Traffic Scenarios

To gain a deeper understanding of traffic dynamics under different behavioral conditions, we defined and simulated the following scenarios:

1. **Conservative Driving:** Characterized by slow reaction times (Tau: 2.0), consistent speed maintenance (Sigma: 0.2), and high patience (Impatience: 0.2).
2. **Aggressive Driving:** Features quick reactions (Tau: 0.5), frequent speed deviations (Sigma: 0.8), and low patience (Impatience: 0.8).
3. **Mixed Driving Behavior:** A realistic blend with medium reaction times (Tau: 1.0), moderate speed deviations (Sigma: 0.5), and neutral patience (Impatience: 0.5).

To dynamically adjust these parameters, we introduced another endpoint in our API, that receives in the body the parameters we want to set:

Listing 5.8: Method to modify behaviours conditions

```
@app.route('/modify_parameters', methods=['POST'])
data = request.json
tau = data.get('tau', None)
sigma = data.get('sigma', None)
impatience = data.get('impatience', None)
modify_rou_file(tau, sigma, impatience)
reload_simulation()
...
```

This allows users to set the **tau**, **sigma**, and **impatience** values on the fly, altering the behavior of the simulated vehicles in real time. On the backend, this endpoint modifies the ‘.rou.xml‘ file which SUMO utilizes to introduce vehicles into the simulation.

On receiving a request, the API modifies the ‘.rou.xml‘ file to adjust the behavior attributes of the ‘vType‘ elements. For instance:

Listing 5.9: Example of SUMO vehicle type with the behavioral changes

```
<vType height="1.48" id="vehicle.ford.mustang" impatience="0.0"
length="4.90" sigma="0.5" tau="1.0" vClass="passenger" width="2.06" />
```

In the example above, the default values for **tau**, **sigma**, and **impatience** are 1.0, 0.5, and 0.0, respectively. When the API receives new values, it updates this element accordingly before restarting the SUMO simulation.

5.4.4 Traffic Light Management

In urban environments, traffic congestion poses a critical challenge. It not only affects economic productivity but also significantly deteriorates the quality of life for residents. A pivotal tool in controlling and modulating this traffic flow is the traffic light system. Traditional traffic lights operate on fixed cycles, often leading to inefficient control in dynamic traffic conditions.

Our simulation environment now harnesses an adaptive algorithm, adapting traffic light phases in response to real-time traffic conditions. Such a dynamic adjustment assures that lanes with higher congestion are prioritized, potentially enhancing overall traffic throughput.

5.4.4.1 Fundamental Goals

- **Real-time Responsiveness**: Actively perceive and promptly adapt to fluctuating traffic conditions, especially during peak hours or unexpected congestion scenarios.
- **Reduced Wait Times**: Aiming to cut down cumulative wait times at intersections by dynamically adjusting to traffic conditions.
- **Boosted Traffic Throughput**: Prioritize congested lanes to augment the total number of vehicles that traverse an intersection within a specific time frame.

Transitioning from a static approach to this adaptive traffic light system addresses the shortcomings of conventional systems, potentially ensuring smoother traffic flow and less congestion.

5.4.4.2 Utility-based Adaptive Traffic Light Algorithm

Our traffic management approach leverages a utility-based algorithm. The core idea is to evaluate the congestion level of each lane and assign it a corresponding utility or reward score. The traffic light phase to be activated is chosen based on maximizing the overall utility. The main goal is to ensure smooth traffic flow and minimize vehicle waiting times.

5.4.4.3 Key Components of the Approach

1. Assess lane congestion using a utility function.
2. Give priority to lanes with vehicles that have been waiting for an extended period.

3. Ensure a minimum time gap between consecutive traffic light phase switches to avoid frequent oscillations.

5.4.4.4 Detailed Algorithmic Steps

1. **For each traffic light**:
   - Fetch lanes governed by the current traffic light.
   - **For each of these lanes**:
     - Compute the lane's utility based on the ratio of occupied lane length to its total length.
   - Compute the combined utility for all possible traffic light phases.
   - Prioritize lanes where vehicles have been waiting longer than a set threshold, e.g., 45 seconds.
   - If the time since the last phase switch exceeds a set interval (e.g., 15 seconds), switch to the phase with the highest combined utility.

Listing 5.10: Utility Function

```
def utility_max_min(lane):
    vehicles_on_lane = traci.lane.getLastStepVehicleIDs(lane)
    total_occupied_length = sum([traci.vehicle.getLength(vehicle)
    for vehicle in vehicles_on_lane])
    lane_capacity = traci.lane.getLength(lane)
    lane_occupancy_percentage = total_occupied_length / lane_capacity
    return lane_occupancy_percentage
```

Listing 5.11: Traffic light algorithm

```
def manage_traffic_lights():
    # Calculate utility for each lane.
    # Sum up utilities for possible traffic light phases.
    # Determine if lanes have had waiting vehicles beyond
    #the set threshold.
    # Switch to the phase with the highest utility,
    #considering minimum time constraints.
```

The utility-based system allows for more dynamic and responsive decision-making in traffic management. The challenge, however, lies in designing an optimal utility function that balances the system's responsiveness against minimizing vehicle wait times.

5.4.5 Sensors and Data Collection

One of the striking features of CARLA is its robust support for a wide array of sensors. In the realm of autonomous driving, these sensors are critical, serving as the eyes and ears of autonomous vehicles. Harnessing these sensors' capabilities, we integrated data collection mechanisms within our API.

In the co-simulation environment, the real power of CARLA capabilities becomes evident when vehicles are dynamically spawned. Upon the departure of new vehicles in the SUMO simulation, they are correspondingly represented within CARLA. To enrich the data collection and analysis capabilities, we ensure that every newly spawned vehicle in CARLA is equipped with an array of sensors.

The following procedure is followed:

1. Monitor for newly departed vehicles in SUMO using Traffic Control Interface (TraCI) to obtain this vehicles.
2. For each new SUMO vehicle, ascertain its corresponding CARLA actor.
3. Attach the desired sensors to this CARLA actor. In our implementation, we've attached:
   - **Radar Sensor:** A radar sensor to capture velocity data of objects in the vehicle's vicinity. It can be crucial for understanding traffic flow and potential hazards.
   - **Obstacle Detector:** This sensor detects any immediate obstacles in the vehicle's path.
4. All these sensors are added to a `sensor_list` as they are required to be saved, or their state is lost through the execution, facilitating management and data retrieval throughout the simulation run.

Each sensor has its dedicated callback function to handle the data it produces. For instance, the radar sensor uses the `radar_callback` and the obstacle detector uses the `Obstacle_callback`. These callbacks process the sensor data, storing in our sensor_network global variable in real time the most updated information about each car, identified by its SUMO id, ensuring real-time analysis and response, as well as storing the data for post-simulation studies.

5.4.6 Vehicle Detection using Light Detection And Ranging (LIDAR)

We developed a system to detect vehicles using a LIDAR sensor. The primary components of this system include the positioning of the LIDAR sensor, data acquisition, data preprocessing, and vehicle detection algorithm. The goal of creating this system was to first, explore the sensors that CARLA provide, and second to verify other than visually that SUMO and CARLA are

interacting in the same way and that what one generates is observable in the other system by creating a mechanism that would count the number of vehicles that would go through our detector.

5.4.6.1 LIDAR Sensor Positioning

We choose to place the sensors on all the traffic lights of our intersection and to obtain the best results, the LIDAR sensor was positioned at an optimal height and orientation.

5.4.6.2 Sensor Configuration and Attributes

The LIDAR sensor is configured with the following attributes:

- Channels: 32
- Points per second: 50,000
- Sensor tick: 1.0s
- Rotation frequency: 100 Hz
- Range: 100 units
- Upper Field of View: -70º



Figure 5.8: Sensors placement

5.4.6.3 Dynamic Sensor Placement

To ensure optimal data acquisition, the sensor is dynamically positioned based on the yaw of the traffic light. An offset of 4 units forward in the direction of the traffic light is computed. This results in the sensor being placed at:

$$\text{lidar location} = (x_{\text{coord}} - \delta x, y_{\text{coord}} - \delta y, z = 4)$$

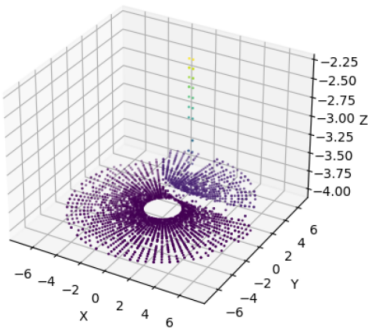


Figure 5.9: LIDAR Sensor output

5.4.6.4 Data Acquisition and Processing

The LIDAR sensor provides raw point cloud data, which undergoes preprocessing for vehicle detection. For visual purposes we created a plot that would take into input the x, y and z values of the raw data. This was mainly because we wanted to optimize the placement of the sensor so it would not detect other lanes vehicles or even static data like lamps. The raw LIDAR data is converted into a numpy array using `np.frombuffer`, simplifying subsequent processing steps. Vehicle points are isolated using a height-based filtering method. The system focuses on points within the height range of 2 to 3.5 units. This filtering step aids in the removal of outlier points that are outside of the typical vehicle height range.

