POLITECNICO DI TORINO

MASTER'S DEGREE IN CINEMA AND MEDIA ENGINEERING

Master's Degree Thesis

Digital Shadow & Virtual Worlds for VR Driving in CARLA

******

Supervisors:
Prof. Andrea Bottino
Prof. Francesco Strada

Candidate:
Andrea Spanu

Politecnico di Torino

April 2025

To my loving family, for their unwavering support and for always pushing me to do my best. To those who have always seen my success as their own, thank you for being a part of this journey.

Acknowledgements

I would like to express my sincere gratitude to Francesca Matrone from the DIATI department of Politecnico for providing the point cloud data of the municipality of Mappano, as well as for her kindness and patience in explaining how to handle the data.

I am also deeply grateful to my supervisors for their invaluable support and guidance throughout the duration of my thesis.

Lastly, thanks to my housemates for not only tolerating my endless complaints but also for surviving them with impressive patience and resilience.

Abstract

Evaluating rare but critical driving situations, such as hazardous road events or complex traffic interactions, is challenging yet essential. Physical tests for every possible scenario are expensive, difficult to manage reliably, and potentially dangerous. Consequently, virtual simulations have become fundamental resources for research in autonomous driving, traffic modeling, and safety analysis. Researchers take advantage of these environments to replicate diverse driving scenarios. Sensor responses allow them to analyze vehicle behavior in a controlled and highly adaptable setting.

Among the virtual simulation tools, CARLA (Car Learning to Act) stands out as an open-source driving simulator widely used for autonomous vehicle research and other multidisciplinary studies. Thanks to its realistic physics engine, customizable sensors and high-quality rendering, CARLA is also used in areas such as computer vision for neural network training, robotics through integration with the Robot Operating System (ROS), human-machine interaction to analyze human behavior in autonomous driving scenarios and urban planning. CARLA uses physically based rendering (PBR) techniques in its rendering pipeline to achieve photorealistic visuals while still maintaining real-time performance. Achieving the right balance between realism and efficiency becomes essential to create meaningful and immersive simulations.

This thesis explores methodologies for generating maps that are compatible with the CARLA environment, focusing on two key objectives: achieving a precise reconstruction of real-world environments through geospatial data and enhancing visual realism to improve simulation immersion. The first approach focuses on 'Digital Shadows', aiming to achieve a high-fidelity representation of real-world environments by leveraging geospatial data sources such as LiDAR scans and OpenStreetMap (OSM). These datasets provide precise topographic and infrastructural details, ensuring that the generated maps closely match real-world road networks

and layouts. The second approach prioritizes visual realism and immersion, utilizing Unreal Engine assets to enhance graphical quality and the overall simulation experience. More generally, achieving high spatial accuracy is based on geospatial data acquisition techniques, such as LiDAR scanning, to capture real-world structures with precision. On the other hand, enhancing the visual fidelity of virtual assets requires procedural generation and optimization methods to achieve both realism and performance.

Table of Contents

List of Figures

Nomenclature

Acronyms / Abbreviations

ADAS	Advanced Driver Assistance Systems
AV	Autonomous Vehicle
BIM	Building Information Modeling
DT	Digital Twin
GIS	Geographic Information System
HDD	Head-Down Display
HLOD	Hierarchical Level of Detail
HMD	Head-Mounted Display
HUD	Head-Up Display
IFC	Industry Foundation Class
IoT	Internet of Things
IPQ	Igroup Presence Questionnaire
LOD	Levels of Detail
OSM	OpenStreetMap
SoS	System of Systems
SSQ	Simulator Sickness Questionnaire

TSR	Temporal Super Resolution
UE	Unreal Engine
UEQ	User Experience Questionnaire
VR	Virtual Reality
VSM	Virtual Shadow Maps

Chapter 1

Introduction

Virtual simulations are computational environments designed to replicate real-world scenarios with varying levels of accuracy and interactivity. They are increasingly integral across a broad range of fields, from aerospace engineering to urban planning, with particularly significant applications in the automotive industry. In driving research, simulations enable controlled replications of road conditions, vehicle dynamics, and driver behavior without the inherent risks and constraints of physical testing. This is especially crucial in the study of autonomous driving and human-machine interaction, where simulations provide a safe and versatile platform to evaluate complex and potentially hazardous situations that would be difficult or impossible to replicate in the real world.

One of the primary advantages of virtual simulations is their ability to create highly flexible and reproducible testing environments. Unlike traditional physical experiments, where factors such as traffic, weather, and road characteristics are often unpredictable, simulations allow precise control over these elements, ensuring consistency and repeatability across trials. This flexibility is invaluable for stress-testing systems under extreme or rare conditions that might not typically be encountered during real-world driving. Furthermore, in autonomous vehicle research, virtual simulations provide the opportunity to expose AI models to a wide variety of scenarios, enabling the training and validation of autonomous systems without the need for extensive physical testing.

However, to achieve optimal results, the realism of virtual simulations is paramount. High-fidelity simulations - which accurately model road geometries, traffic

patterns, vehicle physics, and driver interactions - are critical to ensuring that virtual testing provides valid, actionable data for real-world applications.

Studies on the effectiveness of virtual simulation-based training emphasize the importance of the simulator's realism, which can significantly influence the sense of presence experienced by the trainee. As discussed in the study *The Relationship between Presence and Performance in Virtual Simulation Training* by Jonathan A. Stevens and J. Peter Kincaid (2015) [1], a higher sense of presence in virtual environments is closely linked to improved performance outcomes. Their analysis shows that the relationship between presence and performance is notably influenced by simulator characteristics, particularly visual immersion. Stevens and Kincaid found that higher visual immersion led to both higher presence and better performance from trainees. This finding is crucial, especially when considering the potential of virtual simulations to enhance training transfer, the ability to apply learned skills to real-world situations. In their study, the strongest positive correlation between presence and performance was observed in the treatment where visual immersion most significantly affected the trainees' experience, underscoring the value of realism in fostering effective training environments.

This concept is crucial for driving simulations, especially when creating environments for autonomous vehicle testing and human-machine interaction. To enhance the sense of presence, simulations need to integrate immersive technologies such as high-fidelity rendering, real-time physics, and interactive models. These elements help create a simulation that not only replicates real-world scenarios accurately but also engages the user in a way that maximizes their sense of immersion and interaction. This is particularly relevant in the context of Digital Twin or Digital Shadow methodologies, where high-definition spatial data (e.g., LiDAR point clouds and GIS-based road networks) can significantly enhance the realism of the virtual environment, leading to a stronger connection between the trainee and the simulated world.

This thesis, therefore, aims to explore workflows for creating highly **realistic and immersive environments**, as well as **environments derived from real-world data**, tailored for driving simulations in virtual reality (VR). The goal is to identify efficient, automated methods for generating simulation-ready maps that integrate seamlessly into driving simulators. By improving the quality and applicability of virtual simulations, particularly in the context of autonomous vehicle testing and

human-machine interaction, this research seeks to advance simulation-based training and development in the automotive industry.

1.1 CARLA as a Tool for Simulation

CARLA (Car Learning to Act) is an open-source simulator specifically designed for autonomous driving research in urban environments. Its flexibility and realism make it a powerful tool for testing autonomous systems in diverse and complex scenarios. CARLA allows researchers to evaluate perception, planning, and control strategies by providing detailed sensor simulations, dynamic traffic interactions, and customizable environmental conditions.

Given its capabilities, CARLA serves as the primary simulation platform for this study. The effectiveness of virtual environments in autonomous driving research depends not only on their visual fidelity but also on the accuracy of road networks, the responsiveness of traffic systems, and computational efficiency—especially in Virtual Reality applications. Ensuring seamless integration of high-quality environments into CARLA is therefore crucial for maintaining simulation performance while enhancing immersion and interactivity.

This research leverages CARLA's framework to assess the feasibility of automated workflows for generating digital shadows and realistic urban environments, optimizing them for VR-based driving simulations. By refining these methods, the study aims to contribute to more robust and adaptable simulation ecosystems, supporting advancements in both autonomous vehicle testing and human-machine interaction research.

The following chapters will delve into the methodologies for generating these environments, detailing the processes of data acquisition, map generation, and integration into CARLA. Additionally, we will explore the implementation of force feedback to enhance immersion in virtual reality and examine the structure of the conducted tests.

Chapter 2

State of the Art

This chapter provides an overview of the key concepts and technologies that are essential for understanding immersive driving simulators, with a particular focus on the interrelationship between the concepts of Digital Twin and Digital Shadow. A comprehensive understanding of Digital Twin technology is crucial for grasping the role of Digital Shadow, which lies at the heart of this research. The chapter explores various related fields, including city building, driving simulations, and the integration of Virtual Reality, to lay the groundwork for the development of advanced simulation systems. Additionally, the use of CARLA and Unreal Engine as tools for creating realistic virtual environments will be discussed, highlighting their role in enhancing simulation fidelity and immersion.

2.1 Digital Twin: Definition and Applications

The *Digital Twin* (DT) is a rapidly evolving technology that has become essential in the context of digital transformation and intelligent system upgrades. By integrating data and models, the DT enables functionalities such as monitoring, simulation, prediction, and optimization. At the heart of this concept, Digital Twin modeling ensures an accurate representation of physical entities, allowing it to provide functional services and meet application-specific requirements.

According to the IT Glossary by Gartner, a leading research and advisory company in the field of information technology, "A digital twin is a digital representation of a real-world entity or system. The implementation of a digital twin is an encapsulated software object or model that mirrors a unique physical object, process, organization, person or other abstraction. Data from multiple digital twins can be aggregated for a composite view across a number of real-world entities, such as a power plant or a city, and their related processes."[2]

In 2020, Gartner also published its "Hype Cycle for Emerging Technologies" identifying five key trends driving this hype cycle, including Digital Twins. Brian Burke, Research VP at Gartner, stated, "This Hype Cycle highlights technologies that will significantly affect business, society, and people over the next five to 10 years."[3]

In 2022, Tao et al. [4] analyzed 296 academic papers to explore the various application areas of Digital Twin technology.

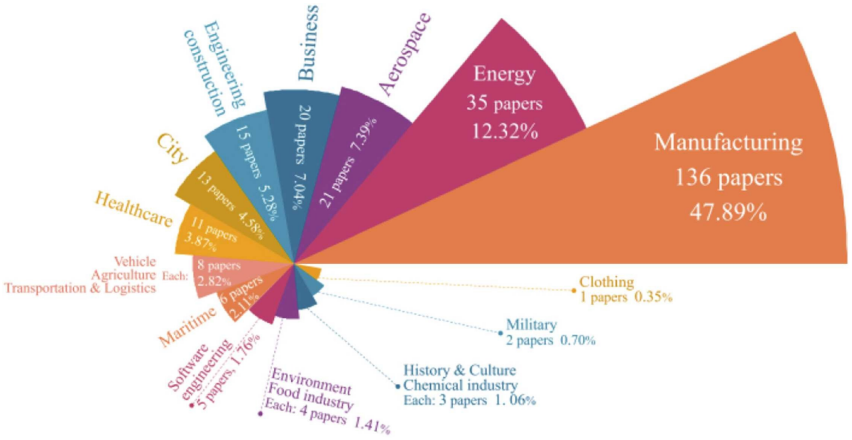

Fig. 2.1 Application Domains of Digital Twin Models [4]

The graph shown in the figure indicates that DT models cover a wide range of disciplines, promoting interdisciplinary research. Overall, Digital Twins (DTs) have great potential to change traditional fields, support emerging areas, and advance research in complex domains.

2.1.1 Differences from Digital Models

It is important to define the difference between a Digital Twin and a model. Wright and Davidson [5] analyzed various definitions to highlight three important components in the DT of an object:

- a representation or model of a physical object,
- a set of evolving data related to the object,
- a process for continuously updating or adjusting the model based on the data.

While the model in a Digital Twin does not need to be inherently data-driven, it must be capable of producing results that correspond directly to measurable quantities. This allows the updating process to rely on real-world data. Additionally, the model will often incorporate other measured factors, such as boundary conditions, loads, or material properties.

A significant advantage of the Digital Twin approach is its ability to account for changes over time, thanks to the use of evolving data. A validated model offers a snapshot of how an object behaves at a specific point in time, but integrating this model into a DT allows for extended use, covering longer periods where both the object and its behavior may change significantly.

A key factor to highlight is that *a Digital Twin must always correspond to an actual physical object.* If there is no real-world counterpart, what remains is simply a model.

Digital Twins are most effective when an object changes over time, rendering the initial model of the object inaccurate, and when data related to these changes can be captured and correlated. If the object remains largely unchanged or if the associated data cannot be collected, the utility of a DT diminishes significantly.

Digital Twins can be classified based on the hierarchical nature of physical entities and their virtual representations. Tao et al. [4] suggest that DT models can be generally divided into three levels: *unit level*, *system level*, and *system of systems (SoS) level*, according to their structure and functionality.

In urban applications, individual constructions are considered at the unit level, while larger complexes, such as industrial parks or residential districts, belong to the system level. At the highest level, the entire city is represented at the SoS level. Most Digital Twin models for cities are focused on the unit and SoS levels.

City-Scale Digital Twin: A Practical Example

Wright and Davidson [5] highlight the use of city-scale Digital Twins as a practical example for testing autonomous systems, especially the decision-making processes in autonomous vehicles (AVs). These vehicles face a broad range of potential hazards on the road. Many of the most dangerous situations, however, are rare, making them unlikely to occur during everyday driving. Conducting physical tests for every possible scenario would be extremely expensive, difficult to manage reliably, and potentially dangerous. For this reason, using a Digital Twin to simulate these tests is a much more efficient and safer approach. In this context, a DT would incorporate:

- an accurate and regularly updated model of the driving environment, including vehicles, pedestrians, weather conditions, and other environmental factors,
- simulations of how the vehicle's sensors would respond, based on real-world data from sensor tests,
- a model of the vehicle's physical responses to driving commands (e.g., steering, braking), considering road surface conditions,
- and the AV's decision-making algorithms.

CARLA implements several of these functionalities, offering a high-fidelity simulation environment for testing autonomous vehicles. It allows for the simulation of dynamic traffic scenarios, pedestrian interactions, and various weather conditions. However, while it provides realistic sensor modeling and driving physics, it does not fully account for complex real-time physical interactions such as variable road surface adhesion.

2.1.2 Digital Twin Modelling

Tao et al. [6] introduced a five-dimensional Digital Twin model, consisting of the following components: physical entity, virtual model, connection, data, and service. Among these, the virtual model is further divided into four distinct sub-models, each of which contributes to the accurate representation of the physical entity. These sub-models are:

Geometric Model defines the shape, structure, and spatial position of the entity, balancing fidelity and efficiency for applications like motion analysis and virtual interaction.

Physical Model supports quality control and property analysis, distinguishing between **static models** (fixed properties) and **dynamic models** (time-dependent phenomena like heat transfer).

Behavioral Model is designed to capture the various behaviors (sequential, random, periodic) of physical entities, addressing uncertainties and anomalies that may affect accuracy.

Rule Model uncovers implicit knowledge and highlights the evolutionary trends and patterns of physical entities over time. It extracts knowledge from life cycle data and expert insights to reveal patterns and evolutionary trends, enhancing decision-making.

By integrating these sub-models, the virtual representation closely mirrors the physical entity, ensuring accurate simulation and interaction within the Digital Twin.

Framework for Digital Twin Development

In a separate study, Tao et al. [7] introduced a theoretical framework for Digital Twin modeling that dissects and examines the various components involved in the Digital Twin process. Their framework encompasses six key aspects: model construction, model assembly, model fusion, model verification, model modification, and model management. Each of these elements plays a crucial role in ensuring that DT models are not only accurate representations of their physical counterparts, as discussed in Subsection 2.1.1 regarding the distinctions between models and Digital Twins, but also capable of adapting to changing conditions and requirements.

Model construction integrates multidisciplinary knowledge to create unit-level DT models, which can later be expanded.

Model assembly combines unit models into larger structures, balancing complexity, accuracy, and efficiency.

Model fusion merges geometric, physical, behavioral, and rule-based dimensions, ensuring coherence across disciplines.

Model verification ensures that the DT accurately reflects the physical system by comparing outputs under identical conditions.

Model modification adjusts parameters when verification reveals deviations, focusing on relevant and error-prone elements.

Model management oversees DT models and knowledge bases, ensuring data integrity, access control, and efficient usage.

2.1.3 Conclusions on Digital Twins

In summary, Digital Twins serve a broad range of applications, with varying requirements for computational speed and update frequency. While some implementations demand real-time synchronization with their physical counterparts, others operate effectively within longer update cycles, allowing for greater model complexity and detail. Understanding the hierarchical and dimensional structures of Digital Twins is essential for selecting the most suitable approach for a given application.

However, the focus of this thesis is not on a fully bidirectional Digital Twin but rather on a Digital Shadow, where data flows unidirectionally from the physical world to its virtual representation. This approach ensures an accurate and detailed reconstruction within CARLA, leveraging external datasets to generate realistic urban environments. The following section will introduce and define this concept in greater depth, outlining its role in the proposed framework.

2.2 From City Digital Twin to Digital Shadow

As urban environments become more complex, information technologies are increasingly central to managing and optimizing city systems. Traditional city management models are giving way to innovative approaches driven by Big Data [1], which enables real-time analysis and enhances decision-making accuracy. Unlike static statistical samples, Big Data continuously captures dynamic patterns in urban behaviors and interactions, spanning areas such as population, economic growth, construction, and infrastructure.

The integration of Digital Twin technology further amplifies this approach, enabling the development of real-time virtual models of city systems. DTs collect and process data from distributed "Internet of Things" (IoT) sensors throughout the urban landscape, creating a continuously updated digital counterpart of the city. This digital representation supports predictive analysis of infrastructure, provides insights into the interactions between people and transportation, and answers complex "what-if" scenarios. By simulating urban dynamics under various economic, environmental, and social conditions, DTs offer a foundation for understanding how "smart cities" can function optimally and for identifying potential vulnerabilities in city operations.

According to Ivanov et al. [9], a Smart City represents a strategic framework that leverages data and digital technologies to promote sustainability, enhance citizen welfare, and support economic growth within urban environments. This concept envisions a unified space where core urban infrastructure — such as environmental systems, emergency management, traffic, and power — interconnect seamlessly, fostering synergy both among existing functions and with future technological advancements.

Yet, achieving a full-scale visualization of an entire city through a digital twin presents substantial hurdles, particularly concerning the thoroughness and precision of urban information. Numerous digital twin models developed to date exhibit issues with accuracy, completeness, and the quality of graphical representations. Additionally, efforts to mitigate the limitations of sensory data using participatory sensing and crowdsourcing have highlighted concerns, including localization inaccuracies and

[1] "Big data is high-volume, high-velocity and/or high-variety information assets that demand cost-effective, innovative forms of information processing that enable enhanced insight, decision making, and process automation." [8]

the potential for unreliable or unverified information. These flaws not only detract from the visual fidelity of the model but also hinder situational awareness, which is crucial for interpreting urban dynamics in real time.

Furthermore, a comprehensive understanding of a city's current condition is often compromised by the omission of contextual elements and non-physical systems, such as social, economic, and political frameworks. The intricate interactions among people, infrastructure, and technology are frequently overlooked, which can significantly undermine the predictive and planning capabilities of digital twins. Without sufficient situational awareness, the effectiveness of urban planning and forecasting diminishes, and an incomplete or inaccurate city model may lead to flawed or suboptimal decision-making. Consequently, the usefulness of a digital twin in urban analysis and future scenario planning depends heavily on the accuracy and richness of the data it integrates.

2.2.1 Conceptualizing the City Digital Twin

The concept of a City Digital Twin entails a comprehensive network of interconnected digital models that simulate various aspects of urban operations and development. These models are engineered to dynamically adjust in real time to the actual conditions of urban infrastructure, harnessing data from diverse sources. Examples of such data streams include [9] information about the movement of residents and vehicles — spanning private, commercial, and public transportation — and congestion metrics from traffic monitoring systems. Additionally, real-time environmental data collected by smart sensors provide measurements of air temperature, humidity, pollution levels, noise, radiation, and water quality, all linked to specific geographic locations. Insights from outdoor cameras contribute by offering traffic conditions, road quality assessments, and event detection, while public portals, meteorological services, and business reports further enrich intelligent data analysis.

However, developing a holistic City Digital Twin requires addressing several complexities. One of the primary challenges lies in ensuring a high degree of integration among the diverse urban domains and information sources. A City Digital Twin must accommodate varying levels of detail and fidelity across different parts of the city. For instance, buildings from different centuries may need to be scanned and integrated at different fidelity levels, often as a byproduct of unrelated

activities like construction or maintenance. Consequently, certain districts may receive high-fidelity updates when scanned in detail, while other areas retain their previous, lower-resolution data.

Despite these variations, the City Digital Twin should adhere to the foundational principle, similar to the manufacturing industry, where all urban planning occurs within a unified, shared digital model. This integration is not merely about the technical characteristics of the model but also how urban services are coordinated. A City Digital Twin must be designed with human factors in mind, facilitating the orchestration of city ecosystems. This consideration ensures that changes to city plans or services can be digitally simulated to predict their impact on urban dynamics. Furthermore, such digital planning capabilities should be accessible to third parties to foster innovation and optimize service delivery.

To truly represent the pinnacle of urban digitization, a City Digital Twin must consist of four essential components as outlined by Lehtola et al. [10]:

1. The model must address the specific needs of the city.
2. It should support both high-fidelity content, like mature Building Information Modeling (BIM) data, and accommodate areas with lower fidelity, acknowledging local differences.
3. Continuous updating of the model is crucial, as cities are perpetually evolving. Autonomous updates from sensor systems — such as IoT networks, drones, and robotic vehicles — are essential, along with professional survey data.
4. The usability and safety of the system are paramount, enabling stakeholders to visualize, share, and interpret information effectively for improved decision-making. Adopting a human factors perspective is essential to unlock the full potential of City Digital Twin systems and enhance city management through informed, collaborative planning.

This integrated approach ensures that a City Digital Twin not only reflects the physical and operational aspects of the city but also serves as a robust tool for future urban development, policy analysis, and ecosystem orchestration.

Digital Representation and Information Linkages in City Digital Twins

City Digital Twins require that our urban environments are digitally recreated with essential features, such as buildings, infrastructure, terrain, vegetation, and other elements. Beyond replicating physical attributes, City Digital Twins must also enable the integration of data from diverse urban processes, an effort that involves understanding how information content is perceived. Cognition, however, differs between humans and computers, with Digital Twins relying on specific digital structures, or representations, to store information in a comprehensible format for machines.

Digital Twin 3D models are generally derived from raw data collected through point clouds or imagery. Key representations include [10]:

Point Clouds: The direct output from various sensor systems, these provide raw, dense data for urban modeling.

Voxels: Aggregated point data represented as cubes with a fixed spatial resolution, voxels facilitate physical simulations, such as visibility analysis, traffic noise mapping, solar radiation assessment, and wind flow studies. The trade-off with voxels lies in their high memory demands, often necessitating a balance between accuracy and scalability.

Mesh Surface Models: Created by triangulating point clouds, these models are valuable for topographic mapping and terrain analysis.

3D Models: Finalized models utilized within Geographic Information Systems (GIS) or BIM platforms, they serve both functional and visualization purposes in city management.

City Digital Twins build upon and extend the foundations of two well-established ecosystems: GIS and BIM, merging them to address the unique demands of urban environments. This dual heritage equips City Digital Twins with a more comprehensive and versatile set of tools and representations.

2.2.2 Geographic Information Systems

Geographic Information Systems are frameworks designed for gathering, managing, analyzing, and visualizing spatial and geographic data. They allow for the mapping

and examination of relationships, patterns, and trends through a combination of geographic features - like buildings, roads, or landscapes - and additional data layers that represent various attributes of those features. The key power of GIS lies in its ability to spatially organize data, enabling users to analyze the impact of physical features on different processes or environments. For example, GIS can help urban planners visualize infrastructure, assess environmental impacts, or manage emergency response strategies effectively.

Modern GIS has evolved significantly from its roots in paper-based cartography. Today's GIS systems leverage digital mapping techniques and powerful database management tools to provide a comprehensive view of urban environments. Vector data, which represent objects as points, lines, or polygons, forms the core of GIS analysis, and these elements can be enriched with additional attributes stored in relational databases. Spatial extensions to databases, such as PostGIS and Oracle Spatial, facilitate the efficient management of spatial data, while specialized platforms like 3DcityDB and DB4Geo handle complex 3D urban datasets.

Two primary standards dominate urban GIS applications: CityGML and CityJSON. CityGML, established by the Open Geospatial Consortium, provides a multi-level detail (LoD) framework, ranging from simple, flat building footprints (LoD 0) to highly detailed models that include interior structures (LoD 4). This hierarchical representation allows for various applications, from basic city planning to detailed indoor navigation. Additionally, CityJSON, a newer format proposed to the Open Geospatial Consortium in 2021, simplifies the exchange of urban data and is poised to enhance interoperability between platforms.

These standards are crucial for developing comprehensive city models that support digital twins. By structuring data in formats compatible with widely used GIS software like ArcGIS and QGIS, city planners and developers can integrate diverse datasets to create a dynamic digital replica of the urban environment. This integration is essential for monitoring city operations, conducting simulations, and optimizing infrastructure through predictive modeling.

2.2.3 Building Information Modeling

Building Information Modeling provides a detailed digital representation of physical structures, encompassing geometric, spatial, and infrastructural information. It has

revolutionized the Architecture, Engineering, and Construction (AEC) fields by offering a more intelligent modeling process compared to traditional 3D models. While 3D models mainly depict the physical aspects of structures, BIM goes a step further by integrating comprehensive metadata about these structures. This added dimension of information facilitates collaboration among professionals, optimizes project planning, and minimizes errors. One of the central benefits of BIM is its ability to improve cost management and streamline construction processes, enhancing efficiency across the project lifecycle.

The Industry Foundation Class (IFC) standard plays a pivotal role in BIM's ecosystem, ensuring data interoperability across various software platforms. By standardizing data formats, IFC enables seamless collaboration between stakeholders, though challenges remain due to the use of proprietary software in the industry. Open standards like OpenBIM are gaining traction to address this issue, promoting wider adaptability and integration.

Despite its advantages, BIM has limitations when applied to the concept of city digital twins. One significant shortcoming is that traditional BIM models are not equipped for real-time updates. Typically, these models are refreshed only at specific milestones, such as post-construction, which restricts their applicability in scenarios requiring dynamic monitoring and continuous data flow. While methods like Scan-to-BIM allow for periodic updates by incorporating as-is conditions, these updates are often infrequent and insufficient for real-time urban management.

In contrast, a digital twin represents a comprehensive, continuously updated virtual counterpart of a physical entity, facilitated by sensors and IoT technologies. This real-time interaction enables predictive simulations, allowing urban planners to forecast and optimize city operations. As such, while BIM serves as a foundational component for digital twins by providing detailed structural data, it requires further integration with real-time technologies to achieve the full potential of smart city management.

2.2.4 Combining BIM & GIS

The integration of GIS and BIM technologies is fundamental to realizing the potential of City Digital Twins. Combining these systems allows urban planners to enrich GIS-based city models with detailed architectural and infrastructural data from BIM.

For example, transforming BIM models into CityGML formats can significantly enhance the level of detail in 3D city models, facilitating simulations and analyses for urban management. However, this integration comes with significant challenges. Processing and managing vast data sets on a city scale require substantial computational resources, and BIM-GIS conversions often face technical obstacles, especially when dealing with large, complex urban environments. Moreover, achieving real-time updates across integrated models is essential but difficult. Automated data acquisition from IoT devices, drones, and other sensor systems becomes necessary to maintain the relevance and accuracy of city models. These updates enable predictive simulations, optimize city infrastructure performance, and support efficient decision-making. The ultimate goal is to create an extended, sensor-integrated city model that goes beyond static representation, enabling dynamic analysis and adaptive urban governance.

2.2.5 The Limitations of Achieving a True Digital Twin

Understanding the limitations of a true digital twin is crucial. As Batty [11] highlights, a digital twin is often envisioned as a real-time mirror of a physical process, matching the system's operations with high precision. However, any model that reflects another system remains, by necessity, an abstraction. Models simplify the complexity of real systems, discarding details that are not essential for the purpose of study or simulation.

This inherent simplification means that models can never replicate every detail of a real system. Instead, they emphasize select features, leading to an unavoidable difference between the model and the original system. Even when digital representations are composed similarly to the physical systems they depict, they remain distinct. The notion that a digital twin could fully mirror a system raises the question: if the twin were perfectly synchronized, how could it be used for simulations, planning, or learning about the system without being indistinguishable from it?

In urban applications, this limitation is apparent. Digital twins of cities derive from the representation of physical assets, with GIS and BIM technologies providing detailed but ultimately incomplete snapshots. These models, even when enhanced with real-time data like energy consumption or traffic flow, lack the integration of social and economic processes that define urban life. They remain, as Batty describes,

mere abstractions that simplify a city's complexity, capturing only a fraction of its dynamic nature.

As technology blurs the line between digital and physical realms, digital twins increasingly merge with their physical counterparts. Yet, the idea of a perfect digital twin — a complete, real-time replica — remains an idealization. The goal of refining digital models to better reflect reality is valuable, but we must acknowledge that a true digital twin, in every sense, may never be fully realized.

2.2.6 Complex Digital Twin examples

In January 2020, the Urban Planning and Development Institute "Giprogor Proekt" compiled a list of the top ten urban digital twins worldwide, featuring cities such as Singapore, Amaravati, Boston, Newcastle, Jaipur, Helsinki, Rotterdam, Stockholm, Rennes, and Antwerp [12].

Fig. 2.2 An example of a complex Digital Twin, showcasing Virtual Singapore, a detailed and dynamic 3D model used for urban planning and simulation. [13]

Virtual Singapore serves as a prominent example of a city digital twin, created to unify diverse 3D modeling initiatives within a collaborative platform accessible to public agencies, citizens, private sector organizations, and research institutions. This digital twin provides a detailed visualization of the city-state, supports real-time data gathering and analysis, and enables simulations as well as "what-if" scenarios that improve planning and decision-making. Nevertheless, it hasn't yet achieved the level

of a complete digital twin, as data flow remains unidirectional from the physical environment to the digital.

Another notable example, as summarized by Shahat et al. [14], is the development process behind Zurich's city digital twin. The process begins with data acquisition, which is structured around the spatial data from Zurich's existing 3D city model. This model includes three main components: a terrain model, a block model, and a roof model. The data sources for these models are diverse: LiDAR imagery supports the terrain model, floor plans from the city's cadastral survey inform the block model, and semi-automated photogrammetry aids in creating the roof model. Each component is built with varying levels of detail to suit different visualization needs. Zurich's digital twin enables a detailed view of street spaces, underground infrastructure, and select public buildings with enhanced levels of detail.

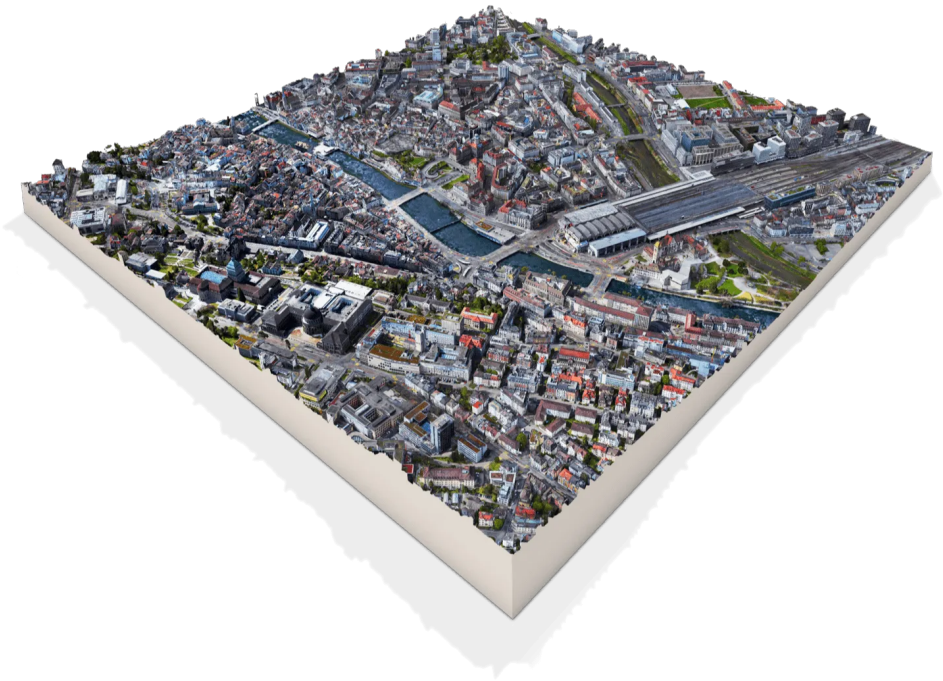

Fig. 2.3 An example of Zurich's city Digital Twin, a sophisticated 3D model used for urban analysis and real-time simulation of city dynamics. [15]

To improve accessibility and interoperability, spatial data and metadata standards were formally established under a federal act, which also underpins Zurich's digital twin governance framework. Open government data is leveraged to facilitate contributions from various stakeholders, while a geoportal aggregates and updates geodata

automatically. This portal also includes a visualization tool that allows users to view 3D city components and ongoing construction projects. The Zurich digital twin has demonstrated several applications in urban planning, including scenario comparison for urban development, integration of urban climate considerations into planning, and facilitation of public participation.

To enhance Zurich's digital twin further, increasing the detail levels of buildings and city elements, integrating BIM and GIS applications, and reducing data update intervals and processing times for 3D models have been identified as priorities.

2.2.7 Research Agenda

Shahat et al. [14] undertook a comprehensive review aimed at assessing the landscape of city-scale digital twin research and formulating a research agenda to propel advancements in this domain. Their analysis involved a thorough search across major academic databases, including Scopus, Web of Science, and Google Scholar, using a broad approach to encompass relevant studies not only on individual buildings but also on larger urban and district scales. This method resulted in a collection of 171 articles as of October 2, 2020.

Following the removal of duplicates and a careful relevance assessment, a final selection of 42 articles was made, which included both theoretical and empirical research while excluding non-peer-reviewed publications and works not in English. Notably, some studies focused on large infrastructure projects were retained for their potential insights into city-scale digital twins. Through this review and thematic classification, significant gaps and challenges in achieving a fully realized city digital twin were identified.

For example, although there have been multiple proposals for creating integrated data models, the existing literature does not yet present a practical application that fully utilizes standardized data schemas to encompass all urban functions and processes. The visualization of the city's intricate physical details also poses a challenge, given the vast amount of data and the computational resources required to manage such extensive digital models.