5.4.6.5 Clustering using DBSCAN

The DBSCAN clustering algorithm segregates points into distinct clusters, with each cluster potentially representing a vehicle. The parameters for DBSCAN include an epsilon value of 0.5 and a minimum sample size of 10.

5.4.6.6 Vehicle Position Detection

Beyond mere vehicle counting, the system calculates the centroid of the largest cluster, considering it as the representative position of the detected vehicle. The position is determined using:

Listing 5.12: Vehicle detection algorithm

```
clustering = DBSCAN(eps=0.5, min_samples=10).fit(vehicle_points)
labels = clustering.labels_
unique_labels = set(labels)

largest_cluster = max(set(labels), key=list(labels).count)
cluster_points = np.array(vehicle_points)[labels == largest_cluster]
centroid = np.mean(cluster_points, axis=0)
return centroid
```

5.4.6.7 Handling Multiple Detections of Same Vehicle

To ensure that the same vehicle isn't counted multiple times, the system tracks the last detected vehicle's position for each traffic light. It computes the Euclidean distance between the previously detected vehicle position and the current one. If this distance is below a threshold (e.g., 2 units), the detection is disregarded.

5.5 Resume

In the comprehensive development of our API, we navigated a multifaceted landscape of challenges, innovations, and triumphs. Starting with a foundational understanding of our objectives, we delved deep into creating an immersive and dynamic environment, leveraging the capabilities of the CARLA simulator. The integration of the LIDAR sensor, pivotal for accurate vehicle detection at junctions, presented its own set of hurdles. However, with relentless iteration and refinement, we successfully calibrated and positioned it to meet our precision standards.

Parallel to our sensor work, we crafted a nuanced algorithm for traffic light management, ensuring seamless vehicular flow and minimizing congestion. The environment was designed to be as close to real-world scenarios as possible, accounting for varying traffic conditions, light transitions, and vehicular interactions. The holistic nature of our approach, from sensor accuracy to traffic light algorithms, culminated in an API that is not just robust in its current state, but also primed for future advancements and scalability.

As we conclude this development narrative, we acknowledge the depth and breadth of work that has shaped our API. The subsequent chapter will pivot to a critical analysis, where we will thoroughly verify and validate the data and insights generated, ensuring that our system's output stands up to rigorous scrutiny and validates the objectives defined previously.

Chapter 6

Evaluation

In this chapter, we delve into the evaluation process, focusing on key objectives, methodologies, and the obtained results.

6.1 Objectives

Our evaluation centered around understanding the robustness and performance of the system. Key objectives included:

Simulation Test Procedure: Initializing a structured simulation test procedure to maintain consistency and reliability.

System Correctness: Ensuring the accuracy and consistency of simulation outputs through reproducibility and consistency tests.

Weather Dynamics: Assessing the impact of various weather conditions on Simulation of Urban Mobility (SUMO) vehicles and evaluating the need for manual adjustments to driving dynamics.

Traffic Flow: Analyzing traffic flow patterns, identifying inefficiencies, and evaluating the impact of dynamic traffic management algorithms.

Driving Behaviors: Understanding the relationship between different driving behaviors and their impact on emissions and trip duration.

Sensor Testing: Evaluating the performance of radar and LiDAR sensors, including visualization, calibration, and comparison with benchmark data.

6.2 Methodology

6.2.1 Simulation Test Procedure:

The procedure for conducting the simulation tests consists of a series of steps that ensure seamless integration between the Car Learning to Act (CARLA) simulation, SUMO, and our API, and the subsequent triggering of the simulation.

1. **Initializing the CARLA Simulation:**
   Launch the CARLA simulation environment by executing the following command in the terminal:

   ```
   ./CarlaUE4.exe -carla-rpc-port=4000
   ```

   This command starts the CARLA simulator and sets the CARLA server to listen on port 4000.

2. **Connecting the API:**
   Once the CARLA simulator is running, connect the API by executing:

   ```
   py -3.7 api.py
   ```

   This command runs the `api.py` script using Python 3.7, establishing a connection between the CARLA simulator and our API.

3. **Triggering the Simulation via Postman:**
   With the CARLA simulator and the API running, we can trigger the simulation test. For this, we use Postman, a popular API testing tool. Send a request via Postman to the appropriate endpoint of the API to initiate the simulation.

By following this procedure, we ensure a structured and consistent approach to our simulation tests, minimizing potential errors or inconsistencies.

6.2.2 Consistency and Reproducibility:

When conducting simulations, especially for systems that can have profound implications on real-world operations, ensuring the correctness and consistency of the outputs is of paramount importance. To validate the correctness of our system, we performed multiple instances of the same simulation and analyzed the resulting outputs. To assess the reliability of our simulation, it was executed eight times, each under identical conditions. All instances shared the same parameters, environment settings, and agents, ensuring consistent initialization. Furthermore, each simulation ran independently, safeguarding against any potential data or parameter overlap. Importantly, these tests were conducted without any traffic light algorithms, providing foundational data for subsequent analyses and comparisons.

6.2.2.1 Simulation Testing Environment

For our testing environment, we defined a simulation duration of 260 seconds. This timeframe was chosen based on preliminary evaluations that determined it as an adequate duration to observe significant traffic patterns and get a relatively good finalized amount of trips without overburdening computational resources.

6.2.2.2 Data Sources

Our primary data sources for the analysis came from the `tripinfo.xml` and `queue.xml` files, which are generated by SUMO. These files are of particular significance due to the breadth of information they capture. Not only do they record key variables at the time of departure and arrival, such as speed and delays, but they also include metrics like the total duration of the route and time loss incurred by each vehicle and queue times for each available lane in the simulation.

An example of what the `tripinfo.xml` file exports[2]:

```
<tripinfos>
    <tripinfo id="<VEHICLE_ID>" \
        depart="<DEPARTURE_TIME>" departLane="<DEPARTURE_LANE_ID>" \
        departPos="<DEPARTURE_POSITION>" departSpeed="<DEPARTURE_SPEED>" \
        departDelay="<DEPARTURE_DELAY>" \
        arrival="<ARRIVAL_TIME>" arrivalLane="<DEPARTURE_LANE_ID>" \
        arrivalPos="<ARRIVAL_POSITION>" arrivalSpeed="<ARRIVAL_SPEED>" \
        duration="<TRAVEL_TIME>" routeLength="<ROUTE_LENGTH>" \
        waitingTime="<SECONDS_WAITING_FOR_TRAFFIC>"
    ... information about further vehicles ...
</tripinfos>
```

Moreover, the **tripinfo.xml** includes an **emissions** tag for each vehicle. This is instrumental for our study as it offers a variety of environmental metrics, such as CO, CO2, HC, PMx, NOx, and fuel consumption. To visualize and analyze these diverse datasets, we utilized the `plotXMLAttributes.py` script, given by SUMO, which allowed us to produce graphical visualizations of the attributes. We also created a 3D animation of the simulation, which we analyzed to better comprehend the interaction of vehicles in the simulated environment.

6.2.3 Weather Dynamics

Weather conditions play a pivotal role in real-world driving scenarios.The CARLA simulator provides a range of weather presets and parameters that can be adjusted to simulate various weather conditions. These conditions include rain, fog, and different times of day, each affecting visibility, friction, and overall driving conditions. To assess the impact of weather on our system,

we conducted simulations under three distinct weather conditions: clear sky, rain, and fog. Each simulation was run for again a duration of 260 seconds, allowing us to observe the system's behavior over an extended period under different weather scenarios.