Consequently, the formulated research agenda outlines three essential elements designed to assist city digital twin researchers, city managers, and developers in their pursuit of developing a comprehensive and accurately mirrored city digital twin.

Optimizing Data Collection and Processing Methods

The enhancement of data management within the city digital twin is an area that warrants continuous research. As a fundamental component in developing a successful city digital twin, the quality and precision of data are crucial for establishing an effective digital platform. While significant strides have been made in areas such as data transfer, storage, and processing, several challenges still prevent the complete realization of the city digital twin's potential as an integrated platform. A key issue is the standardization of data; the diverse and often incompatible data generated from different urban domains underscores the necessity for cohesive data standards that can facilitate the integration of various software applications used in city digital twins.

Some research efforts have suggested the adoption of standardized data models or open standards, but these practices have not yet achieved widespread acceptance. The multitude of methodologies and software in use across different city digital twin projects, coupled with the rapid advancement of technology and the variety of data produced, makes it challenging to establish a one-size-fits-all approach.

Additionally, improving the Level of Detail within city models is vital for producing a more thorough and accurate representation. However, increasing the LOD can result in larger model sizes, which in turn complicates processing capabilities. Consequently, there is a need to find a balance between enhancing the LOD for better visual representation of the city digital twin and keeping the model size manageable to ensure efficient data handling, essential for conducting analyses, making predictions, and implementing various applications.

Enhancing the Integration of Socio-Economic Dimensions

The city, as a complex SoS, necessitates an intricate model that accurately reflects its various systems and domains. One of the most significant challenges in visualizing and operationalizing this complex urban environment lies in the integration of non-physical systems, including socio-economic processes and activities. Although some research has explored aspects of incorporating these non-physical elements into city digital twin models — such as behavior modeling based on the needs and mental attributes of agents — much potential remains untapped. The integration of socio-economic components can significantly enhance various urban domains, including urban planning, transportation, and environmental management.

By improving the modeling of socio-economic behaviors and activities, while also accounting for contextual factors like political frameworks, governance structures, and cultural dynamics, we can gain deeper insights into urban dynamics. This approach will facilitate more comprehensive analyses and simulations of future scenarios, ultimately leading to improved planning and decision-making. However, challenges extend beyond data collection and modeling; they also encompass issues related to privacy and data security. The ability of authorities to access personal information raises serious concerns about public privacy, necessitating transparent and constructive dialogue with citizens. Moreover, risks associated with data breaches by third parties or hackers present significant threats. Therefore, while it is crucial to explore how to effectively model these non-physical components, understanding their potential impacts on the digital twin model is equally important.

Fostering Synergistic Integration of Digital and Physical Entities

Achieving a fully mirrored city digital twin necessitates a strong mutual integration with its physical counterpart, a concept that has not been extensively addressed in the existing literature. The capability to update the digital twin with real-time or near-real-time information from the physical environment is well established. However, facilitating the reverse data transfer, which would allow control over the city's physical aspects, presents significant challenges. In manufacturing contexts, Kritzinger et al. [16] differentiated between a digital model, **digital shadow**, and digital twin. In a digital model, data transfer is manual, while in a digital shadow, data flows automatically from the physical entity to the digital one. Conversely, a true digital twin allows for seamless bi-directional data flow. Currently, no existing city digital twin has achieved this level of interaction, leading to the characterization of many as merely digital shadows rather than fully functional digital twins.

To develop a genuine mutual integration between the digital twin and its physical counterpart, the model must possess the capability to issue control commands to the physical environment. This integration can enhance urban management by improving efficiency and service delivery. Implementing artificial intelligence and actuators for feedback to the physical environment has been proposed as a strategy for achieving this control. While this approach has shown promise in areas like energy management and construction, research is still needed for broader urban applications. Additionally, leveraging advancements in robotics and automation is critical for realizing the full integration of the city digital twin and enhancing its

operational control capabilities. Nonetheless, it is essential to address the potential risks associated with increased automation in urban management, such as social inequality and regulatory issues.

2.2.8 Conclusions on Digital Shadows and City Building

In this section, we explored the concepts of Smart Cities and City Digital Twins, highlighting their importance in urban planning. While the idea of a true Digital Twin remains elusive due to the inherent simplifications of digital models, the concept of Digital Shadows offers a more achievable solution. Unlike Digital Twins, which require continuous data exchange, Digital Shadows rely on static digital representations of physical systems, making them more practical for urban monitoring and analysis.

Examples like Zurich and Virtual Singapore showcase the potential of City Digital Twins, but the challenge lies in the lack of clear guidelines for evaluating their effectiveness. The integration of BIM and GIS is essential to bridge the gap between digital models and real-world dynamics. Ultimately, while the full realization of a Digital Twin for a city is still distant, Digital Shadows present a viable alternative for enhancing urban decision-making and planning.

2.3 Driving Simulations

Building upon the concept of City Digital Twins and Digital Shadows, driving simulation plays a crucial role in translating digital urban environments into interactive and testable scenarios. The ability to replicate real-world road networks and traffic conditions within a simulated environment enhances urban planning, transportation analysis, and vehicle development.

Driving simulation is a powerful tool for studying human driving behavior, developing advanced driver assistance systems (ADAS), and evaluating complex scenarios in a controlled and safe environment. The ability to recreate driving conditions realistically allows researchers and engineers to analyze responses to various situations without the risks associated with real-world testing. This aspect is particularly crucial in scenarios that involve hazardous conditions, emergency maneuvers, or distracted driving studies, where real-life experiments would be impractical or unsafe.

One of the main advantages of driving simulation lies in its ability to provide a highly controlled and repeatable environment. Unlike real-world testing, where conditions can vary unpredictably due to traffic, weather, and human factors, simulations allow for precise manipulation of variables. This capability makes it an essential tool for studying driver decision-making processes under different circumstances, such as drowsy driving, reaction times to sudden obstacles, or the cognitive load induced by in-vehicle distractions. Furthermore, the cost-effectiveness of simulations compared to real-world testing makes them an attractive solution for both research institutions and the automotive industry, significantly reducing development times while ensuring driver safety [17].

The technological advancements in driving simulation have led to highly immersive systems that integrate advanced vehicle physics modeling with interactive interfaces, including virtual reality environments, motion platforms, and haptic feedback. These innovations enhance the realism of simulations, ensuring that test subjects experience a highly representative driving environment. For example, modern simulators can replicate road textures, vehicle dynamics, and even the forces acting on the driver, making them an effective alternative for training and system validation. These aspects are particularly relevant for autonomous vehicle development, where vast amounts of driving data are needed to train machine learning models. By

using simulations, developers can expose autonomous systems to a wide range of driving conditions, including rare but critical edge cases that would be difficult or dangerous to reproduce on real roads.

Another crucial application of driving simulation is the development and testing of ADAS. These systems, designed to improve vehicle safety and driving comfort, require extensive validation before deployment. Simulators provide a risk-free environment where engineers can test features such as automatic emergency braking, lane-keeping assistance, and pedestrian detection in diverse and repeatable conditions. Additionally, simulations allow for fine-tuning of these systems based on human-in-the-loop studies, where real drivers interact with the technology to assess usability and effectiveness before real-world implementation.

In terms of driver education and training, simulators are widely used to enhance driver preparedness for challenging situations. Training programs can expose drivers to scenarios such as icy roads, sudden mechanical failures, or emergency braking conditions, helping them develop better situational awareness and response strategies. This application is particularly beneficial for professional drivers, including those in law enforcement, emergency response, and heavy transport, who need to be well-equipped to handle high-risk scenarios.

2.4 The CARLA Simulator and Unreal Engine

CARLA [18] is an **open-source** driving simulator that has played a pivotal role in the advancement of autonomous vehicle research, particularly within urban environments. Initially developed using **Unreal Engine 4** (UE4), CARLA was designed to provide high-fidelity simulations for the testing and validation of autonomous driving systems. The simulator's architecture takes full advantage of UE4's advanced rendering capabilities, allowing for realistic environments and dynamic simulations that mimic real-world scenarios, such as the presence of traffic, pedestrians, and varying weather conditions. Additionally, CARLA utilizes a **server-client system**, where the server runs the simulation and renders the environment, while the client, connected via Python, controls the vehicle and gathers sensor data.

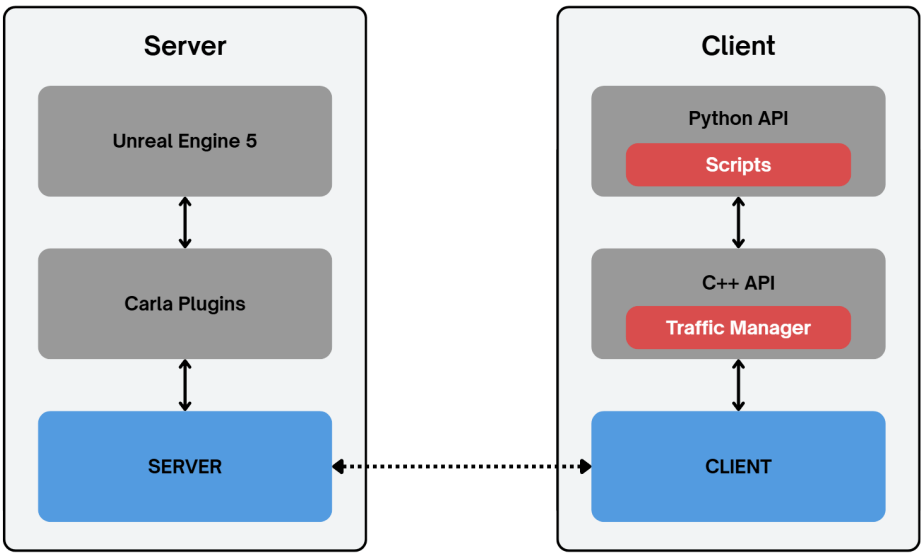

Fig. 2.4 The Client-Server architecture of CARLA separates simulation logic from execution, with the client handling control via the Python and C++ APIs, while the server runs the environment in Unreal Engine 5 with CARLA Plugins.

Recently, CARLA has transitioned to Unreal Engine 5 (UE5), which brings a wealth of new capabilities that significantly enhance the realism and flexibility of the simulator. UE5 introduces several cutting-edge technologies that broaden CARLA's potential for urban driving simulation. One of the key advancements is **Nanite**, which allows for highly detailed and complex environments without sacrificing performance. Nanite's virtualized geometry system enables the rendering of billions of polygons in real time, providing a level of detail that was previously unattainable in large-scale open-world simulations. This is particularly important when simulating urban environments, where the complexity of the architecture and infrastructure must be accurately captured.

Another major improvement brought by UE5 is **Lumen**, a fully dynamic global illumination system that allows for more realistic lighting and shadows. This technology ensures that the lighting in the simulated environment reacts to changes in time of day and weather conditions in a natural and consistent manner. Lumen's advanced capabilities enhance the visual fidelity of simulations, ensuring that autonomous driving models are tested in environments that closely resemble real-world scenarios.

In addition to these graphical advancements, UE5 provides improved physics and material handling, which significantly enhance the realism of vehicle interactions

with road surfaces, traffic, and pedestrians. CARLA's NPC behavior is another key feature contributing to the simulator's realism. Non-player vehicles (NPVs) are based on the standard UE vehicle model, but with adjusted kinematic parameters to improve their realism. These vehicles follow lanes, respect traffic lights and speed limits, and make decisions at intersections. The presence of pedestrians further enriches the environment, with their movements randomized but guided by navigation maps that allow them to avoid collisions with vehicles. Pedestrians are also equipped with accessories like smartphones or shopping bags, adding a layer of detail that heightens the simulation's immersive quality [18].

Fig. 2.5 CARLA's simulation environment supports dynamic weather conditions, influencing both the scene appearance and NPC behaviors, such as using umbrellas in rainy scenarios. [18]

Furthermore, CARLA offers exceptional flexibility in configuring the sensor suite of the autonomous agent. The simulator supports a wide range of sensors, including RGB cameras and pseudo-sensors that provide ground-truth depth and semantic segmentation. These sensors can be configured for specific positions, orientations, and parameters, such as field of view and depth of field, allowing for a high degree of customization. Notably, CARLA also includes a semantic segmentation pseudo-sensor that categorizes the environment into 12 different classes, such as road, lane markings, traffic signs, and pedestrians. This suite of sensors is crucial for evaluating

the performance of autonomous driving policies, providing accurate data for testing and training in highly detailed and varied conditions.

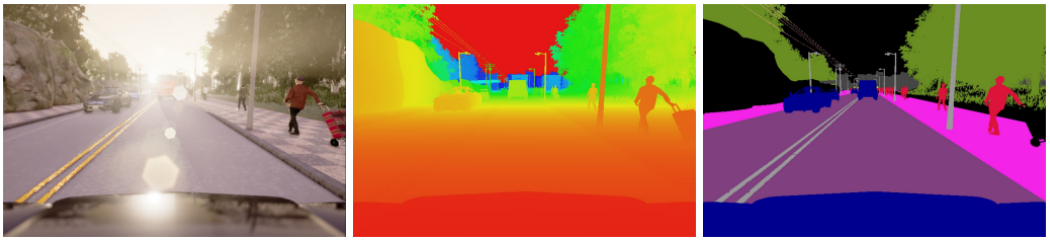

Fig. 2.6 Different sensor modalities in CARLA provide diverse environmental perceptions, including standard RGB vision, depth estimation, and semantic segmentation for scene understanding. [18]

CARLA goes further by tracking the vehicle's state in real time, providing data such as GPS-like coordinates, speed, acceleration, and **collision impacts**. It also tracks compliance with traffic regulations, such as lane adherence and speed limits, ensuring that the vehicle responds appropriately to dynamic scenarios. These metrics are vital for assessing the performance of autonomous driving models and ensuring that they are capable of navigating urban environments safely and effectively.

The transition to Unreal Engine 5 significantly expands CARLA's capabilities, offering not only improved graphical fidelity but also enhanced physics, NPC behavior, and sensor configurability. These advancements enable the simulation of even more complex and realistic urban environments, providing researchers and developers with a powerful platform for testing autonomous driving systems in a highly customizable, controlled virtual environment. With the improvements brought by UE5, CARLA continues to be at the forefront of autonomous driving research, supporting the development, training, and evaluation of next-generation autonomous vehicle technologies.

2.5 Virtual Reality in Driving Simulations

Virtual reality technology has revolutionized driving simulators by enhancing immersion, spatial awareness, and overall user experience. Traditional simulators, which rely on large panoramic screens and high-resolution projectors, often lack the depth and realism necessary to fully replicate real-world driving conditions. Additionally, some simulators incorporate motion platforms to simulate accelerations, such as centrifugal forces and braking effects, but these setups require complex and expensive hydraulic systems [19].

The development of smaller and portable simulators for training purposes has led to the substitution of large projection screens with single flat screens or multiple monitors, along with simplified motion or vibration systems. VR devices present a natural solution for compact and cost-effective driving simulation while maintaining a high degree of immersion. Head-mounted displays (HMDs) eliminate the limitation of viewing angles, allowing drivers to turn their heads naturally while maintaining a continuous virtual experience. Furthermore, VR enhances the realism of driving simulation by providing high-quality stereoscopic depth perception and spatialized 3D sound.

A key advantage of VR-driven simulators is their ability to create an immersive environment where users perceive depth and motion more naturally. The stereoscopic rendering provided by HMDs enhances depth perception, while head-tracking mechanisms ensure that changes in perspective align with the user's movements. This level of realism positively influences driver behavior, as participants respond more naturally to virtual scenarios, leading to more accurate behavioral studies and training exercises [20].

However, VR-based driving simulation is not without challenges. Motion sickness, or simulator sickness [21], is a common issue, primarily caused by discrepancies between visual stimuli and physical motion. The latency in head tracking, frame rate inconsistencies, and the limited field of view of HMDs can exacerbate discomfort and impact user performance. Addressing these issues requires optimizing rendering techniques, reducing latency, and refining motion synchronization between the virtual and physical environments.

Another technical limitation involves computational constraints. High-fidelity VR simulations require significant processing power to maintain high frame rates and ensure real-time responsiveness. Achieving an optimal balance between graphical detail and performance remains a challenge, particularly when simulating complex driving scenarios that involve dynamic objects, changing weather conditions, and detailed urban environments.

Bayarri et al. [19] propose a multi-layered structure for organizing driving simulation environments. This framework consists of three hierarchical levels: the topological layer, the visual representation layer, and the motion description layer. The topological layer defines the abstract road network, decomposing the urban environment into links and junctions. The visual representation layer maps these elements to their physical counterparts, such as buildings, sidewalks, and traffic infrastructure. The motion description layer, though not directly visible, is essential for modeling vehicle dynamics by defining lanes and traffic rules. This structure enables efficient database management and optimizes rendering and simulation performance.

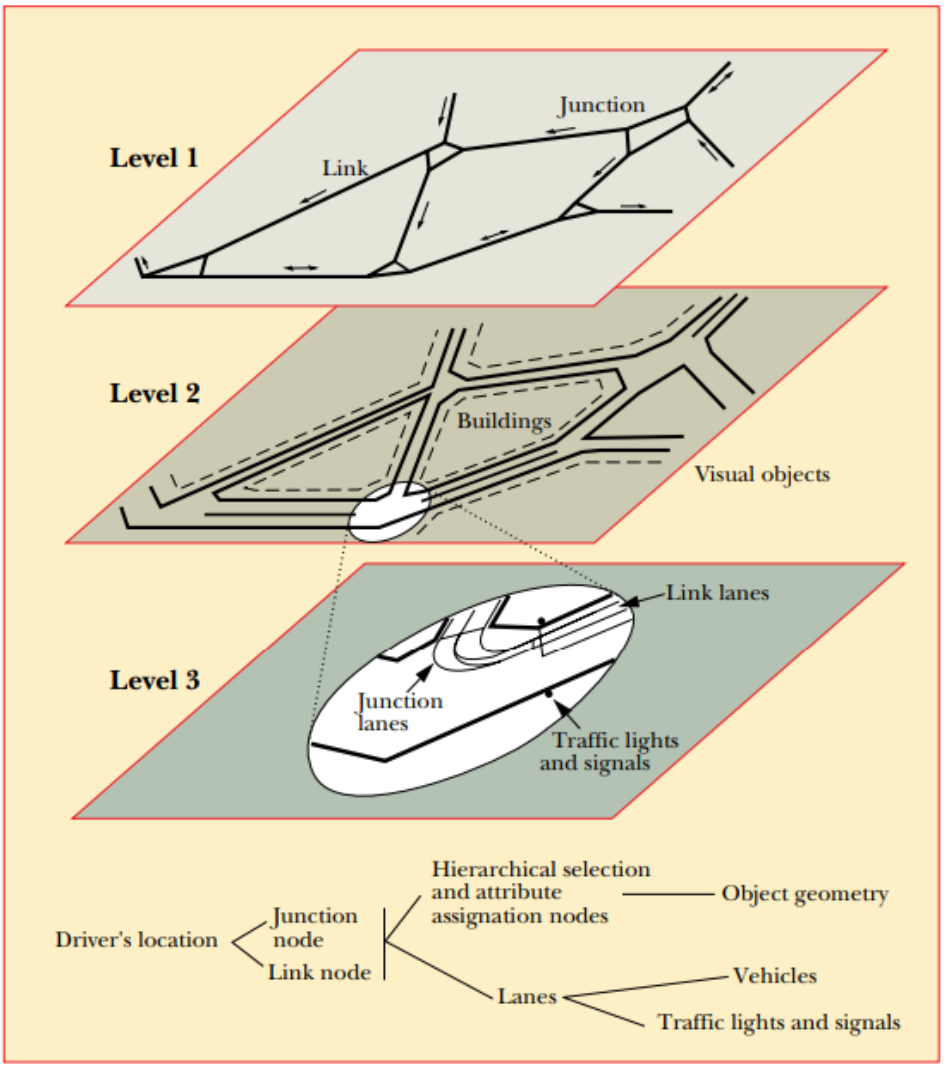

Fig. 2.7 Bayarri et al. hierarchical organization of driving simulation data, structured into three layers: the road network (links and junctions), the visual entities (buildings, sidewalks, traffic elements), and the motion-related structures (lanes). This layered approach optimizes database navigation, spatial organization, and real-time rendering efficiency. [19]

In addition to visual immersion, realistic driving simulation requires an accurate representation of vehicle dynamics. The behavior of a car in a simulation is influenced by parameters such as mass, engine power, tire friction, and suspension. Vehicles are typically represented as rigid bodies with multiple wheel colliders, and the configuration of the suspension model determines the applied forces. A detailed cockpit model, accurate traffic signals, and other environmental factors further contribute to the realism of a virtual driving experience.

Despite these challenges, VR-based driving simulators present clear advantages over traditional systems. Their ability to replicate real-world conditions in a controlled environment makes them highly useful for testing driver reactions, evaluating vehicle safety features, and training individuals in hazardous driving situations without real-world risks. Training software in traditional setups, such as single-screen environments, suffers from limited immersion, as a standard monitor does not provide an adequate field of view, nor does it replicate intuitive head movement and acceleration feedback.

As technology advances, improvements in VR rendering, real-time physics simulation, and haptic feedback integration are expected to further enhance the effectiveness of these simulators. Learning how to drive requires extensive training and cognitive effort, and VR-based training has demonstrated benefits in adapting to individual learners' needs. Studies have shown that VR simulations have been successfully used for security training and infrastructure analysis, demonstrating their potential in various fields beyond driving simulation.

In conclusion, virtual reality is reshaping the landscape of driving simulation by providing a more immersive and interactive experience. While challenges related to motion sickness and computational performance persist, ongoing developments in VR hardware and software promise to make VR-driven driving simulators an indispensable tool for research, training, and vehicle development.

Chapter 3

Designing the Pipeline

Designing a pipeline for generating realistic and structured environments in CARLA requires a careful evaluation of available tools and methodologies. Initially, CARLA's approach to map creation relied on manually designed assets and procedural road generation, which, while effective, posed limitations in terms of scalability and fidelity to real-world environments. To overcome these challenges, an analysis was conducted to identify external tools that could enhance the accuracy and efficiency of the process, enabling the integration of geographic and 3D data into CARLA.

This chapter examines the methods originally used for map creation in CARLA and explores various tools that allow for the utilization of real-world data to generate playable environments. Specifically, it discusses tools such as Cesium and BlenderGIS, analyzing their role in extracting, processing, and converting geographical data into a structured format suitable for CARLA. The objective is to define a workflow that improves the integration of real-world elements into the simulator while maintaining flexibility and accuracy in the generated maps.

3.1 CARLA's Original Map Creation Method

CARLA proposes the **Digital Twin Tool** as a solution for the procedural generation of 3D environments based on road networks extracted from OpenStreetMap (OSM). Through its interface within CARLA's Unreal Engine editor, users can select a specific map region, download the corresponding road network, and generate a CARLA-compatible environment. The tool procedurally fills the areas between roads with automatically generated buildings, adjusting their shape and dimensions to fit the layout. This approach allows for the rapid creation of diverse and complex urban scenes with minimal manual effort.

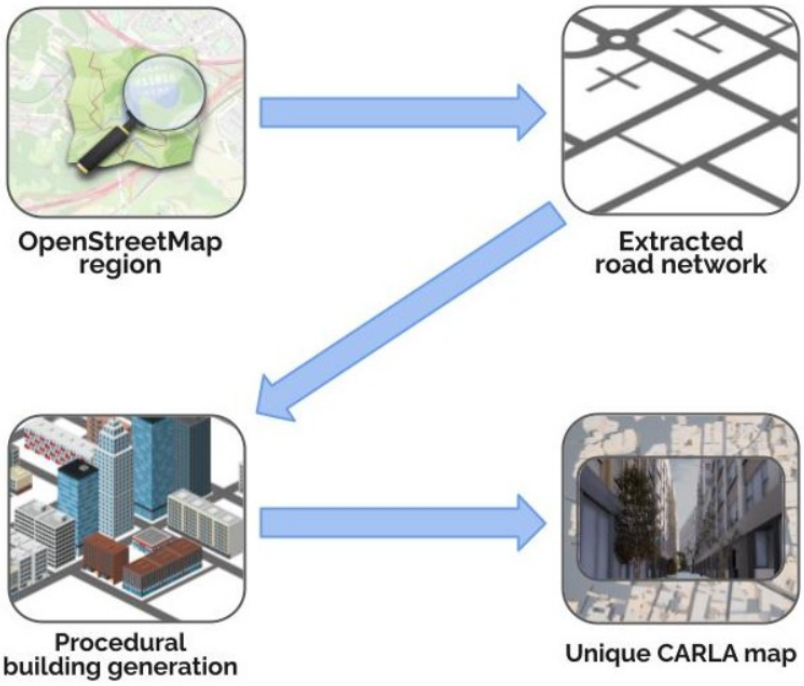

Fig. 3.1 Workflow of the Digital Twin Tool, from selecting an OpenStreetMap region to generating a procedural CARLA map. [22]

The generation process begins with the extraction of road networks from OSM data, forming the foundation of the map. The tool then applies realistic road textures, markings, and surface details. Next, the empty spaces between the roads are populated with buildings whose footprints and height information are derived from OSM. These virtual buildings are further enhanced with procedurally generated facades, including windows, doors, and balconies. The architectural style varies depending

on the building dimensions: taller structures are typically assigned an office-like appearance, while smaller ones adopt residential or commercial designs. Additional vegetation elements are placed along sidewalks to enhance the scene's realism.

While this method is highly efficient for quickly generating large environments, its procedural nature makes it unsuitable for applications requiring high accuracy and real-world fidelity. Since the buildings and urban elements are not direct digital replicas but rather approximations generated from limited OSM attributes, this approach lacks the precision necessary for Digital Twin applications.

As an alternative, the CARLA documentation also suggests using **RoadRunner** or TrueVision Designer for road network generation and OpenDRIVE (.xodr) export. These tools provide greater control over road layout and structure, making them more suitable for detailed and accurate environment reconstruction. Once the road network is generated, the map can be further refined using CARLA's built-in assets or Unreal Engine's Foliage tool for additional environmental details.

These methods were used in CARLA for Unreal Engine 4, but they are no longer available in the current versions of the simulator. Furthermore, they do not represent viable solutions for the objective of creating a City Digital Shadow, as they rely on procedural generation rather than accurately replicating real-world urban environments.

3.2 BlenderGIS

BlenderGIS is an add-on for Blender that facilitates the integration of geospatial data into 3D modeling workflows. It allows users to import and manipulate geographic datasets, including terrain elevation models (DEM), OpenStreetMap (OSM) data, and satellite imagery, enabling the reconstruction of real-world environments within Blender. Through its tools, BlenderGIS supports the import of road networks, building footprints, and elevation data, offering a way to generate large-scale urban environments with georeferenced accuracy.

In the context of this work, BlenderGIS was considered as a potential tool for processing and converting geographic data into a format suitable for integration within CARLA. The ability to import OSM data directly into Blender provides a structured way to generate city layouts, while the support for DEM data allows for

the recreation of realistic terrain. However, despite its capabilities, BlenderGIS has significant limitations that make it unsuitable for accurately constructing Digital Shadows.

One of the primary drawbacks is the way buildings are generated. The add-on creates basic block structures that do not accurately reflect real-world architectural details. This lack of precision can lead to misalignment and intersection issues when combining the generated buildings with OSM-based road data. Additionally, the procedural nature of its building generation results in structures that do not always correspond to their real-world counterparts, making it difficult to achieve a level of detail and fidelity necessary for realistic simulations.

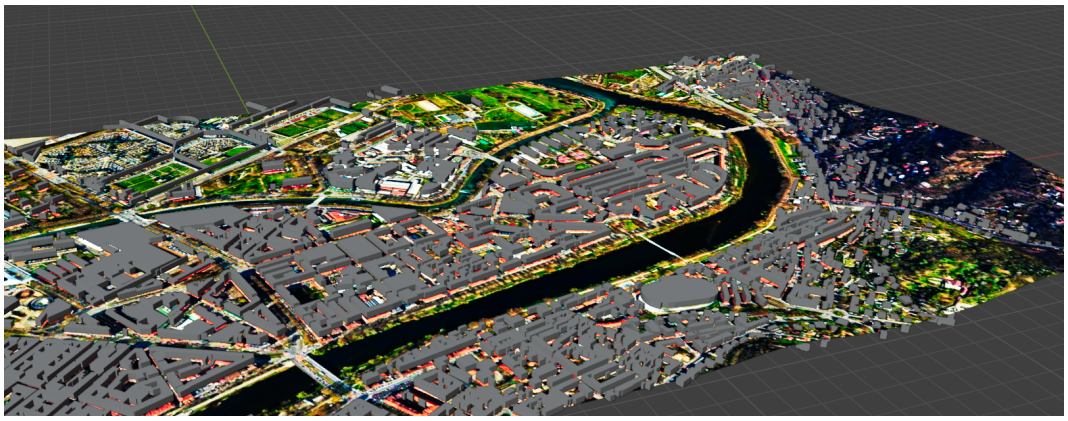

Fig. 3.2 A 3D city model created using BlenderGIS, showing imported OpenStreetMap data with basic building footprints.

Due to these limitations, BlenderGIS does not provide the accuracy required for this project. While it can serve as a preliminary tool for visualizing geospatial data, its simplified approach to urban modeling makes it unsuitable for generating structured environments that align precisely with the requirements of a City Digital Shadow.

3.3 VectorZero (RoadRunner)

RoadRunner supports a variety of Geographic Information System formats, allowing users to import external GIS data as references for road construction. This approach

aims to improve the accuracy of generated maps by integrating real-world elevation, imagery, and point cloud data into the scene.

The proposed workflow involves downloading aerial imagery, elevation data, and point clouds from the U.S. Geological Survey (USGS) National Map Explorer Interface. These datasets serve distinct purposes: aerial imagery provides a visual reference for road alignment and surface textures, elevation data defines terrain height variations, and point cloud data offers detailed object positioning for elements like trees, buildings, and road markings. Once imported, RoadRunner's tools allow for further customization of the scene, refining road geometry and urban layouts. Since these tools are fundamental to the map creation process, a more detailed explanation of their functionalities will be provided in Chapter 5.

However, the USGS National Map Explorer Interface no longer allows direct downloads of these datasets, and even when available, its coverage is limited to U.S. territory, making it unsuitable for applications outside the United States. This limitation required an evaluation of alternative data sources to follow the workflow for locations outside the US, particularly for Italian regions. Several alternative platforms and tools were identified to acquire equivalent datasets:

Copernicus Sentinel: The Copernicus Programme, operated by the European Space Agency (ESA), provides free Earth observation data. The Sentinel satellites, particularly Sentinel-2, offer high-resolution optical imagery, making them a valuable alternative to USGS aerial imagery. Sentinel data can be accessed through Copernicus Open Access Hub and processed using GIS software. However, Sentinel-2 imagery may lack the same level of detail as dedicated aerial photography.

OpenTopography: A platform that provides high-resolution elevation and LiDAR point cloud data from various global sources. Although highly accurate, data availability varies significantly by region, with detailed datasets often restricted to specific research projects or governmental releases.

TINitaly: A high-resolution Digital Elevation Model (DEM) dataset specifically for Italy, offering a 10-meter resolution terrain model based on national surveys. This dataset serves as an effective alternative for elevation data in the Italian territory.

QGIS: Several plugins within QGIS, an open-source GIS software, allow users to retrieve and process multiple geographic datasets:

- QuickOSM: Extracts OpenStreetMap (OSM) data, including road networks and building footprints.
- SRTM: Provides Shuttle Radar Topography Mission (SRTM) elevation data, useful for terrain modeling.
- LAStools: Processes LiDAR point cloud datasets, offering the ability to extract detailed 3D environmental features.

Fig. 3.3 Visualization of satellite imagery and OpenStreetMap road data of Mappano within QGIS.

Despite these alternatives, finding high-resolution datasets for all three required elements — imagery, elevation, and point clouds — remains a challenge, especially for regions outside the United States. In Italy, for instance, high-quality LiDAR datasets are often restricted or available only for specific areas, limiting their applicability for large-scale digital twin generation. This difficulty in acquiring complete datasets increases the complexity of the workflow, requiring additional processing and data integration steps to achieve results comparable to those originally intended by the RoadRunner workflow.

3.4 Cesium for Unreal

Cesium is a platform designed for 3D geospatial visualization, enabling the rendering and interaction with large-scale geographic datasets. It is widely used in applications that require **real-time visualization of terrain**, **satellite imagery**, and **3D city models**, such as simulations, urban planning, and geospatial analysis. The core of Cesium's technology is based on a tile-based streaming system, which allows for the efficient visualization of vast datasets without requiring extensive local storage.

To integrate Cesium's capabilities within Unreal Engine, the Cesium for Unreal plugin was developed. This plugin enables users to import 3D tilesets using **API requests** to Cesium's data services, which include terrain models, satellite imagery, and photogrammetric city reconstructions. However, a key limitation of this approach is that Cesium's API access is paid, meaning that continuous use of the system requires an active subscription. This dependency on commercial data services makes it a less flexible solution for long-term or large-scale simulations.

Beyond the API restrictions, another major drawback of Cesium is its limited mesh resolution, which makes it unsuitable for a City Digital Shadow in CARLA. Driving within a simulated environment requires precise road geometry to ensure a realistic and immersive experience. The terrain and road surfaces provided by Cesium often lack the necessary detail, resulting in uneven surfaces and imprecise alignments that make driving simulations unrealistic. Given these constraints, Cesium was not adopted as a primary method for generating environments in CARLA.

Despite these limitations, a test was conducted using Cesium as a background layer for a map set in Val Susa. In this test, Cesium provided the underlying terrain and imagery, while the roads were exported from RoadRunner using OpenStreetMap (OSM) data. This hybrid approach allowed for a more structured driving experience by maintaining precise road geometry while benefiting from Cesium's large-scale terrain visualization.

To enhance environmental realism, Unreal Engine's foliage tool was also used in this test, in combination with procedural tree generation techniques. The reason for this approach was that foliage in Unreal Engine can only be applied to Landscape actors, while Cesium's map is not recognized as a Landscape. This limitation was

overcome by procedurally placing trees, ensuring that the scene contained natural elements without requiring direct integration with the Cesium terrain.

Fig. 3.4 Test scenario featuring a road generated in RoadRunner, integrated into an environment provided by Cesium for Unreal.