6.2.4 Traffic Flow

To initiate our traffic flow analysis, it's vital to comprehend the inherent patterns and behaviors exhibited within traffic systems. Much like blood flowing through veins, traffic represents the lifeblood of urban infrastructure. Its rhythms, surges, and lulls all speak to a city's functionality, efficiency, and even its culture. Analyzing these flows gives insight not just into logistical challenges, but also into the very nature of human movement within structured confines. By focusing on the peak hour flow, we aimed to scrutinize one of the most taxing and common scenarios faced in urban traffic management. The lessons and observations drawn from this study provide a foundational understanding upon which adaptive solutions, like dynamic traffic management algorithms, can be designed and implemented.

6.2.4.1 Baseline Traffic Flow

To establish a baseline, we conducted simulations without any traffic management algorithms. This allowed us to observe the natural flow of vehicles in the simulated environment and identify any inherent inefficiencies or congestion points.

6.2.4.2 Dynamic Traffic Management

Introducing dynamic traffic management algorithms, such as traffic lights and lane management, aimed to optimize traffic flow. We implemented traffic light control algorithms that dynamically adjusted signal timings based on real-time traffic conditions. Additionally, lane management strategies were employed to balance the distribution of vehicles across multiple lanes.

6.2.4.3 Performance Metrics

Key performance metrics for traffic flow analysis included:

Average Speed: Calculated as the mean speed of all vehicles in the simulation.

Throughput: Measured as the number of vehicles passing through a given point within a specified time frame.

Queue Length: Assessed by analyzing the length of vehicle queues at various points in the simulation.

Travel Time: The total time taken by each vehicle to complete its route.

These metrics provided a comprehensive overview of the system's efficiency in managing traffic flow under different conditions.

6.2.5 Driving Behaviors

Understanding the impact of driving behaviors on overall system performance is crucial for developing a comprehensive autonomous driving system. Our evaluation focused on analyzing the effects of different driving behaviors on emissions, trip duration, and overall driving experience.

6.2.5.1 Driving Behavior Scenarios

We defined three driving behavior scenarios:

Aggressive Driving: Simulated by setting higher acceleration and speed parameters for a subset of vehicles.

Defensive Driving: Simulated by setting lower acceleration and speed parameters for another subset of vehicles.

Mixed Driving: A combination of aggressive and defensive driving behaviors to mimic real-world scenarios.

6.2.5.2 Metrics and Analysis

Key metrics for driving behavior analysis included:

Emissions: Assessing the environmental impact of different driving behaviors through the emissions data recorded in the `tripinfo.xml` file.

Trip Duration: Measuring the time taken by each vehicle to complete its route under different driving scenarios.

Traffic Flow Disruptions: Identifying any disruptions or congestion caused by aggressive driving behaviors.

Analyzing these metrics provided insights into the trade-offs between driving aggressiveness, environmental impact, and overall system efficiency.

6.2.6 Sensor Testing

The accuracy and reliability of sensors are critical for the success of autonomous driving systems. Our evaluation focused on testing radar and LiDAR sensors, including calibration, visualization, and comparison with benchmark data.

6.2.6.1 LiDAR Sensor Testing

We assessed the performance of LiDAR sensors through the following steps:

Calibration: Ensuring accurate alignment and calibration of LiDAR sensors to detect and measure objects correctly.

Visualization: Analyzing the visualization output of LiDAR sensors to identify potential blind spots or inaccuracies using the matplotlib library.

Benchmark Comparison: Comparing the LiDAR sensor data with benchmark data files to validate the reliability of environmental mapping.

6.3 Results and Discussion

The results of our evaluation provide valuable insights into the robustness and performance of the autonomous driving system. In this section, we present and discuss the key findings from each aspect of the evaluation.

6.3.1 Consistency and Reproducibility

The reproducibility tests demonstrated a high level of consistency in simulation outputs. All eight instances of the simulation under identical conditions produced consistent results, validating the correctness of our system.

From each run, three main metrics were extracted:

1. **Number of Cars**: A simple count of how many vehicles actually completed the route.
2. **Mean Duration**: The average duration taken by all cars during the simulation. This metric provides insights into the overall efficiency and flow of the system.
3. **Total CO2 Emissions**: An aggregate of all CO2 emissions produced by the cars. This is a crucial metric for understanding the environmental impact of the system.

The extracted data was then analyzed for consistency across the eight runs. We were particularly interested in determining:

- Any deviation in the metrics across the runs.
- The percentile error in the extracted metrics to understand the range and likelihood of any discrepancies.

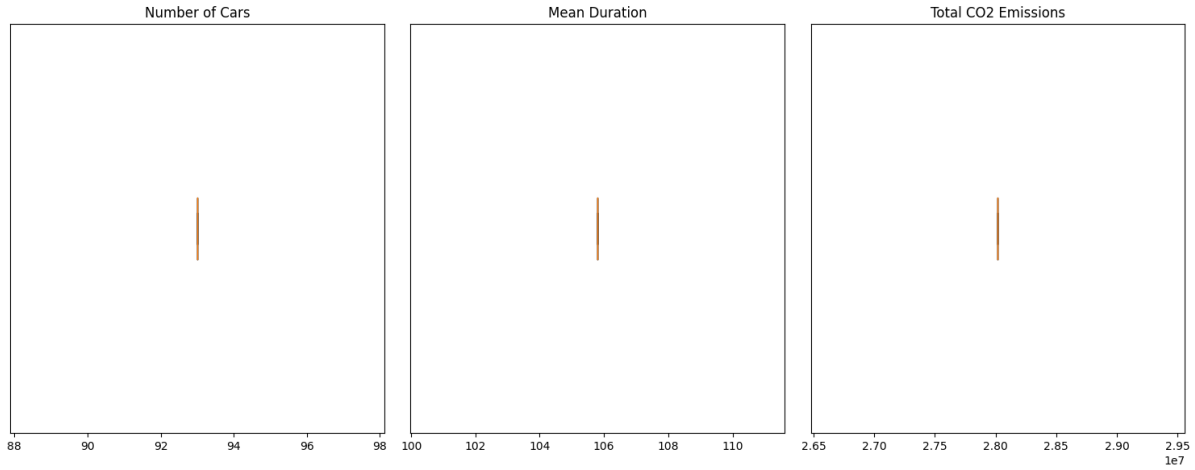


Figure 6.1: Deviation throughout all simulations

Upon analyzing the results of the six simulations, it was observed that the values across the three metrics were consistent. Specifically:

The percentile error analysis further corroborated this observation. All percentile errors were found to be negligible, confirming the robustness and consistency of our simulation outputs.

Metric	**Deviation**
Number of Cars	0%
Mean Duration	0%
Total CO2 Emissions	0%

The results from the repeated runs and the subsequent analysis provide strong evidence of the system's correctness. The simulation outputs the same results every time, suggesting that, under identical conditions, our system behaves predictably and reliably. This consistency is crucial, especially when using the simulation to inform real-world decisions or to derive insights about a system's operation.

While the consistency is reassuring, it's also essential to continually validate the system against real-world data and scenarios to ensure its broader applicability and accuracy. For now, we can be reasonably confident in the fidelity and reliability of the simulation outputs.

6.3.2 Weather Dynamics

Upon introducing diverse weather conditions such as rain, snow, and fog in CARLA and testing again for six times, we observed that the dynamics of SUMO vehicles remained largely unchanged. This observation was consistent across varying levels of weather severity, observing output files that SUMO provided.

- **Rain:** Despite the slippery conditions that come with rain, SUMO vehicles maintained their predefined driving dynamics.
- **Fog:** Reduced visibility due to fog also did not influence the driving patterns of SUMO vehicles.
- **Deposits:** Deposits of rain did not affect the braking time that the cars had through the simulation.
- **Wind:** Powerful wind conditions did not affect the velocity nor the mean time to finish the predefined routes.