The results of this test highlighted several key insights. As mentioned previously, the API cost limitations restricted the use of Cesium to the trial period provided by Google Cloud, making it impractical as a long-term solution. Additionally, the resolution constraints of Cesium tilesets necessitated a hybrid approach, rather than relying solely on Cesium for road generation. However, Cesium for Unreal does offer some notable advantages: it features an efficient Level of Detail (LOD) system, intuitive performance parameters that allow users to easily control computational load, and a straightforward way to navigate large-scale environments by simply adjusting latitude and longitude. These aspects make it useful for exploratory applications, but ultimately unsuitable for high-fidelity driving simulations such as those required in CARLA.

Overall, the evaluation of different methods for generating Digital Shadows highlighted significant limitations in terms of detail and accuracy. None of the approaches analyzed provided a sufficient level of fidelity to be directly applied to high-quality driving simulations. The observed constraints—whether related to resolution, adaptability, or long-term feasibility—demonstrate the need for a more effective strategy. The next chapter introduces the selected approach, detailing the reasoning behind its choice and its implementation.

Chapter 4

Point Clouds for Digital Shadow

LiDAR (Light Detection and Ranging) is a remote sensing technology that emits laser pulses to measure distances and create high-precision 3D point clouds. A point cloud is a dense collection of spatial data points that represent the geometry of an environment, capturing surfaces, buildings, and other structures with remarkable detail. This technique is widely used for urban modeling, terrain reconstruction, and autonomous driving simulations due to its ability to generate accurate digital representations of real-world environments.

According to Wang et al. [23], LiDAR point clouds offer significant advantages over traditional photogrammetry methods, particularly in geometric accuracy and structural completeness, making them an essential tool for 3D urban modeling. LiDAR data can be processed to extract road networks, classify urban features, and generate structured models suitable for simulation environments.

Given these capabilities, LiDAR-based point clouds were explored as a method for generating a testable version of a City Digital Shadow within CARLA. Unlike procedural generation or low-resolution GIS-based workflows, LiDAR data provides an accurate foundation for simulation environments, improving realism and ensuring a more precise representation of urban spaces. This chapter details the process of utilizing the point cloud to generate a Digital Shadow.

4.1 SyntCity

SynthCity [24] is a large-scale synthetic point cloud dataset designed to support research in automatic point cloud classification and segmentation. The dataset consists of 367.9 million points and represents an urban/suburban environment, generated using a simulated Mobile Laser Scanner (MLS) within Blender using the Blensor plugin.

AREA	BUILDING	CAR	NATURAL GROUND	GROUND	POLE LIKE	ROAD	STREET FURNITURE	TREE	PAVEMENT	TOTAL
1	14,355K	591K	372K	503K	239K	26,410K	198K	1,034K	2,532K	**46.23M**
2	12,241K	1,123K	526K	832K	248K	32,818K	97K	1,707K	2,981K	**52.57M**
3	16,222K	915K	186K	712K	206K	26,521K	227K	866K	2,546K	**48.40M**
4	14,767K	510K	563K	622K	175K	25,111K	415K	1,673K	2,800K	**46.64M**
5	11,424K	909K	897K	593K	256K	29,239K	92K	2,358K	2,982K	**48.75M**
6	16,521K	709K	918K	1,034K	174K	26,782K	365K	2,234K	3,184K	**51,92M**
7	1,895K	226K	330K	272K	83K	11,178K	22K	568K	938K	**15.51M**
8	4,833K	544K	607K	984K	171K	25,914K	36K	962K	2,702K	**36.75M**
9	5,716K	381K	390K	656K	84K	11,897K	19K	687K	1,321K	**21.15M**
Total	**97.97M**	**4.78M**	**12.09M**	**6.21M**	**21.99M**	**215.87M**	**1.47M**	**5.91M**	**1.64M**	**367.95M**

Fig. 4.1 Visualization of SynthCity point cloud data, categorized by area and object type. [24]

One of the key motivations behind SynthCity is the lack of large-scale, high-quality annotated point cloud datasets for training deep learning models. While datasets such as Paris-Lille-3D and Semantic3D exist, they often suffer from class imbalances, static scanning methods, or insufficient coverage for deep learning applications. In contrast, SynthCity was designed to simulate real-world MLS data while providing complete and noise-free ground-truth annotations for nine object categories: roads, sidewalks, ground surfaces, vegetation, buildings, poles, street furniture, and vehicles.

In the context of this work, SynthCity was used for preliminary testing in its reduced version, as processing the full dataset requires substantial computational resources. However, some significant challenges were encountered when working with this dataset. First, SynthCity is stored in the .parquet format, a highly efficient but less commonly supported format in traditional point cloud processing tools. Converting it to more widely used formats such as .xyz or .ply proved to be non-trivial, requiring additional data preprocessing steps.

Listing 4.1 Script for converting SynthCity point cloud data from Parquet to XYZ format.

```
import pandas as pd

def converter(file_path, filename):
    # Load the Parquet file
    df = pd.read_parquet(file_path)

    # Select the required columns (with or without color)
    columns = ['X', 'Y', 'Z'] + (['R', 'G', 'B'] if {'R', 'G',
        'B'}.issubset(df.columns) else [])

    # Save directly in .xyz format without header and index
    df[columns].to_csv(f"{filename}.xyz", sep=' ', index=False,
        header=False)
```

Another issue observed was the presence of shadowed areas in the point cloud regions where points were missing due to occlusions caused by other objects. This is a common artifact in LiDAR-based datasets, where objects such as buildings or trees obstruct the laser scanner, resulting in incomplete reconstructions of certain areas. These gaps in the data can affect the accuracy of models relying on the dataset and highlight one of the challenges of using synthetic LiDAR scans for detailed urban modeling.

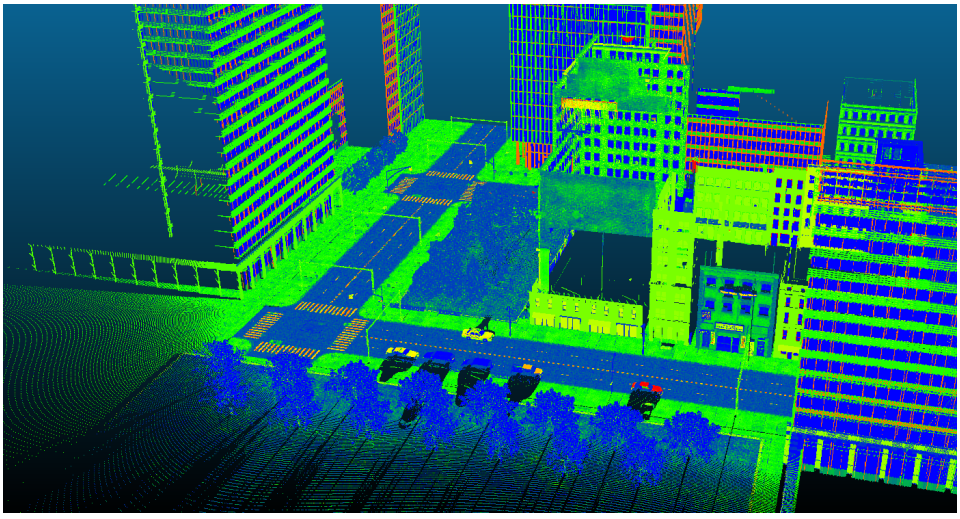

Fig. 4.2 Visualization of the SynthCity Area1 point cloud in CloudCompare.

Despite these limitations, SynthCity provided a useful test dataset for preliminary experiments. However, issues related to format compatibility and missing data made it unsuitable for direct use in the City Digital Shadow pipeline. The need for additional processing, both for file conversion and for handling incomplete areas, limited its practicality within this workflow.

4.2 Mappano Point Cloud

The Mappano point cloud consists of 156 million points and represents a portion of the municipality, with a particular focus on a local school, which was captured with higher resolution. Despite the high point density, the dataset exhibits occlusion shadows, where certain areas lack points due to obstructions during the LiDAR scan. Additionally, unlike SynthCity, this point cloud lacks segmentation and requires extensive cleaning before use.

Fig. 4.3 The Mappano point cloud displayed in CloudCompare.

The initial goal was to use this point cloud to reconstruct a mesh of the neighborhood and import it into CARLA. However, this approach presented several challenges. CloudCompare, the software used for processing, allows mesh reconstruction but does not support texture export, making it necessary to use MeshLab for texturing. During testing, it was observed that MeshLab introduced artifacts in the reconstruction, whereas CloudCompare produced a cleaner but untextured mesh, limiting its usability for a realistic simulation.

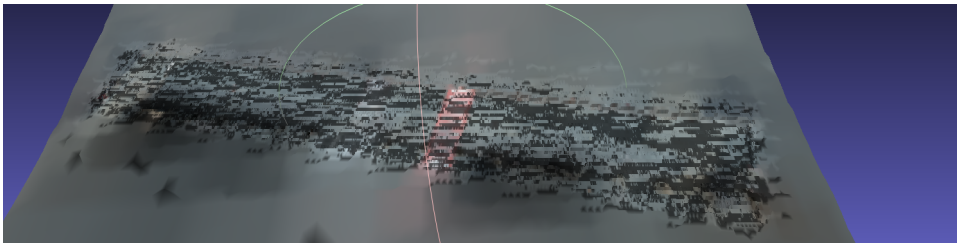

Fig. 4.4 Result of the Poisson mesh reconstruction algorithm applied in MeshLab, highlighting unwanted artifacts generated during the process.

Before generating the mesh, it was necessary to compute the normals of the point cloud, as they are essential for mesh reconstruction. CloudCompare [25] provides two methods for this:

HoughNormals Plugin, which detects normals based on geometric structures.

Normals > Compute, which estimates normals based on neighboring points.

Once the normals were computed, the Poisson Surface Reconstruction algorithm (Plugins > PoissonRecon) was applied to generate the mesh. However, this method tends to close open surfaces, introducing unwanted geometry in urban environments where discontinuities (e.g., roads, open spaces) should be preserved. Increasing the depth parameter enhances detail but exacerbates this issue. A useful strategy to mitigate this problem is the export of scalar fields, which allows filtering out the artificially generated surfaces, leading to a cleaner final result.

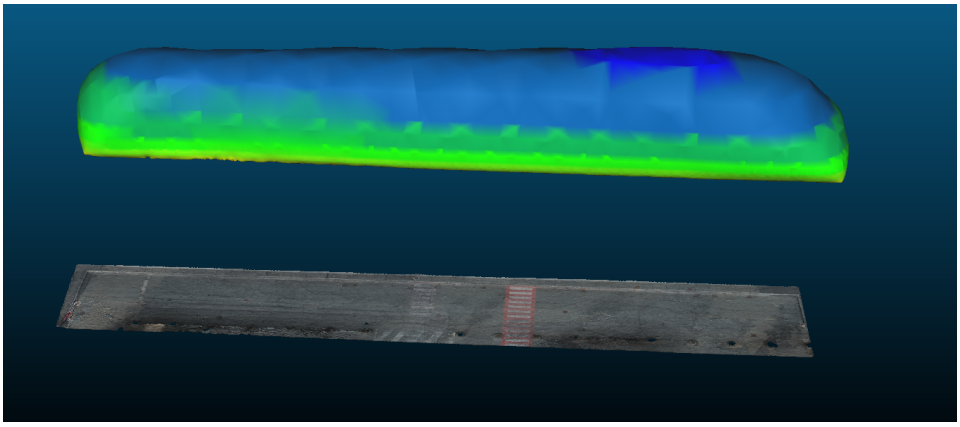

Fig. 4.5 Comparison of the point cloud before and after filtering with scalar fields in CloudCompare to remove artificially closed surfaces.

Given the difficulties in achieving a usable mesh, an alternative workflow was explored. To prepare the dataset for this new approach, the main road was manually cleaned using CloudCompare's selection and deletion tools, removing unnecessary points and refining the data for further processing.

4.3 Integration into Unreal Engine

The LiDAR Point Cloud Support plugin for Unreal Engine was used to create a testable Digital Shadow, enabling the import, visualization, and editing of point cloud data within the engine. This plugin offers multiple coloration and shading techniques, as well as dynamic Level of Detail scaling, which optimizes performance even when working with large datasets.

To integrate the Mappano point cloud into Unreal Engine, it was first converted from .e57 to .ply using CloudCompare. This step was necessary to ensure that the RGB color information was preserved, as Unreal supports point clouds with embedded color data. Unreal Engine allows the import of various point cloud formats, including .xyz, .pts, .las, .laz, and .e57, with different performance characteristics. The global point budget system in Unreal determines the maximum number of points rendered at a given time, balancing visual fidelity and performance.

Fig. 4.6 The Mappano point cloud imported into Unreal Engine through the LiDAR Point Cloud Support plugin.

The plugin provides different scaling methods to manage point rendering:

- **Per Node**: Scales points based on the estimated density of their containing node.
- **Per Node Adaptive**: Adjusts density adaptively based on the current view.
- **Per Point**: Scales points individually based on depth.
- **Fixed Screen Size**: Uses screen-space scaling for consistent sprite rendering.

By utilizing these features, the Mappano point cloud was successfully imported into Unreal Engine, allowing for real-time rendering and interaction while maintaining a reasonable performance-cost balance. However, managing large-scale point clouds within Unreal requires careful configuration of rendering parameters, such as the r.LidarPointBudget, which defines the total number of points displayed on-screen simultaneously.

After analyzing the results obtained from the point cloud visualization, an additional test was conducted by generating a mesh of Mappano in CloudCompare and importing it into Unreal Engine. Since CloudCompare does not support texture export, the resulting mesh was untextured (white) upon import. To address this, a lower-density version of the point cloud was used in Unreal, essentially serving as a color reference for the mesh.

Fig. 4.7 The Mappano mesh and point cloud integrated in Unreal Engine.

The test results were not satisfactory, as the generated mesh still contained imperfections that prevented achieving the desired outcome. Consequently, it was decided to rely solely on the point cloud, limiting the rendering budget to 7 million points simultaneously on screen. To enhance the environment, part of the vehicles and trees from the point cloud were replaced with CARLA assets. This approach aimed to create an explorable virtual space while achieving a distinctive aesthetic in VR.

Fig. 4.8 The Mappano point cloud is combined with CARLA assets in Unreal Engine, creating an interactive simulation environment.

4.4 Road Generation in RoadRunner

To generate the road network of Mappano in RoadRunner, OSM data for the municipality was first downloaded from OpenStreetMap and then imported using the *SD Map Viewer Tool*. This process provided an initial road layout, serving as a foundation for further adjustments.

Once the roads were generated from the OSM data, they needed to be refined and adapted to match the real environment of Mappano. This step was carried out using municipal reference data and RoadRunner's editing tools. The details of this adaptation process are discussed in Section 5.2 (Adaptation for CARLA) in Chapter 5.

Once the desired road network was finalized, it was exported using the CARLA export option in RoadRunner, including the .filmbox (FBX) format. This export was necessary because the generated roads could serve as a more uniform collider for vehicles in CARLA, improving driving physics and interactions.

Before exporting, the *OpenDRIVE Export Preview Tool* was used to compute the OpenDRIVE data, ensuring that the generated road network contained all the necessary information for CARLA's navigation system.

Back in Unreal Engine, the FBX file was imported, while the .xodr file was placed inside an OpenDrive folder within the project's file structure. It is crucial to ensure that the OpenDRIVE file has the same name as the Unreal Engine level, as CARLA relies on this naming convention for proper integration.

Finally, the BP_OpenDrive actor was added to the Unreal scene. By selecting this actor, the Generate Routes option was used to create splines for vehicle navigation, enabling CARLA's autonomous driving system to interact with the imported road network.

Fig. 4.9 The final Mappano map in CARLA integrates LiDAR point cloud data, CARLA assets, and OpenDrive information to create a navigable Digital Shadow simulation environment.

Chapter 5

Matrix City Sample

This chapter examines the City Sample [26], developed by Epic Games, as a case study for integrating large-scale, highly realistic urban environments into simulation platforms. Built with Unreal Engine 5, this sample leverages advanced technologies such as **World Partition**, **Nanite**, and **Lumen**, enabling efficient rendering and dynamic lighting in an extremely detailed cityscape.

The first section provides an in-depth analysis of the structure and key components of the Matrix City Sample, highlighting its technical foundations and the strategies employed to achieve high fidelity and performance. The second section focuses on adapting this environment for CARLA, outlining the integration techniques required to make it compatible with the simulator's framework. The third section presents the additional steps necessary to further refine the map for VR, ensuring an optimized and immersive experience. Finally, the last section explores the use of computer vision techniques to automate parts of the adaptation process, reducing manual effort.

5.1 Overview

The City Sample project represents a significant step forward in the generation of large-scale, high-fidelity urban environments. Originally developed as a demonstration of Unreal Engine 5's rendering and simulation capabilities, it provides a highly detailed digital city that serves as a testing ground for real-time interactive applications. This virtual environment, which draws from the assets and design principles used in The Matrix Awakens: An Unreal Engine 5 Experience, was conceived to explore new methodologies in world-building, procedural city generation, and real-time cinematic production.

"Sixteen kilometers square, photoreal, and quickly traversable, it's populated with realistic inhabitants and traffic. The experience is a tangible demonstration that UE5 offers all the components you need to build immersive, ultra-high-fidelity environments." [27]

While its initial purpose was to demonstrate the potential of next-generation rendering and simulation, the City Sample also offers an adaptable framework for applications beyond cinematic storytelling. In particular, its urban environment provides a valuable base for driving simulation in virtual reality, where realism, spatial coherence, and system responsiveness are critical.