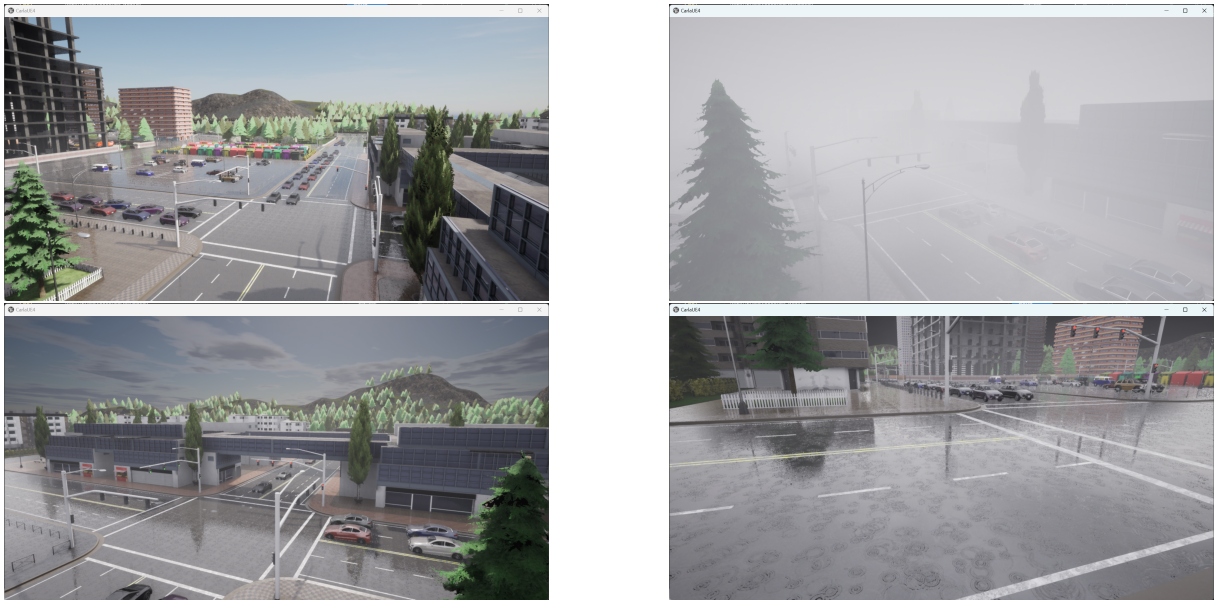


Figure 6.2: Different type of environmental conditions

6.3.2.1 Manual Adjustments to Vehicle Dynamics

Given the absence of inherent dynamic changes with weather variations, we explore into the initiative to manually modify the driving dynamics of the SUMO vehicles. Our aim was to emulate realistic driving behaviors under specific weather conditions. For instance:

- **Reduced Speed:** In response to rain or snow, we programmed a reduction in the speed of SUMO vehicles to mimic cautious driving behavior in real-life scenarios.
- **Increased Stopping Distance:** Slippery conditions due to wet or icy roads can increase the braking distance. We modified SUMO vehicles to take longer to come to a stop under such conditions.

This was done by increasing or decreasing the accelerator factor and the deceleration factor that each vehicle had.

6.3.2.2 Implications and Future Work

While the manual adjustments introduced more realism to our simulations, it highlighted the need for an intrinsic link between weather dynamics in CARLA and vehicle behaviors in SUMO. Future iterations of our integration could benefit from a deeper connection between these two aspects, offering an even more accurate and comprehensive co-simulation environment.

6.3.3 Traffic Flow

Initially, we were keen to observe the network's response to a specific traffic pattern: peak hour flow. The results of these observations were primarily gleaned from the outputs of the `tripInfo.xml` and `queue.xml` files as we previously stated.

In the preliminary stages of our simulation, before the introduction of any dynamic traffic management algorithms, we conducted a baseline observation of the traffic light behavior across the network as we stated in the previous section. One of the glaring inefficiencies that came to the fore was the suboptimal timing of the traffic signals.



Figure 6.3: Traffic intersection overview

There were evident instances where traffic lights remained green for extended duration even when there were no vehicles present in the lane to pass through the intersection. Such behavior not only reduced the overall efficiency of the traffic system but also contributed to unnecessary waiting times for vehicles in other lanes, exacerbating congestion and potentially leading to increased emissions. This observation further highlighted the pressing need for a dynamic and

adaptive traffic light management system, tailored to respond in real-time to the traffic conditions on the ground.

Upon reviewing our traffic data, we found notable fluctuations in trip duration, even for vehicles traversing the same route. This inconsistency can be attributed to the static traffic light system. In an urban environment, it's unrealistic to expect identical trip duration for every vehicle on a particular route, even under ideal conditions. One striking observation from the queue.xml file was the extensive queue times for certain lanes, reaching up to an alarming 140 seconds.

A closer look at the **tripinfo.xml** data provided insights into the vehicles' movement. Specifically, we recorded 93 vehicles successfully completing their routes within the predetermined timeframe. This count signifies that these vehicles adeptly navigated the intersection and concluded their predefined routes as intended. An essential gauge of efficiency in any traffic system is the volume of vehicles that manage to accomplish their intended paths within a set period, and this count of 93 vehicles served as a pivotal benchmark in our analysis.

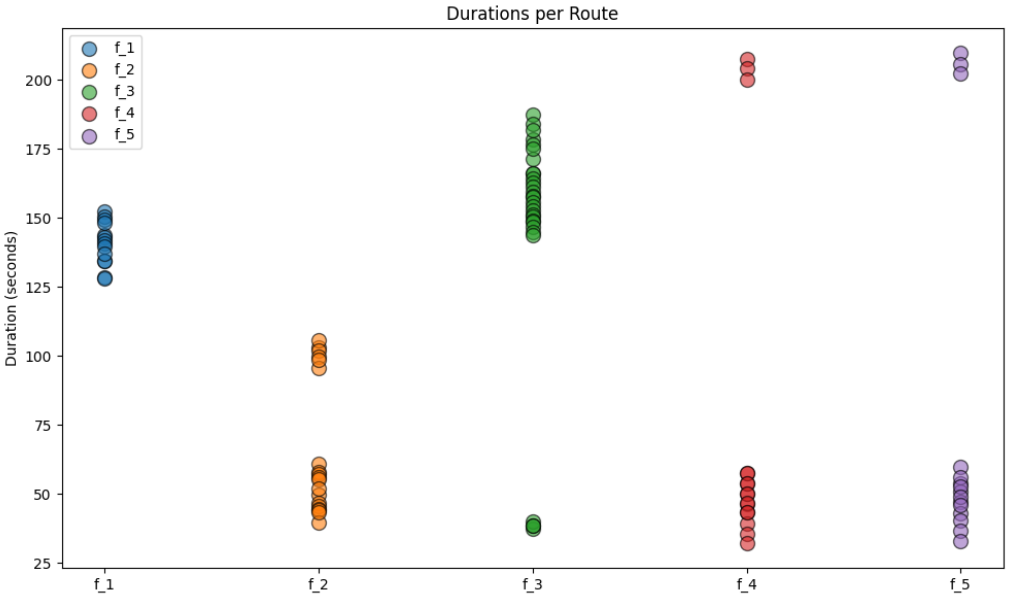


Figure 6.4: Duration per route per lane

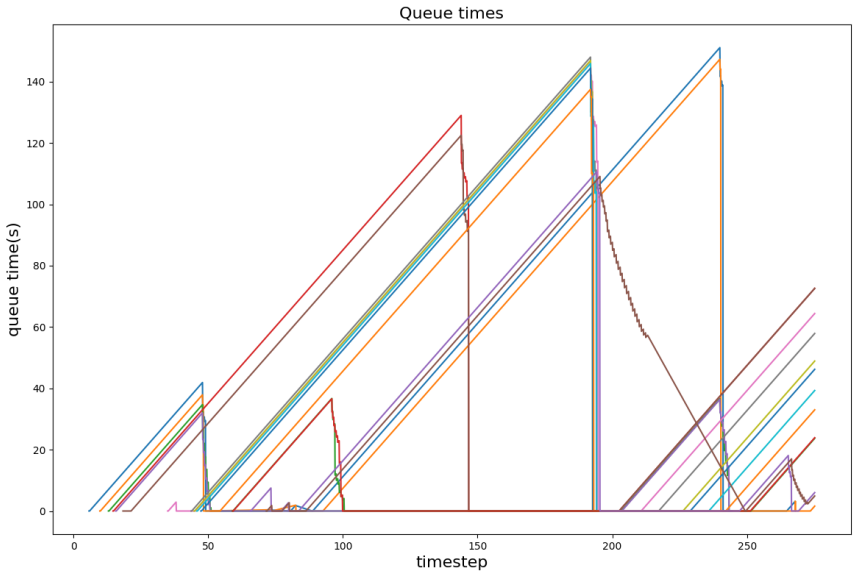


Figure 6.5: Queue times per route per lane

6.3.3.1 Post-Implementation Observations

Traffic flow remains one of the most pivotal metrics in the evaluation of transportation systems. It provides crucial insights into the efficiency and effectiveness of implemented transport changes. One of the primary observations made during our recent evaluations was the significant increase in the number of vehicles, indicating a potential change in traffic dynamics. The alterations made to the traffic light system in our simulation led to a noticeable increase in vehicle counts.Below is a breakdown of the observed changes:

Metric	**Before**	**After**
Total Vehicles	93	245

Table 6.1: Comparison of traffic metrics before and after the implementation of the traffic light algorithm.

To quantify this change, we calculated the percentage increase P using the formula:

$$P = \left(\frac{\text{After} - \text{Before}}{\text{Before}} \right) \times 100\%$$

Inserting our observed values:

$$P = \left(\frac{245 - 93}{93} \right) \times 100\%$$

$$\Rightarrow P \approx 163.44\%$$

Consequently, post the traffic light alterations, there was an impressive rise of approximately 163.44% in traffic flow. Such a marked increase emphasizes the critical role of traffic light patterns in shaping urban traffic dynamics and underscores the need for continuous assessment and optimization.

6.3.3.2 Trip Duration

Comparing Figures 6.6 and 6.7, we can discern the efficacy of the `manage_traffic_lights` algorithm. The reduction in trip duration is a testament to the algorithm's potential to enhance traffic flow and reduce vehicle wait times. This, in turn, could have broader implications, such as reduced fuel consumption and emissions, leading to both economic and environmental benefits as we will approach later in this section.

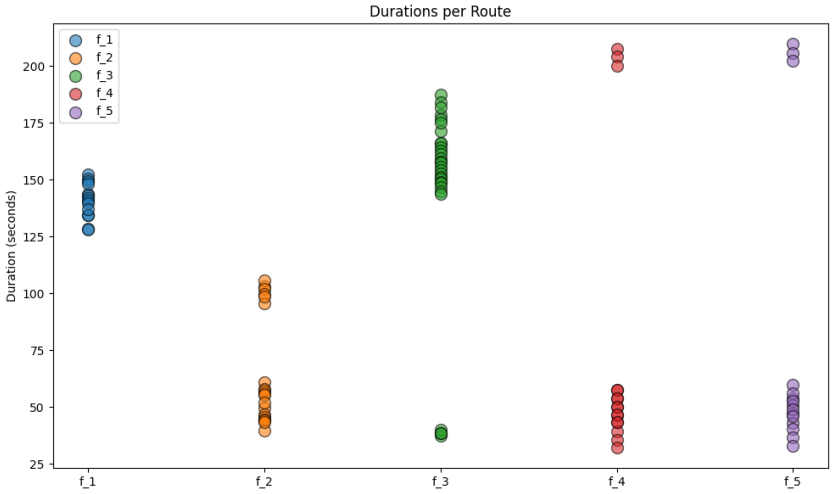


Figure 6.6: Duration per route before algorithm integration.

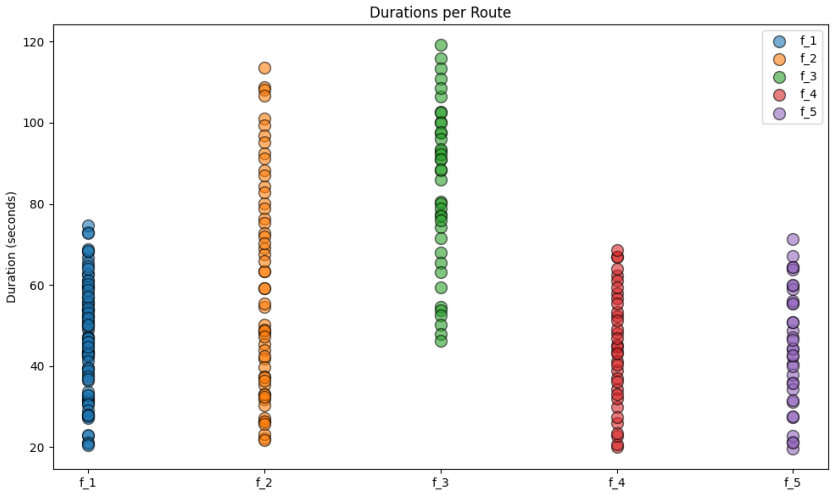


Figure 6.7: Duration per route after algorithm integration.

Route	Average Duration (Before)	Average Duration (After)
f_1	140.82	46.97
f_2	66.2	60.27
f_3	144.87	83.87
f_4	76.3	44.5
f_5	79.1	44.74

Table 6.2: Comparison of median trip duration for each predefined route.

Upon implementing the new traffic light system, a significant change in the average duration for each predefined route was observed. Table 6.2 provides a side-by-side comparison of the average duration before and after the modification.

- For route f_1, there was a dramatic decrease in the average duration by 66%. The duration was reduced from 140.82 seconds to 47 seconds.
- Route f_2 witnessed a decrease of 9% in its average duration, bringing it down from 66.2 seconds to 60 seconds.
- Similarly, route f_3 showed a reduction of 42%, with the duration dropping from 144.87 seconds to 84 seconds.
- Route f_4 experienced a more modest decrease of 42%, transitioning from 76.3 seconds to 44.5 seconds.
- Lastly, route f_5's average duration was decreased by 44%, changing from 79.1 seconds to 44.74 seconds.

Taking the average duration of all routes into account, there was an overall average reduction of approximately 40%. This underlines the significant positive impact the modified traffic light system had on enhancing the efficiency of traffic flow within the simulation.

6.3.4 Queue in lanes analysis

Moreover, another remarkable achievement of our updated system was the drastic reduction in queue time per lane. Previously, the queue time per lane reached peaks of around 140 seconds, which was a significant bottleneck in the traffic flow. With the new system in place, this was trimmed down considerably, with the maximum queue time per lane being slightly over 60 seconds. This change not only ensures a smoother traffic experience but also potentially reduces the environmental impact caused by idling vehicles.

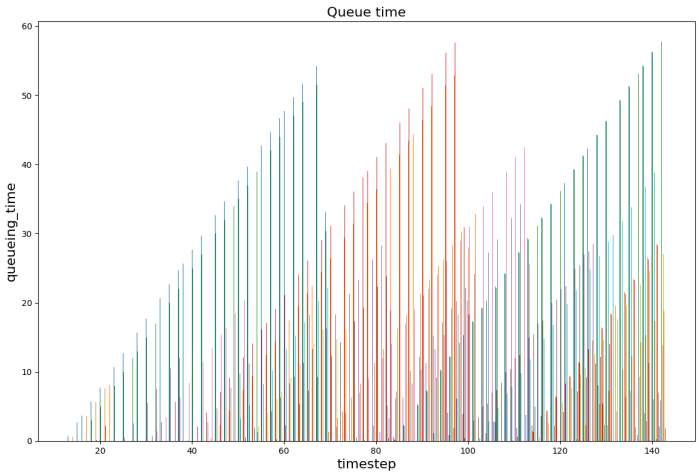


Figure 6.8: Total Queue time for each lane over time

6.3.5 Emissions Analysis

One integral part of our project was to gauge the environmental repercussions.

Since the number of trips truly varies before and after the implementation of our traffic lights algorithm, we shifted our focus to the mean CO_2 emissions. This adjustment in our analysis approach is vital because, while the total emissions provide an overall snapshot, the mean offers a clearer insight into the typical emissions of an individual trip. If the mean emissions decrease, it is an indicator that, on average, each journey post the algorithm's implementation is producing less CO_2. This can be directly attributed to the efficiency of our traffic lights algorithm.

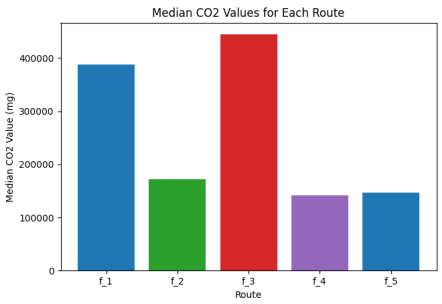


Figure 6.9: CO2 emissions without managing traffic

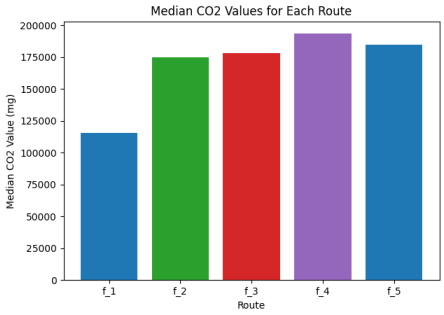


Figure 6.10: CO2 emissions after managing traffic

Analyzing this in detail, we can evidently observe that the emissions have reduced significantly. The range has diminished from 400,000 mg to half of that value 6.10. Moreover, the bars in the visualization now appear more balanced, implying a state of equilibrium in the emissions throughout all the lanes. In further analysis we observed a reduction of about 41.53% in the mean emissions of Co2 per car that accentuates and validates the data we just observed.