Given the computational complexity of such environments, this thesis adopts the Small City version of the project, which maintains the core structural and visual elements of the original city while optimizing performance for real-time applications. Unlike the Big City variant, which spans approximately 16 square kilometers and was designed for high-end hardware, the Small City provides a more manageable yet equally detailed setting for testing navigation, perception, and user experience in virtual reality.

Furthermore, the transition from conventional level management to a streaming-based architecture ensures that only relevant portions of the environment are loaded at any given time. This optimization is essential in VR applications, where maintaining stable performance and minimizing latency are critical for user immersion. The ability to dynamically manage assets without sacrificing visual fidelity allows for scalable virtual environments that can be tailored to specific simulation needs.

Integrating this environment into a VR-based driving simulator, this thesis aims to assess the level of immersion provided by a procedurally generated urban setting. By analyzing factors such as visual fidelity and system responsiveness, the study explores how real-time rendering and scalable asset management contribute to the perception of presence and realism in virtual driving simulations.

To effectively simulate large-scale urban environments in real-time, it is essential to leverage advanced rendering, asset management, and level streaming techniques. The following subsections provide an overview of the key Unreal Engine 5 technologies that enable the creation and optimization of expansive virtual cities, ensuring both high visual fidelity and performance efficiency.

5.1.1 Lumen

Lumen plays a crucial role in ensuring that lighting within the City Sample environment remains both dynamic and photorealistic. Unlike traditional methods that rely on precomputed global illumination, Lumen enables real-time adjustments to indirect lighting, making it particularly well-suited for interactive and immersive applications such as virtual reality driving simulations. This system adapts to changes in direct lighting and scene geometry, combining ray tracing techniques with screen-space approximations to deliver high-quality results within the constraints of real-time rendering.

In the City Sample, Lumen operates using **Hardware Ray Tracing**, which enhances the fidelity of reflections and indirect illumination by tracing rays against actual geometry rather than relying solely on lower-resolution approximations. This method proves particularly beneficial for rendering complex urban environments where lighting conditions vary significantly between open streets, enclosed alleyways, and interior spaces. Moreover, emissive materials, such as illuminated signs or streetlights, contribute to scene lighting dynamically, reinforcing the overall realism of the simulation.

Given the need for real-time performance, Lumen employs a combination of different ray-tracing methods, prioritizing efficiency while maintaining visual fidelity. It utilizes Screen Traces as an initial pass to gather lighting information before applying more accurate techniques, such as Signed Distance Fields for software-based ray tracing or Hardware Ray Tracing for systems that support it. In the context

of this thesis, the integration of Lumen within the VR driving simulator ensures that lighting conditions respond dynamically to environmental changes, enhancing the user's sense of presence and depth perception in the virtual city.

Fig. 5.1 Effect of Lumen Global Illumination and Reflections: disabled (left) vs. enabled (right).

5.1.2 Virtual Shadow Maps

Virtual Shadow Maps (VSM) are an advanced shadow mapping technique introduced in Unreal Engine 5, offering consistent, high-resolution shadows in complex, dynamically lit environments. VSM works seamlessly with other UE5 technologies like Nanite, Lumen, and World Partition, making it particularly effective for virtual reality simulations that require realistic lighting and shadowing.

Traditional dynamic shadowing techniques often face limitations in maintaining high-quality results over large areas without compromising performance. VSM addresses this challenge by providing a unified shadowing system that intelligently applies quality where needed most. This allows for consistent shadow quality on both small and large objects, even over greater distances, with natural soft penumbra and contact hardening effects. This capability is essential for achieving visual realism in virtual simulations, where precise and natural lighting is critical for creating a believable experience. VSM ensures high-quality shadows without significantly impacting performance, making it a valuable tool for demanding real-time applications.

Fig. 5.2 Comparison of Cascaded Shadow Maps (left) and Virtual Shadow Maps (right).

5.1.3 Post Processing Local Exposure

Local Exposure is a new technique in Unreal Engine 5 that allows for dynamic, scene-specific exposure adjustments. Unlike the traditional global exposure system, this technique automatically modifies exposure locally, within parameters set by the artist, to preserve details in both highlights and shadows. In environments with high dynamic range (HDR) content, such as those featuring dynamic lighting, global exposure alone often struggles to balance bright highlights with dark shadows effectively.

In the context of this thesis, where a VR-based driving simulator is used to test a procedurally generated urban environment, Local Exposure plays a crucial role in ensuring consistent visual quality. In environments like the City Sample, where lighting can be dynamically adjusted or changed depending on the time of day, areas of the scene may become overexposed or underexposed. Local Exposure addresses this issue by applying localized adjustments, ensuring that each part of the scene is exposed correctly, even when the overall lighting varies. This method helps maintain a consistent visual experience in real-time applications, where per-scene lighting adjustments might not always be feasible, as is the case in an open world like the City Sample, where users can freely navigate and explore the environment.

Fig. 5.3 Comparison of Post Processing Local Exposure: with (right) and without (left) applied adjustments.

5.1.4 Nanite

Nanite is a revolutionary technology in Unreal Engine 5, enabling the use of virtualized geometry for rendering highly detailed static meshes. In the City Sample, Nanite is applied to all Static Meshes, eliminating the need for traditional LOD systems. By rendering pixel-scale detail and only processing geometry that can be perceived, Nanite optimizes performance without sacrificing visual fidelity. Similar to how Virtual Texturing handles texture detail, Nanite ensures that the system only computes the necessary detail for what is visible to the camera, making it highly efficient.

Nanite's dynamic mesh format and rendering capabilities allow for real-time adjustments to the level of detail as the player moves through the world. This adaptive system automatically adjusts the level of detail for objects based on their proximity to the camera, rendering high levels of detail for nearby objects while reducing the detail for those farther away. Non-visible geometry is culled, further enhancing performance.

In the City Sample, which consists of billions of polygons from tens of thousands of objects, Nanite enables the use of film-quality assets in real-time applications. This technology simplifies the creation of large-scale environments, requiring little to no additional setup beyond enabling Nanite for static meshes. The City Sample thus demonstrates how Nanite can effectively manage and render vast, complex urban

environments with exceptional detail and efficiency, a critical factor for immersive simulations in VR, as explored in this thesis.

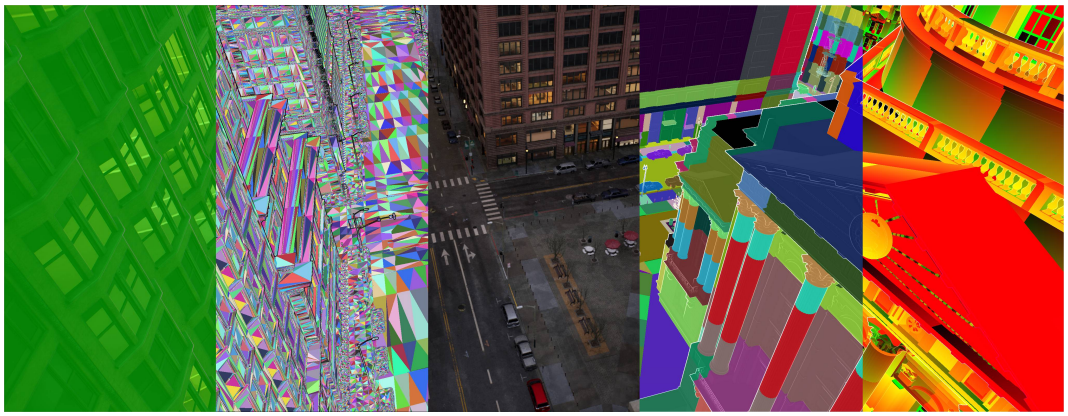

Fig. 5.4 Nanite Visualization Methods.

5.1.5 Temporal Super Resolution

The increasing complexity of large, dynamic environments, like the ones found in the City Sample project, demands high levels of detail while maintaining performance efficiency. Rendering at native 4K resolution is a common challenge when aiming for cinematic quality in vast open worlds with millions of polygons.

Temporal Super Resolution (TSR) is a next-generation anti-aliasing method designed to balance the visual fidelity of high-resolution rendering with the performance constraints of modern hardware. By utilizing a temporal upscaling algorithm, TSR allows for rendering at lower input resolutions while still delivering images with quality approaching native 4K resolution. This approach significantly reduces GPU frame time, enabling higher performance without sacrificing image clarity, as seen in the City Sample environment.

5.1.6 World Partition

In the context of simulating real-world environments, such as the City Sample, managing vast, dynamic spaces is crucial. World Partition is a system in Unreal Engine 5 that addresses the challenges of creating and managing large-scale environments by dynamically dividing the world into smaller, manageable sections. This

allows for more efficient handling of assets as players traverse large areas without compromising performance.

Previously, developers needed to manually manage levels and sublevels, carefully streaming in and out assets based on proximity. The World Partition system automates this process by dividing the world into grid-based cells, which are then dynamically loaded and unloaded based on the player's location. This system ensures that only relevant parts of the environment are rendered at any given time, thus optimizing performance in large urban settings like those used in this thesis.

For VR-based driving simulations, where real-time performance and responsiveness are paramount, World Partition's seamless data management and streaming are key in creating an immersive and efficient experience. This technology enables expansive and interactive virtual cities without the performance limitations of traditional methods.

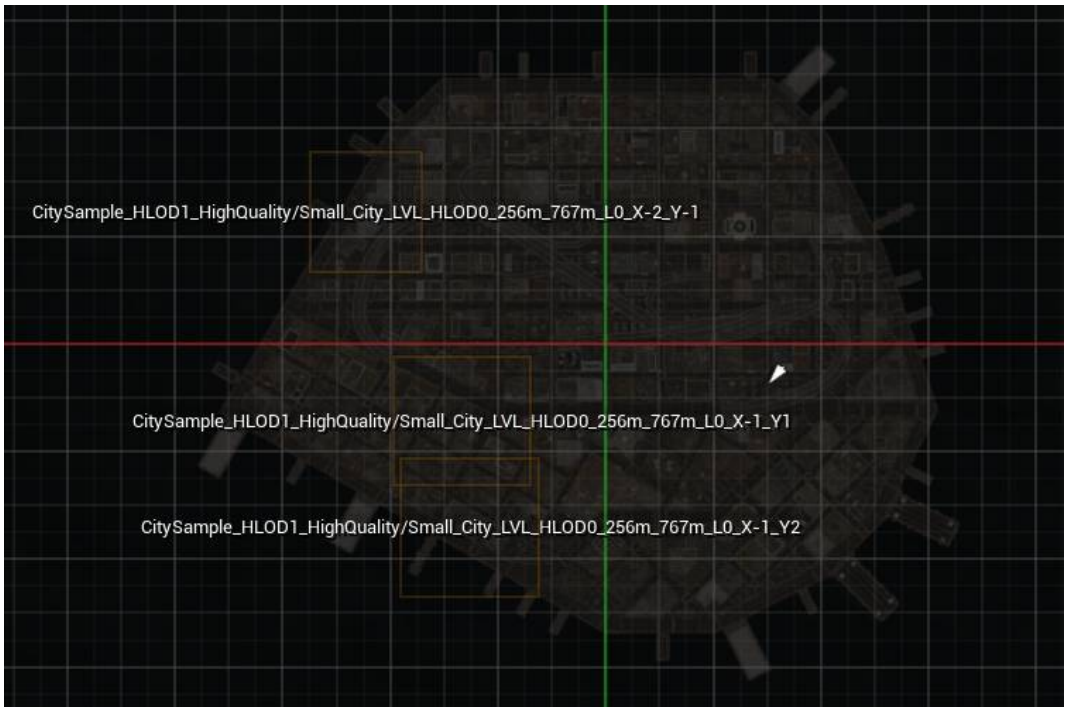

Fig. 5.5 World Partition Map in Editor.

5.1.7 HLODs & Streaming

A key component of World Partition is Hierarchical Level of Detail (HLOD), which enhances performance by simplifying distant objects. HLOD works by grouping adjacent actors and replacing them with a single static mesh when they are far enough from the player or streaming source. This significantly reduces the number of rendered objects while maintaining visual consistency.

HLOD generation is based on the lowest Level of Detail available for each asset. If a custom asset does not have predefined LODs, the system generates an HLOD from the full-resolution model, which can be computationally expensive. The LOD system follows a hierarchy where LOD0 represents the highest fidelity asset, LOD1 is a lower-resolution version, and subsequent LODs progressively reduce mesh complexity.

The effectiveness of HLOD is directly linked to draw distance. When combined with World Partition Streaming, HLODs become visible at a minimum distance determined by the configured grid loading settings. If they are loaded only when they appear small on the player's screen, they occupy minimal memory while still maintaining an immersive environment.

This system plays a crucial role in optimizing large-scale open worlds, making it particularly relevant for simulations and digital twin applications that require both extensive environments and real-time performance.

5.1.8 Data Layers

In Unreal Engine 5, Data Layers complement the World Partition system by providing an additional layer of control over which assets are loaded based on specific criteria. While World Partition handles the automatic streaming of world cells, Data Layers allow for the grouping and management of objects into distinct categories, which can be individually loaded or unloaded as needed. This approach replaces the older Layer system in Unreal Engine 4, which often required manual curation to manage content efficiently.

In the context of the City Sample, Data Layers are used to organize various world elements, such as procedural objects, rooftop props, and freeway assets. The Data Layers Outliner provides an intuitive interface to manage these layers, allowing

developers to toggle visibility, add, or remove objects based on the requirements of the simulation. This functionality supports dynamic loading and unloading in both the Editor and during gameplay.

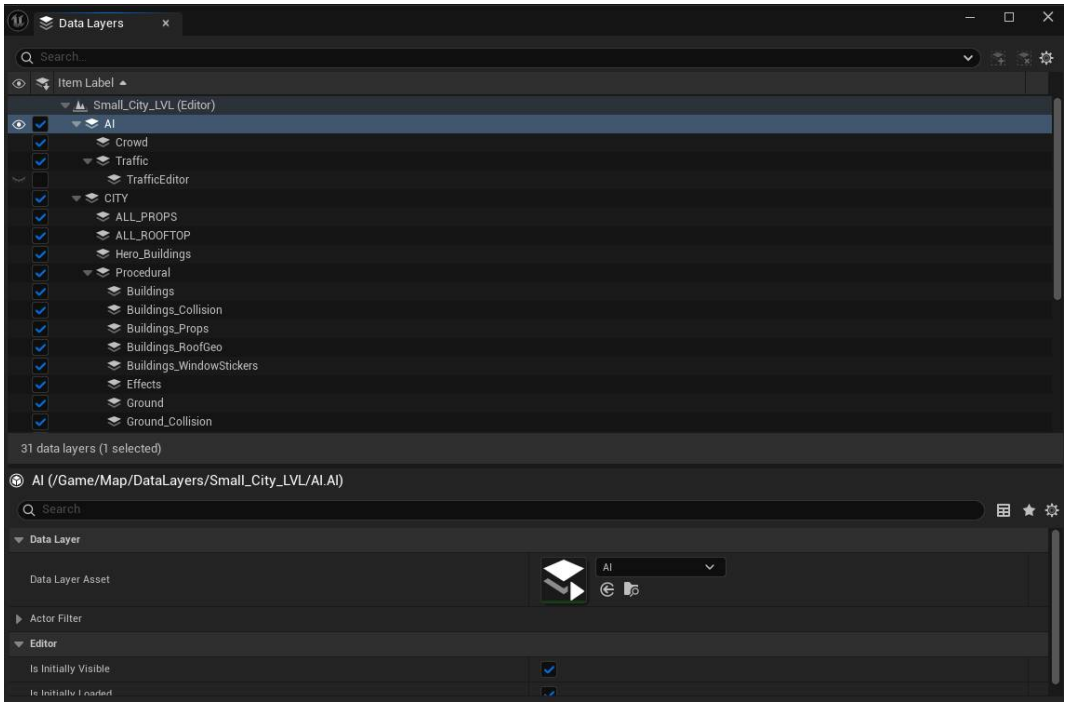

Fig. 5.6 Data Layers Outliner.

5.1.9 Procedurally Generated City

The procedural generation approach used in the City Sample project showcases how large-scale urban environments can be efficiently created using **SideFX's Houdini** in combination with Unreal Engine 5. Every element of the city, from road networks and freeway systems to buildings and street furniture, is generated procedurally, allowing for the rapid creation of detailed urban landscapes. This process is particularly relevant for simulations requiring large, highly detailed city environments without the need for extensive manual asset placement, which is precisely the case in this thesis.

In the workflow, Houdini is responsible for generating the city structure, providing a rich set of metadata that Unreal Engine 5 can use to refine the final layout. The **Rule Processor**, a tool developed specifically for City Sample project, converts point

cloud data from Houdini into thousands of instances in Unreal Engine, ensuring an efficient rendering pipeline. This procedural approach enables the generation of diverse urban layouts while maintaining consistency and performance.

For this thesis, procedural generation offers an effective method for creating detailed urban environments for VR-based driving simulations. While the focus is on smaller, test-scale environments rather than large-scale cityscapes, leveraging Houdini and procedural techniques can streamline the creation of realistic environments.

5.1.10 Houdini City Generation Project

The City Sample project provides a Houdini project that includes all the necessary tools and assets to generate cities procedurally. This project serves as a foundation for creating urban environments with a high level of automation, allowing users to define city layouts, road networks, and building placements with minimal manual intervention.