6.3.6 Driving Behaviors

To understand the dynamics of different driving conditions, we have classified several driving behaviors based on three core metrics: Tau (reaction time), Sigma (speed deviation), and Impatience. The following table offers a side-by-side comparison as we introduced in section 5.4.3.

Driving Behavior	Tau	Sigma	Impatience
Conservative Driving	2.0	0.2	0.2
Aggressive Driving	0.5	0.8	0.8
Mixed Driving Behavior	1.0	0.5	0.5

Table 6.3: Different scenarios generated based on Tau, Sigma, and Impatience.

In the complex ecosystem of urban transportation, driving behaviors play a pivotal role in dictating traffic flow, safety, and environmental sustainability. To disentangle the multifaceted impacts of various driving behaviors, we've opted for a categorized approach, segmenting our initial analysis based on broad behavioral traits: **Conservative**, **Aggressive**, and **Mixed**.

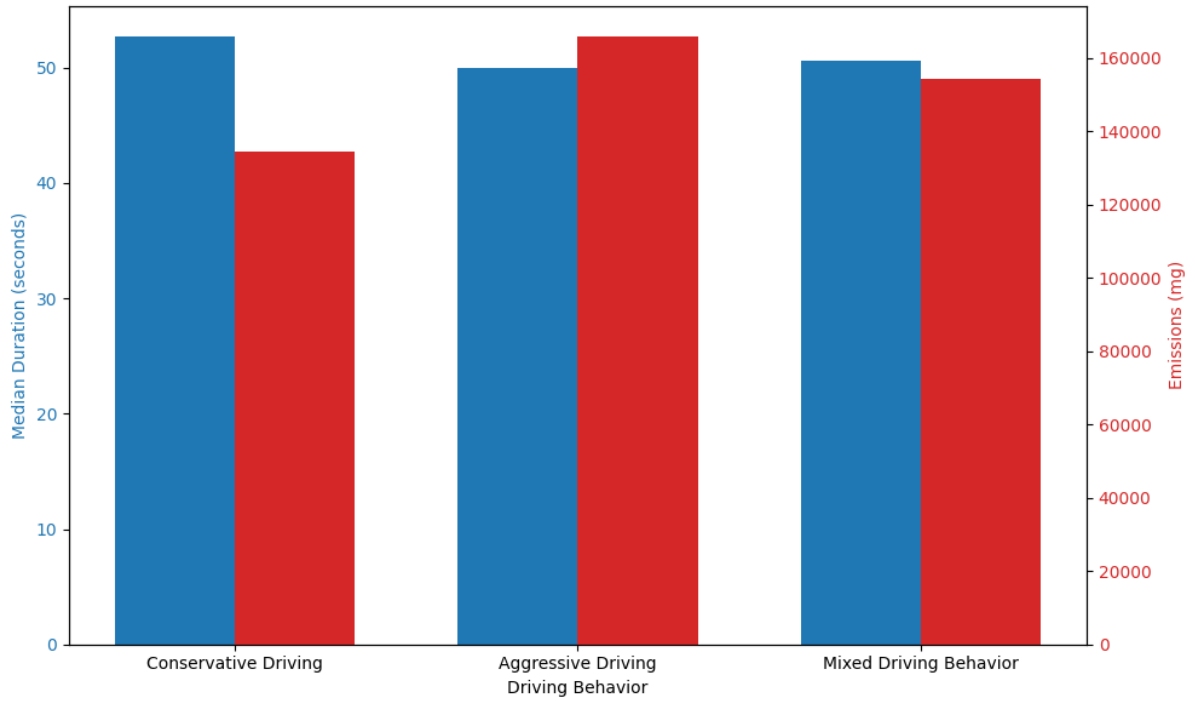


Figure 6.11: Impact of Driving Behavior on Emissions and Trip Duration

Upon evaluating the relationship between driving behaviors and their resulting impacts on emissions and trip duration, distinct patterns come to the forefront. The aggressive driving style, characterized by rapid accelerations, higher speeds, and frequent gear changes, unsurprisingly leads to higher emissions. Such driving dynamics demand more fuel and consequently produce higher amounts of pollutants. Interestingly, this aggressive behavior results in shorter trip

durations, suggesting that the vehicle reaches its destination faster but at an environmental cost.

In contrast, the conservative driving style produces the opposite effect. The results reveal a noticeable reduction in emissions, a testament to the efficiency of maintaining steady speeds, avoiding rapid accelerations, and possibly operating more often in optimal gear ranges. However, this efficiency comes at the price of a longer trip duration, implying that while the journey might be more environmentally friendly, it is less time-efficient.

The mixed driving behavior operates as a midpoint between the two extremes. It exhibits a balanced profile, combining elements of both aggressive and conservative styles. The outcome is a more evened-out emission level and trip duration. Such a mode suggests that it's possible to strike a balance between time efficiency and environmental considerations.

It's essential to underline that while aggressive driving may offer time benefits, it might not always be the best choice from an environmental and, at times, safety perspective. Conversely, while conservative driving appears environmentally superior, it might not always be practical in scenarios where time is a significant factor. The mixed approach seems to offer a pragmatic solution, harmonizing both time and emission concerns, and might be the most judicious choice in varied driving situations.

6.3.7 Sensor Testing

Our initial focus was on leveraging the Radar sensor's capabilities.The data acquired from the radar sensor was subsequently subjected to intricate processing. This transformation involved converting the data points into graphical representations. These points were then color-coded based on the nature of the detected objects(Adapted from Carla sources)[13]. Specifically:

- Blue for points approaching the vehicle.
- Red for points moving away from it.
- White for points static regarding the vehicle.

6.3.8 Vehicle Detection using Light Detection And Ranging (LIDAR)

In the pursuit of ensuring a comprehensive vehicle detection system, we embarked on a journey that integrated the modern marvels of LIDAR technology. This section delves deep into the intricacies of the development and testing phases, shedding light on our methodologies, challenges, and analytical processes.



Figure 6.12: Radar processing

6.3.8.1 LiDAR-based Vehicle Detection

Our system, strategically positioned at the traffic light, was geared to harness the raw power of LIDAR data to meticulously detect and enumerate vehicles cruising beneath. To visualize and subsequently calibrate the sensor data, we employed the robust capabilities of Matplotlib. This allowed us to dynamically render the raw point cloud data, offering a visual representation of vehicles and other environmental entities.



Figure 6.13: LiDAR placement

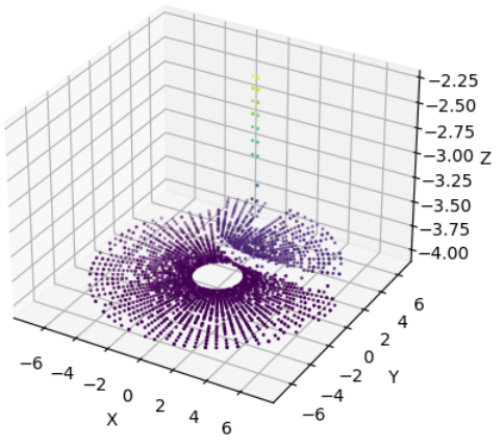


Figure 6.14: LiDAR sensor scanning with no car

The LiDAR sensor calibration was an iterative process, involving numerous adjustments to achieve optimal orientation and positioning. The need for precision was paramount, and through continuous visualization using Matplotlib, we could tweak and perfect the sensor's attributes, such as range, field of view, and orientation relative to the traffic light. The rendered plots acted as our north star, guiding us through areas needing refinement and affirming the correctness of our configurations.

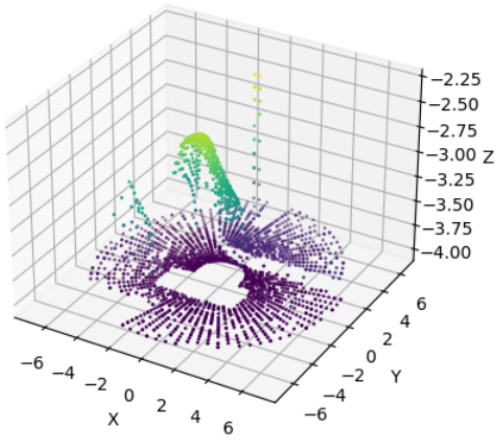


Figure 6.15: LiDAR sensor scanning cars

6.3.8.2 Benchmark: `tripinfo.xml`

We once again to validate our work, a benchmark was imperative. The `tripinfo.xml` file served this purpose, acting as a yardstick against which our system's efficiency could be gauged. This file diligently logs vehicles as they conclude their predefined route, which, in our experimental

setup, was earmarked to the next lane location immediately succeeding the LiDAR-equipped traffic light.

6.3.8.3 Detection results and validation

When comparing our LiDAR-based system's results with the standard benchmark, we noticed a difference of 6 vehicles. Our system identified 256 vehicles.And as we previsouly have stated and confirmed with the output tripinfo.xml that outputs 245 vehicles and only counts vehicles that have finished their journey. So, the extra vehicles our LiDAR detected that weren't in the tripinfo.xml are likely those that passed our detection point but hadn't reached their destination, that are marked as red in the following image 6.16.

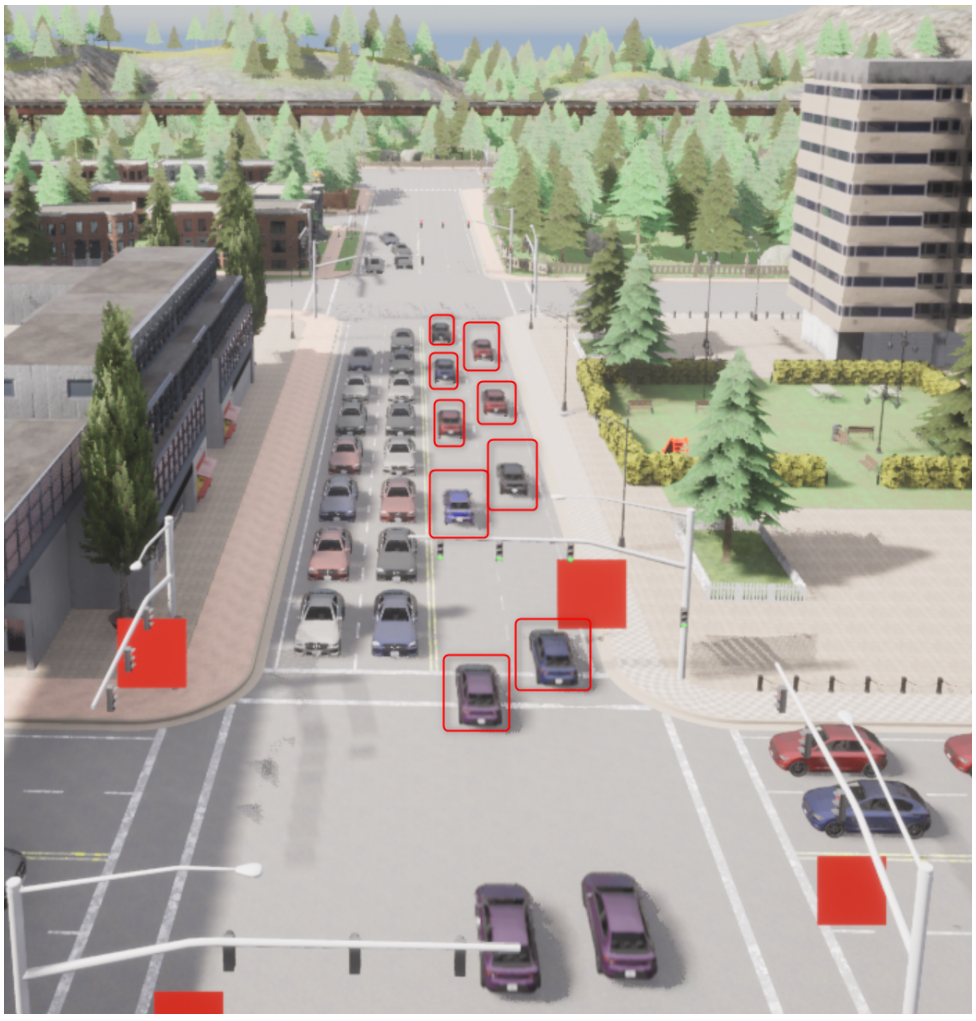


Figure 6.16: Overview of last simulation step

A detailed comparison between the LiDAR detection method and the tripinfo.xml benchmark reveals valuable insights. Both systems exhibit a commendable consistency in monitoring vehicular movements, highlighting the alignment of data across different simulation models. Although there was a slight discrepancy with two vehicles being double-identified, this serves as a promising

benchmark. It suggests potential enhancements in vehicle detection by integrating cameras and other sensors in the future.

6.3.9 Overall System Performance

The combination of these evaluations provides a comprehensive understanding of the overall system performance. The system showcased adaptability to diverse conditions, efficient traffic flow management, and robust sensor capabilities. The findings contribute valuable insights for further refinement and optimization of the autonomous driving system.

6.4 Conclusion

In summary, the testing and analysis phase has provided significant insights into the performance, capabilities, and potential areas of improvement for the project. These evaluations not only validate the initial objectives and criteria set out at the beginning but also highlight the robustness and adaptability of the developed solution in various scenarios.As we transition to the next phase, we carry forward the lessons learned, ensuring that the project is not only theoretically sound but also validated.

Chapter 7

Conclusions

This chapter embarks on a journey through the integration of two formidable simulation tools, Car Learning to Act (CARLA) and Simulation of Urban Mobility (SUMO), pivotal in the realm of urban traffic management and simulation. As urban environments evolve in complexity, the demand for sophisticated simulation tools becomes imperative, providing a comprehensive perspective on potential scenarios.

Our narrative begins by contextualizing the project within the broader landscape of IoT and Smart Cities. In the era of IoT, we leverage its capabilities to collect information across diverse contexts. This data becomes the cornerstone for smart cities, driving enhancements in the quality of life for their citizens.A significant stride in our work lies in the integration of digital twins, facilitating model testing without real-world deployment. This not only optimizes model efficiency but also minimizes impact, a crucial aspect in the context of urban environments.

The integration of SUMO and CARLA provides a holistic representation of urban traffic, allowing for nuanced studies on traffic dynamics and vehicle interactions. The Light Detection And Ranging (LIDAR) sensor's dynamic positioning and clustering algorithms enable precise detection of vehicles, contributing to the authenticity of the simulated traffic. The utility-based algorithm considers lane congestion, prioritizes waiting vehicles, and dynamically adjusts traffic light phases, leading to more efficient traffic flow. The Application programming interface (API) allows for dynamic changes in weather attributes, providing a valuable tool for testing the robustness of autonomous algorithms in various weather scenarios.

Driving behaviors emerged as key factors influencing emissions and trip duration. The nuanced interplay between aggressive and conservative driving styles reveals trade-offs between efficiency and sustainability in urban driving environments. A balanced approach, as seen in mixed driving scenarios, offers insights into potentially optimized driving strategies for real-world applications.

Our innovative approach to dynamic traffic light management injects real-time data and analysis into the urban environment, simulating a reactive infrastructure. This not only underscores the potential of smart cities but also exemplifies adaptive infrastructure responding

to immediate needs.

Central to our achievements is the development of a robust Application programming interface (API) seamlessly integrating CARLA and SUMO. This Python-based API stands as a testament to the versatility and dynamism of the language, facilitating rapid development, prototyping, and seamless integration. Leveraging Flask and lock mechanisms ensured not only functionality but also concurrent access handling and real-time request responsiveness.

In conclusion, our project extends beyond a demonstration of technological capabilities. It signifies a paradigm shift in how we perceive, interpret, and manage urban traffic scenarios. Through the harmonious integration of sophisticated tools and data-driven strategies, the future of urban traffic management appears promising, efficient, and increasingly intelligent.

Looking ahead, as both CARLA and SUMO evolve, the door opens for further exploration, integration, and improvements in the dynamic field of urban traffic simulation. This project serves as a reminder of the exciting possibilities that emerge when cutting-edge tools are harmoniously integrated for a larger purpose—creating smarter, safer, and more efficient urban environments.

Bibliography

[1] Z. Allam and Z. A. Dhunny. On big data, artificial intelligence and smart cities. *Cities*, 89:80–91, 2019.

[2] P. Alvarez Lopez, M. Behrisch, L. Bieker-Walz, J. Erdmann, Y.-P. Flötteröd, R. Hilbrich, L. Lücken, J. Rummel, P. Wagner, and E. Wießner. Microscopic traffic simulation using sumo. In *2019 IEEE Intelligent Transportation Systems Conference (ITSC)*, pages 2575–2582. IEEE, November 2018.