In Houdini, the city generation process is based on curves that define key structural elements such as the overall city dimensions, the freeway layout, and the City Arteries. These curves act as the framework upon which the rest of the procedural generation is built, ensuring logical road connectivity and urban density distribution.

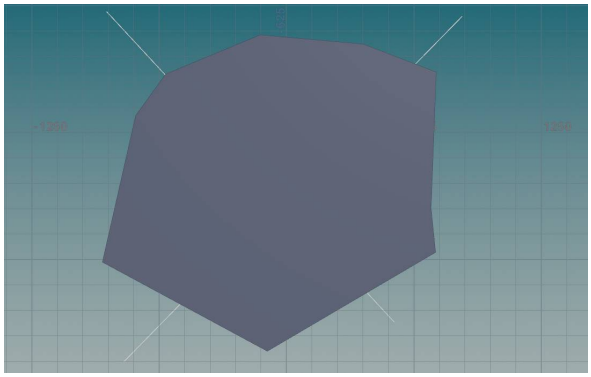

Fig. 5.7 Curves used to define the city layout in Houdini. The closed grey curve represents the city boundary, while the two lines indicate the primary City Arteries that structure the road network.

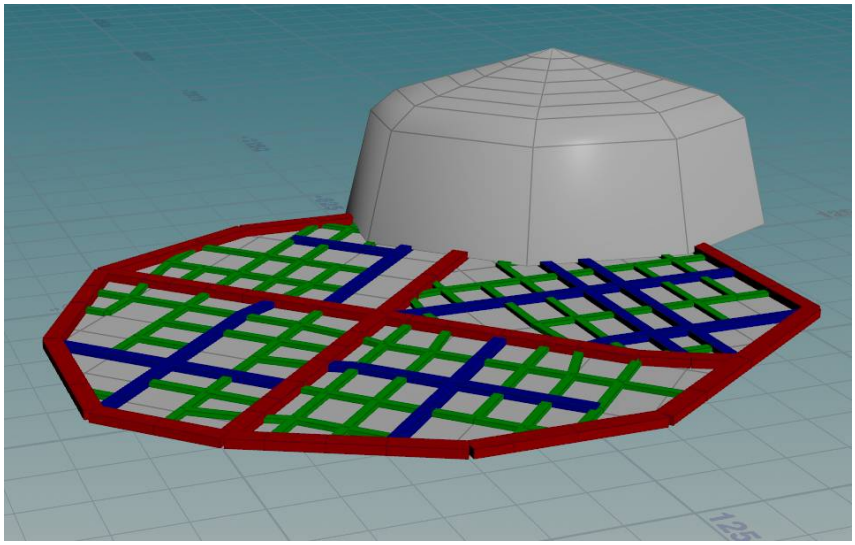

Fig. 5.8 City zones added in Houdini to define building heights, such as skyscrapers, within the procedural city generation workflow.

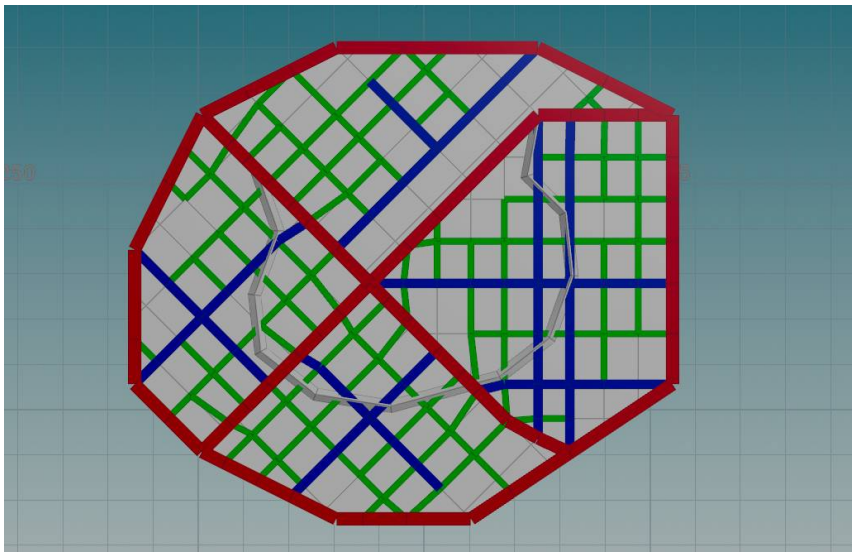

Fig. 5.9 Preliminary city layout representation in Houdini. The added curve defines the freeway, integrating it into the procedural city generation process.

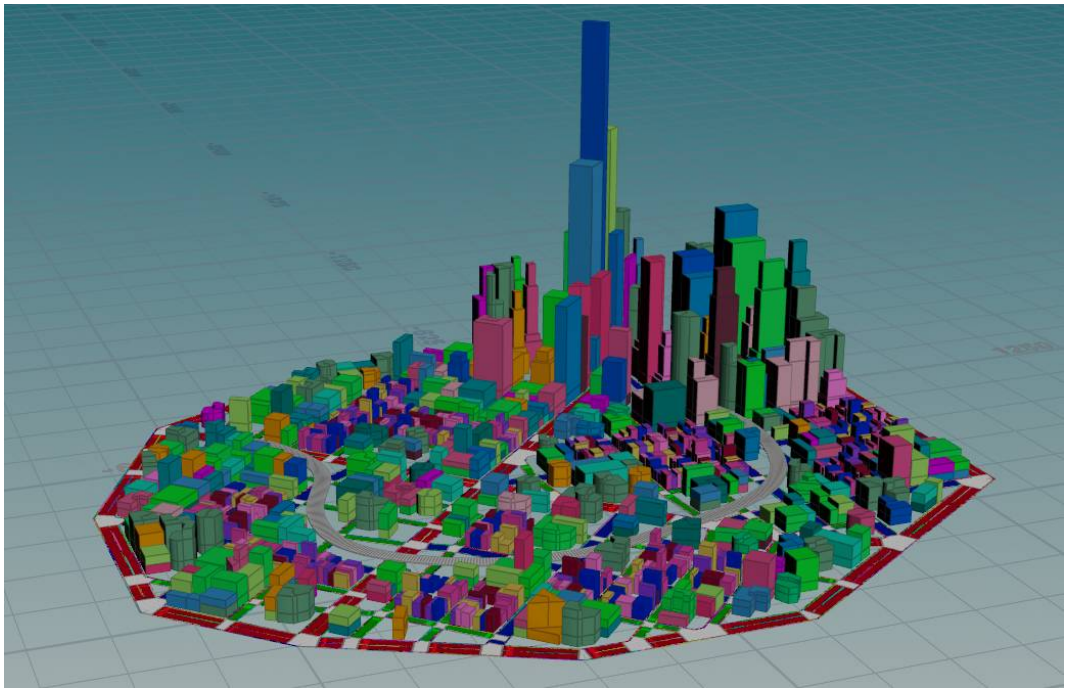

Fig. 5.10 Final city layout generated in Houdini using the City Processor, showcasing the complete procedural workflow.

Once the city layout is defined, Houdini generates a point cloud representation that includes metadata for buildings, roads, and additional street elements. This data is then exported and processed in Unreal Engine 5 using the Rule Processor, a tool developed to convert procedural data into optimized assets within the game engine.

For a more detailed breakdown of the Houdini city generation process, Epic Games provides a dedicated document [28] that outlines each step. Additionally, a second document [29] focuses on the import workflow into Unreal Engine, explaining how to convert and optimize the procedural city data for real-time rendering.

5.2 Adaptation for CARLA

The adaptation of the Matrix City Sample for CARLA was carried out in three main phases. The first phase involved gathering reference data directly from Unreal Engine to guide the subsequent steps. The second phase focused on the creation of road networks using RoadRunner, ensuring compatibility with CARLA's road system. Finally, the third phase integrated the generated data into CARLA, enabling its use within the simulator.

5.2.1 Reference Extraction

The first step in adapting the city environment for CARLA involved capturing a structured set of reference images directly within Unreal Engine. To achieve this, a Blueprint system was developed to automate the process using an orthographic camera with an orthowidth of 7000. The goal was to generate a **high-resolution mosaic** of images that could serve as a visual reference for the subsequent steps of road creation and integration.

To define the coverage area, two input parameters were introduced: the number of images to be captured along the X and Y axes. In this case, 5 images were taken along the X-axis and 10 along the Y-axis, thus determining the grid subdivision of the map. This configuration ensured the capture of a total of 50 images (5×10), with the mosaic centered around the Pawn's (Player Start) origin. Additionally, the algorithm used for image capture was iterative, progressively adjusting the camera's position to determine the optimal offset values between each shot. These values were empirically derived through a series of trials to ensure precise alignment and minimize overlap between consecutive images. The final offset values used were 13450 for the X-axis and 7000 for the Y-axis.

To maintain consistency across different display settings, the scene was configured to run in windowed mode with a fixed resolution of 1920×1080, preventing any unintended variations in output. Additionally, to simplify the visualization and focus on the essential ground elements, Data Layers — discussed in Subsection 5.1.7 — were leveraged to selectively hide unnecessary assets, leaving only the Ground and Ground Decals active for the next steps.

This structured approach ensured that a precise and organized set of images was generated, providing a reliable reference for the subsequent road network reconstruction.

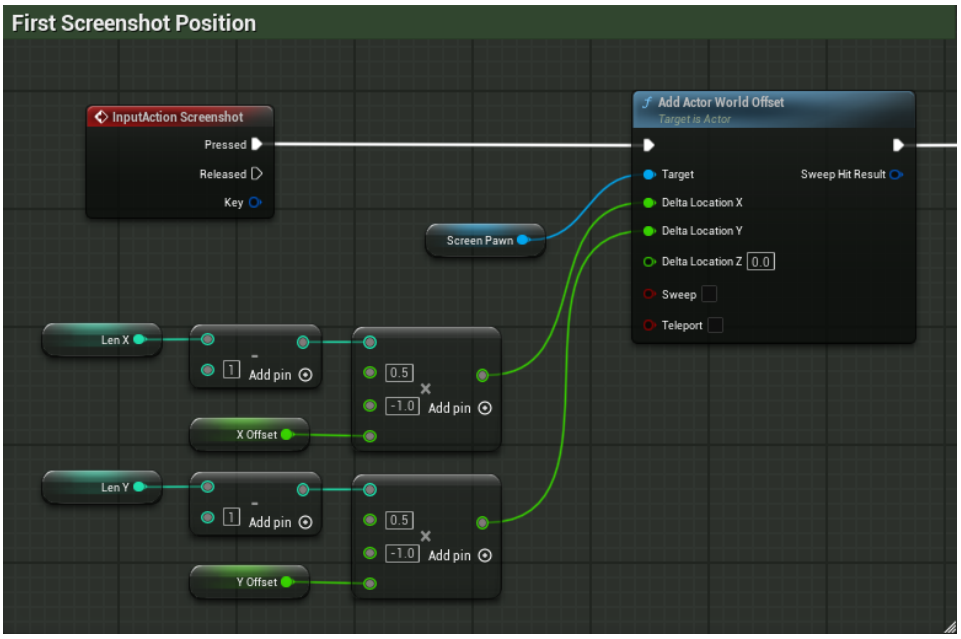

Fig. 5.11 The figure illustrates the initial offset application, positioning the camera at the top-left corner of the mosaic grid to begin the image capture process.

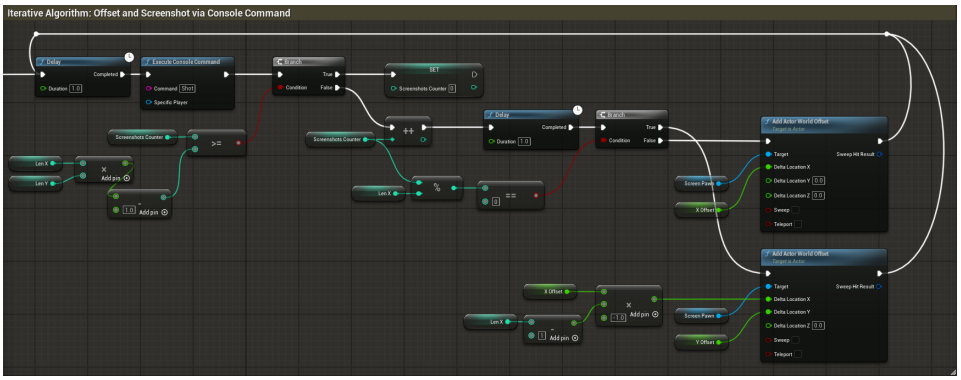

Fig. 5.12 The figure shows the iterative algorithm used to displace the camera across the grid, applying offsets between each shot and utilizing a console command to capture the images at each step.

Subsequently, the images were captured and processed through a Python script that reconstructs the mosaic. The script takes as input the number of images along the

X and Y axes, along with overlay values, which are independent from the previous ones and used to apply corrections. In this case, the values for the images were set to 5 along the X-axis and 10 along the Y-axis, with overlay values of 8 for the X-axis and 0 for the Y-axis.

Listing 5.1 Script for processing images and applying overlay adjustments.

```
def create_mosaic(img_folder, out_path, len_x, len_y, ol_x, ol_y):
    image_files = [f for f in os.listdir(img_folder)]
    images = []

    for f in image_files:
        image_path = os.path.join(img_folder, f)
        img = Image.open(image_path)
        images.append(img)

    if len(images) < len_x * len_y:
        print("Insufficient number of images for the specified grid.")
        return

    width, height = images[0].size

    mosaic_width = width * len_x -(len_x -1) * ol_x
    mosaic_height = height * len_y -(len_y -1) * ol_y
    mosaic = Image.new('RGB', (mosaic_width, mosaic_height))

    for row in range(len_y):
        for col in range(len_x):
            index = row * len_x + col
            if index >= len(images):
                break
            x_offset = col * (width -ol_x)
            y_offset = row * (height -ol_y)
            mosaic.paste(images[index], (x_offset, y_offset))

    mosaic.save(out_path)
    print(f'Mosaic created and saved as {out_path}')
```

Fig. 5.13 This figure shows the reconstructed mosaic obtained by combining the captured images. The grid is generated based on the defined parameters, with the overlay corrections applied: len_x = 5; len_y = 10; ol_x = 8; ol_y = 0.

5.2.2 Road Definition and Export

In the second phase of adapting the City Sample map for CARLA, the mosaic created in the previous step was used as a reference for reconstructing the road network. Specifically, the mosaic image was imported into RoadRunner and set as the Aerial_Image for the project.

To ensure the image was correctly aligned with real-world coordinates, a Custom Projection was applied. The projection was defined in Well-Known Text (WKT) format, a standard for representing coordinate system definitions in a text-based format. This enabled proper projection of the image within the RoadRunner environment. The Scene's Projection was selected to configure the Custom Projection settings.

Next, the Meter Offset was adjusted to account for the positioning of the Player Start, which is placed at the center of the mosaic. It was important to note the differences in coordinate systems between RoadRunner and Unreal Engine: the Y-axis is inverted between the two platforms, and the units of measurement change from meters in RoadRunner to centimeters in Unreal Engine. For instance, if the Player Start position in Unreal Engine is (-32000, -12500), the corresponding offset to be applied in RoadRunner is (-320, 125). Applying this offset is crucial since the OpenDrive data to be generated is tightly linked to the coordinates relative to the origin.

Finally, the Resolution setting was empirically determined, with a value of 0.07036, based on the parameters chosen in the previous steps.

The same offset was then applied to the World Origin by selecting the World Settings tool and adjusting the Center with the previously calculated offset. This ensured that the entire scene in RoadRunner was correctly aligned with the Player Start position in Unreal Engine.

Next, the Aerial_Image was placed in the workspace. Once the image was selected, the World Settings tool was used to adjust the workspace size by applying the *Fit Bounds To Selected* command. This command automatically scaled the workspace to fit the selected image, ensuring that the reference image occupied the correct area in the project and was aligned with the world coordinates.

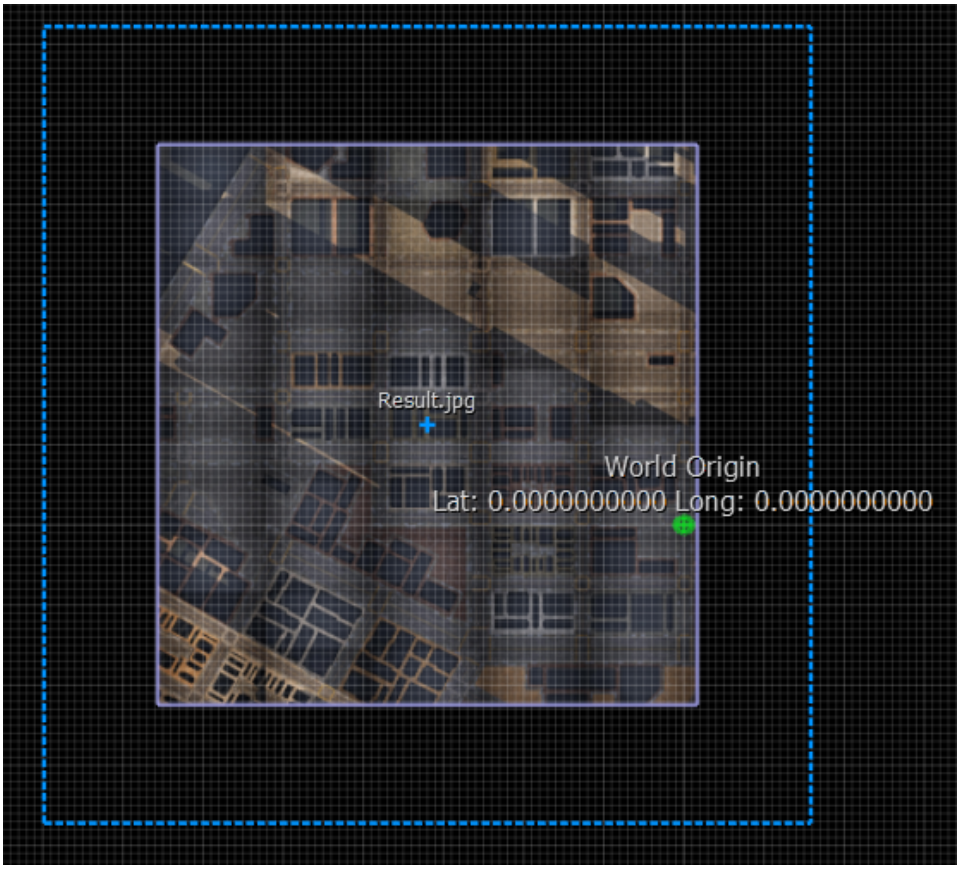

Fig. 5.14 Aerial image set as a reference in RoadRunner.