[3] J. M. Anderson, N. Kalra, K. D. Stanley, P. Sorensen, C. Samaras, and T. A. Oluwatola. *Autonomous Vehicle Technology: A Guide for Policymakers*. RAND Corporation, Santa Monica, CA, 2016.

[4] L. Atzori, A. Iera, and G. Morabito. The internet of things: A survey. *Computer Networks*, 54(15):2787–2805, 2010.

[5] M. Bilal, L. O. Oyedele, J. Qadir, K. Munir, S. O. Ajayi, O. O. Akinade, H. A. Owolabi, H. A. Alaka, and M. Pasha. Big data in the construction industry: A review of present status, opportunities, and future trends. *Advanced Engineering Informatics*, 30(3):500–521, 2016.

[6] F. Biljecki, J. Stoter, H. Ledoux, S. Zlatanova, and A. Çöltekin. Applications of 3d city models: State of the art review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889, 2015.

[7] S. Boschert, C. Heinrich, and R. Rosen. Next generation digital twin. 05 2018.

[8] L. D. Burns, W. C. Jordan, and B. A. Scarborough. Transforming personal mobility. *The Earth Institute, Columbia University*, 2013.

[9] B. Chen and H. Cheng. A review of the applications of agent technology in traffic and transportation systems. *Intelligent Transportation Systems, IEEE Transactions on*, 11:485 – 497, 07 2010.

[10] T. Deng, K. Zhang, and Z.-J. M. Shen. A systematic review of a digital twin city: A new pattern of urban governance toward smart cities. *Journal of Management Science and Engineering*, 6(2):125–134, 2021.

[11] B. v. A. Dimitris Milakis and B. van Wee. Policy and society related implications of automated driving: A review of literature and directions for future research. *Journal of Intelligent Transportation Systems*, 21(4):324–348, 2017.

[12] N. Dlodlo, O. Gcaba, and A. Smith. Internet of things technologies in smart cities. In *2016 IST-Africa Week Conference*, pages 1–7, 2016.

[13] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, and V. Koltun. CARLA: An open urban driving simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*, pages 1–16, 2017.

[14] D. Gavalas, P. Nicopolitidis, A. Kameas, P. Bellavista, L. Lambrinos, and B. Guo. Smart cities: Recent trends, methodologies, and applications. *Wireless Communications and Mobile Computing*, 2017:1–2, 10 2017.

[15] M. Grieves and J. Vickers. *Digital Twin: Mitigating Unpredictable, Undesirable Emergent Behavior in Complex Systems*, pages 85–113. Springer International Publishing, Cham, 2017.

[16] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013. Including Special sections: Cyber-enabled Distributed Computing for Ubiquitous Cloud and Network Services Cloud Computing and Scientific Applications — Big Data, Scalable Analytics, and Beyond.

[17] G. Hancke and B. Silva. The role of advanced sensing in smart cities. *Sensors (Basel, Switzerland)*, 13:393–425, 12 2012.

[18] I. A. T. Hashem, V. Chang, N. B. Anuar, K. Adewole, I. Yaqoob, A. Gani, E. Ahmed, and H. Chiroma. The role of big data in smart city. *International Journal of information management*, 36(5):748–758, 2016.

[19] H. Herath and M. Mittal. Adoption of artificial intelligence in smart cities: A comprehensive review. *International Journal of Information Management Data Insights*, 2(1):100076, 2022.

[20] R. Hussain and S. Zeadally. Autonomous cars: Research results, issues, and future challenges. *IEEE Communications Surveys Tutorials*, 21(2):1275–1313, 2019.

[21] G. Karimi. Introduction to yolo algorithm for object detection. Accessed: 26 September 2023.

[22] R. Lea and M. Blackstock. City hub: A cloud-based iot platform for smart cities. In *2014 IEEE 6th International Conference on Cloud Computing Technology and Science*, pages 799–804, 2014.

[23] P. Li, A. Kusari, and D. J. LeBlanc. A novel traffic simulation framework for testing autonomous vehicles using SUMO and CARLA. *CoRR*, abs/2110.07111, 2021.

[24] P. Liu and P. Zhenghong. China's smart city pilots: A progress report. *Computer*, 47:72–81, 10 2014.

[25] Y. Lu, C. Liu, K. I.-K. Wang, H. Huang, and X. Xu. Digital twin-driven smart manufacturing: Connotation, reference model, applications and research issues. *Robotics and Computer-Integrated Manufacturing*, 61:101837, 2020.

[26] F. Manuri and A. Sanna. A survey on applications of augmented reality. *ACSIJ Advances in Computer Science: an International Journal*, 5(1):18–27, 2016.

[27] S. R. Newrzella, D. W. Franklin, and S. Haider. 5-dimension cross-industry digital twin applications model and analysis of digital twin classification terms and models. *IEEE Access*, 9:131306–131321, 2021.

[28] A. Razmjoo, A. H. Gandomi, M. Pazhoohesh, S. Mirjalili, and M. Rezaei. The key role of clean energy and technology in smart cities development. *Energy Strategy Reviews*, 44:100943, 2022.

[29] T. Ruohomäki, E. Airaksinen, P. Huuska, O. Kesäniemi, M. Martikka, and J. Suomisto. Smart city platform enabling digital twin. In *2018 International Conference on Intelligent Systems (IS)*, pages 155–161, 2018.

[30] T. Ruohomäki, E. Airaksinen, P. Huuska, O. Kesäniemi, M. Martikka, and J. Suomisto. Smart city platform enabling digital twin. In *2018 International Conference on Intelligent Systems (IS)*, pages 155–161, Sep. 2018.

[31] P. Seuwou, E. Banissi, and G. Ubakanma. *The Future of Mobility with Connected and Autonomous Vehicles in Smart Cities*, pages 37–52. Springer International Publishing, Cham, 2020.

[32] A. Sharif, J. Li, M. Khalil, R. Kumar, M. I. Sharif, and A. Sharif. Internet of things — smart traffic management system for smart cities using big data analytics. In *2017 14th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, pages 281–284, 2017.

[33] Y. Shi, Z. Liu, Z. Wang, J. Ye, W. Tong, and Z. Liu. An integrated traffic and vehicle co-simulation testing framework for connected and autonomous vehicles. *IEEE Intelligent Transportation Systems Magazine*, 14(6):26–40, 2022.

[34] H. Wang, X. Chen, F. Jia, and X. Cheng. Digital twin-supported smart city: Status, challenges and future research directions. *Expert Systems with Applications*, 217:119531, 2023.

[35] J. Wang, C. Jiang, K. Zhang, T. Q. S. Quek, Y. Ren, and L. Hanzo. Vehicular sensing networks in a smart city: Principles, technologies and applications. *IEEE Wireless Communications*, 25(1):122–132, 2018.

[36] R. Xu, Y. Guo, X. Han, X. Xia, H. Xiang, and J. Ma. Opencda: an open cooperative driving automation framework integrated with co-simulation. In *2021 IEEE International Intelligent Transportation Systems Conference (ITSC)*, pages 1155–1162. IEEE, 2021.

[37] H. Yang, Y. Shen, M. Hasan, D. Perez, and J. Shull. Framework for interactive m3 visualization of microscopic traffic simulation. *Transportation Research Record*, 2672(44):62–71, 2018.

[38] C. Yin, Z. Xiong, H. Chen, J. Wang, D. Cooper, and B. David. A literature survey on smart cities. *SCIENCE CHINA Information Sciences*, 58(100102):1–18, 2015. Received April 3, 2015; accepted May 27, 2015; published online August 25, 2015.

[39] R. Zhang, F. Wang, J. Cai, and et al. Digital twin and its applications: A survey. *Int J Adv Manuf Technol*, 123:4123–4136, 2022.

[40] W. Zhang, S. Guhathakurta, and E. B. Khalil. The impact of private autonomous vehicles on vehicle ownership and unoccupied vmt generation. *Transportation Research Part C: Emerging Technologies*, 90:156–165, 2018.

[41] Z. Zhang, F. Wen, Z. Sun, X. Guo, T. He, and C. Lee. Artificial intelligence-enabled sensing technologies in the 5g/internet of things era: from virtual reality/augmented reality to the digital twin. *Advanced Intelligent Systems*, 4(7):2100228, 2022.

[42] J. Zhao, H. Xu, H. Liu, J. Wu, Y. Zheng, and D. Wu. Detection and tracking of pedestrians and vehicles using roadside lidar sensors. *Transportation Research Part C: Emerging Technologies*, 100:68–87, 2019.