Fig. 5.15 RoadRunner toolbar, displaying the essential tools used for road creation, editing, and export.

With the reference image properly positioned, the next step was to construct the road network using the Aerial_Image as a guideline. Roads were created using the **Road Plan Tool** [A], which allowed for the placement of road segments aligned with the reference. Once the basic structure was in place, the **Road Chop Tool** [B] was used to divide longer segments into smaller sections, enabling finer adjustments and facilitating the creation of intersections.

To define speed regulations, the **Road Speed Limits Tool** [C] was applied to assign appropriate limits to each road segment. The intersections were then configured

using the **Custom Junction Tool** [D], which allowed for the creation of complex junctions. Additionally, the **Maneuver Tool** [E] was used to specify permitted vehicle movements, ensuring that all turning restrictions and lane connections were correctly defined.

Traffic control elements were incorporated through the **Signal Tool** [F], which enabled the assignment of stop signs, yield signs, or traffic lights at intersections. Lanes were adjusted through several tools: the **Lane Tool** [G] enabled the selection and editing of lanes, including the direction of travel; the **Lane Width Tool** [H] modified lane widths to fit realistic road dimensions; the **Lane Add Tool** [I] allowed for the addition of extra lanes when required; and the **Lane Carve Tool** [J] was used to narrow the roadway where necessary.

Once the road network was fully constructed, the **OpenDRIVE Export Preview Tool** [K] was used to generate the OpenDRIVE data. These files contained all the necessary information for CARLA's simulation, including road geometry, lane connectivity, and traffic regulations, ensuring that autonomous vehicles could navigate the environment realistically.

For the integration of the City Sample map, which already included traffic signals and signage as separate assets, traffic signals were intentionally excluded from the OpenDRIVE export. This was because CARLA manages traffic signals using trigger boxes that notify autonomous vehicles of nearby traffic control elements. Had the traffic signals been included in the OpenDRIVE export, they would have been dynamically generated in the simulation. However, by omitting them, manual adjustments were required within Unreal Engine to ensure the correct operation of the traffic system.

Fig. 5.16 Road network created in RoadRunner with OpenDRIVE data preview.

To finalize the export process in RoadRunner, the CARLA export option was selected, including only the .udatasmith and .xodr files. The .fbx file containing the road assets was excluded, as it was not required for this integration.

5.2.3 CARLA Data Integration

Once the .xodr data has been exported, it is crucial to rename the file to match the corresponding level name in Unreal Engine. This ensures proper recognition and integration within the project. The .xodr file is then placed inside the project's OpenDrive folder, where it can be imported into the map using the OpenDriveActor.

Within the OpenDriveActor Detail Panel, the Generate Routes option can be selected, along with Generate Vehicle Spawn Points. Enabling these options automatically generates splines that define the road network and creates spawn points at

intersections. This highlights the importance of correctly setting the world origin in RoadRunner beforehand, since OpenDRIVE data cannot be repositioned within the level, it must be accurately placed during the export process to ensure proper alignment in Unreal Engine.

As mentioned earlier, OpenDRIVE data alone is not enough to handle intersections properly, as it does not manage traffic signals. To ensure vehicles respond correctly to traffic lights, trigger boxes must be used to inform them of the current signal state.

This was achieved by modifying and adapting three key blueprints: BP_TrafficLightNew, BP_CustomTriggerComponent, and BP_TrafficLightGroup. A crucial aspect of BP_TrafficLightGroup is that it organizes traffic lights into phases, ensuring that only one phase is active at a time. This means that in a standard setup, no two traffic lights within the same group will turn green simultaneously.

However, at four-way intersections, it was necessary to allow vehicles traveling in opposite directions along the same road axis to move at the same time. To accomplish this, instead of using a single TrafficLightGroup, each intersection was assigned two separated BP_TrafficLightGroup instances.

For example, consider an intersection with four traffic lights: A, B, C, and D, where A and C control one direction, and B and D the perpendicular direction. To ensure synchronized movement, one TrafficLightGroup managed A and B, while the other controlled C and D. This setup guaranteed that vehicles traveling along the same road axis (A and C, or B and D) received green signals simultaneously. Any other configuration — such as grouping A and D together — would have resulted in conflicting traffic phases.

To integrate the City Sample traffic light assets, the modified version of the BP_TrafficLightNew blueprint was first adjusted by replacing the Static Mesh with the desired model. Then, the blueprint was further modified to assign the dynamic version of the material controlling the lights: *M_StopLight_Main*. This modification allows the traffic light state to be updated by adjusting the *Crosswalk Control* parameter, which controls the signal phase: a value of 1.0 corresponds to green, 0.5 to yellow, and 0.0 to red.

The BP_CustomTriggerComponent only required modifications to update its references to the newly modified BP_TrafficLightNew actor. This ensured that

the trigger component correctly interacted with the updated traffic light system, maintaining proper signal behavior within the simulation.

Fig. 5.17 comparison between CARLA's default traffic lights (top) and the City Sample traffic lights (bottom), highlighting differences in design and detail.

5.3 Optimization for Virtual Reality

The optimization of the City Sample map played a crucial role in achieving a balance between visual fidelity and performance, particularly in the context of virtual reality applications. One of the most significant steps in this process was the calculation of Hierarchical Level of Detail, which allowed for a substantial reduction in rendering complexity by merging multiple assets into simplified versions displayed at a distance. This approach proved particularly beneficial in reducing draw calls and improving overall performance.

A key consideration when using HLODs is their impact on Unreal Engine's Scalability Settings. Once HLODs are generated, these settings lose much of their effectiveness because the HLODs are precomputed with specific rendering parameters that remain fixed unless they are recalculated. This means that modifications to scalability settings after HLOD generation will not influence the appearance of objects represented by HLODs, making it essential to determine the optimal settings before initiating the HLOD computation.

To implement HLODs effectively, two hierarchical levels were created. The first level involved generating a new Open World level, which provided the foundation for the hierarchical structure. Building upon this, a second level was introduced as a child layer, using the "Approximated Mesh" type, which works seamlessly with Nanite. This integration allows Unreal Engine to dynamically manage the complexity of geometries, ensuring that only the necessary level of detail is rendered based on the viewer's distance.

Fig. 5.18 A comparison of scene rendering in Unreal Engine without (top) and with (bottom) HLOD.

Before HLOD generation, additional optimizations were applied by carefully tuning the Engine Scalability Settings. Given the visual complexity of the City Sample map, particular attention was given to elements that significantly impact the VR experience. Anti-aliasing was set to Epic to reduce jagged edges and shimmering, which are particularly noticeable in VR. Despite this adjustment, aliasing remains somewhat intrusive, especially over long distances. Reflections were another crucial factor, given the reflective nature of the environment. During testing, it became apparent that setting reflections to High introduced noticeable artifacts in VR. This led to a choice between Medium and Epic, with the latter being selected to maintain a higher level of realism, despite the increased computational load.

Other settings were adjusted to balance quality and performance. Global Illumination and Shadows were both set to High, ensuring a realistic representation of lighting and depth without excessively straining performance. Effects were also kept at High to preserve environmental realism. The View Distance was configured at the Far setting to prevent abrupt pop-in effects, which can be particularly disruptive in VR. In contrast, parameters such as Foliage, Shading, Landscape, Post-Processing,

and Texture were set to Medium, as their impact on the overall visual experience was deemed less critical compared to other factors.

Several additional settings were specifically tailored to enhance VR performance. Occlusion Culling was enabled to optimize rendering efficiency by ensuring that objects outside the player's view were not processed. Lumen and Nanite were both utilized to enhance real-time lighting and optimize high-detail mesh rendering. To support these technologies, DirectX 12 was selected as the default RHI. Additionally, Static Lighting was disabled in favor of a fully dynamic lighting approach, which is more suitable for large open-world environments. Virtual Textures were also enabled to improve texture memory management, allowing for more efficient handling of high-resolution assets.

Regarding reflections, Lumen was chosen as the primary reflection method, with the Reflection Capture Resolution set to 128 to balance performance and quality. Similarly, Dynamic Global Illumination was configured to use Lumen, ensuring realistic indirect lighting. Virtual Shadow Maps were enabled as the shadow mapping method to enhance shadow fidelity and reduce artifacts. Another key optimization was the activation of Stereo Foveation, with the level set to Medium and Dynamic Foveation enabled. This technique reduces rendering resolution in the peripheral areas of the field of view, optimizing performance without compromising visual clarity in the central vision.

One feature that was deliberately left disabled was Instanced Stereo. While this option is designed to reduce GPU workload by merging the two render passes into a single process, it introduced an issue where decals were only rendered in one eye, breaking immersion in VR. Given this drawback, Instanced Stereo was not used in the final configuration.

In conclusion, the combination of HLOD calculations, fine-tuned scalability settings, and targeted VR optimizations allowed for a well-balanced environment that maintains high visual quality while ensuring stable performance in VR. These adjustments provide a solid foundation for further refinements, particularly in addressing remaining anti-aliasing limitations and optimizing reflections to enhance overall immersion.

5.4 SPIN

SPIN (Spatial and Interaction Space Graph Reasoning) [30] is a deep learning model designed for road extraction from high-resolution aerial images. Traditional convolutional neural networks (ConvNets) often struggle with this task due to their limited ability to capture long-range dependencies between road segments and their difficulty in distinguishing roads from surrounding elements such as vegetation, buildings, or shadows. To address these challenges, SPIN introduces a graph reasoning approach that operates in two distinct spaces: the spatial space, which improves connectivity between separate road segments, and the interaction space, where image features are projected into a latent space that facilitates the separation of roads from other topographical features.

The model is built on a stacked hourglass architecture, which combines convolutional layers with residual connections to extract and refine features at different levels. Additionally, it employs the SPIN Pyramid, a multi-scale reasoning mechanism that enhances segmentation accuracy by processing information at different resolutions. SPIN has demonstrated strong performance in road segmentation tasks, achieving high accuracy even in challenging conditions where roads are partially occluded or have varying widths. Another advantage of this approach is its computational efficiency, as it requires fewer resources compared to transformer-based models while still achieving competitive results.

A possible application of SPIN in the context of this work is the automatic extraction of road networks from satellite or digital images, with the goal of converting the output into a format compatible with OpenStreetMap. The segmentation output is a binary mask distinguishing roads from other areas, but to be useful in GIS applications, it needs to be processed into a structured road network graph composed of nodes and connections. This graph can then be exported in GeoJSON format, a widely used standard for geographic data, and subsequently converted into OSM data using tools like osmium or ogr2osm. Automating this process could significantly reduce the manual effort required to generate detailed road maps, whether for real-world applications or for digital twin simulations.

The SPIN model is publicly available on GitHub, although its training process is not fully documented. Despite this, it represents a promising approach for auto-

mated road extraction, with potential applications in autonomous driving, digital cartography, and the generation of realistic simulation environments.

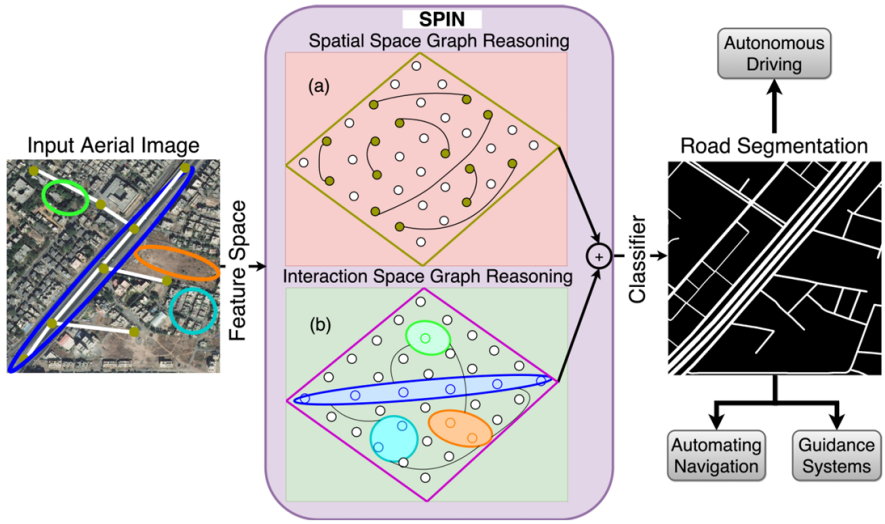

Fig. 5.19 Road segmentation pipeline using SPIN, from aerial image input to extracted road network for autonomous driving and navigation systems. [30]

Chapter 6

Enhancing Immersion in the Simulation

To improve the realism and immersion of the driving experience, additional hardware components were integrated into the simulation setup. These elements were designed to provide *physical feedback*, making the interaction with the virtual environment more natural and responsive.

The system included a **force feedback steering wheel**, an **inertial seat**, and a **pedal set** with distinct mechanisms for acceleration, braking, and clutch control. Each of these components contributed to a more engaging and dynamic simulation, enhancing the perception of vehicle behavior and road conditions.

This chapter explores the role of these haptic feedback systems and their integration into Unreal Engine to enhance the overall driving experience.

6.1 Steering Wheel and Force Feedback Integration

The steering wheel plays a fundamental role in the realism of a driving simulation, as it provides *haptic feedback* that allows drivers to perceive road conditions, tire grip, and vehicle dynamics. While visual information is the primary source of feedback for a driver, additional sensory cues, such as steering torque and resistance, significantly enhance the perception of vehicle behavior. As noted in the literature, *"although drivers obtain a substantial amount of information for driving from vision, information from other sensory modalities may also provide relevant information about the state of the car or even the surrounding environment"* [31].

A key aspect of force feedback steering systems is their ability to *simulate torque variations based on road interactions*. These forces allow the driver to experience understeer, oversteer, and road irregularities, making the simulation feel more natural and engaging. Studies have shown that *"the addition of steering torque decreased steering variance when the driver is controlling the vehicle after a turn or skid"* [31], emphasizing the importance of realistic force feedback in maintaining vehicle control.

Beyond basic force feedback, high-fidelity simulation environments must ensure that the steering system accurately replicates real-world vehicle behavior. The interaction between the steering wheel, road surface, and vehicle physics is critical to a convincing experience. According to research, *"a fundamental haptic cue is the feedback force at the steering wheel. It renders the vehicle–road interaction and is considered very important for driving a vehicle"* [32]. However, achieving this level of fidelity requires precise calibration of damping, inertia, and resistance, ensuring that the driver receives realistic counterforces when maneuvering the vehicle.

In this project, the force feedback steering wheel was integrated into the simulation environment to enhance driver immersion and vehicle control. The system was calibrated to reflect realistic torque resistance and dynamic steering responses, ensuring that users could perceive road grip variations and vehicle weight shifts naturally.

6.2 Motion Seat Integration

Motion seats play a crucial role in driving simulations by replicating the forces experienced in real-world driving, thereby enhancing the user's sense of presence and control. Unlike purely visual or force-feedback-based systems, motion platforms introduce kinesthetic feedback, allowing users to perceive **acceleration**, **braking**, and **lateral forces** in a way that closely mimics reality. This multisensory input is particularly valuable in improving driver behavior studies, training applications, and overall simulation realism.

Fig. 6.1 The complete simulation setup featuring the Atomic A3 racing seat, highlighted Fanatec equipment including the steering wheel, pedals, and shifter, providing an immersive driving experience.

Ghafarian et al. [33] highlight that *"dynamic vehicular motion simulators are essential tools for evaluating driving behavior, vehicle dynamics, and human-machine interactions, as they provide controlled and repeatable conditions while mimicking real-world motion cues"*. The ability to generate precise motion responses is especially beneficial in scenarios requiring high situational awareness, such as urban driving, where road irregularities, sudden maneuvers, and varying speeds affect vehicle handling. By integrating a motion seat, the simulator does not merely present a visual scene but also engages the driver's proprioceptive senses, reinforcing the realism of the experience.

For this project, the Atomic A3 motion seat was employed, managed via the Actuate GUI plugin. This tool allows users to select from a variety of pre-configured motion profiles or develop custom ones tailored to specific simulation needs. A custom configuration was created to better adapt the motion response to the driving conditions simulated, ensuring a more coherent and immersive experience. Through this integration, the simulation delivers a more physically engaging environment, bridging the gap between virtual representation and real-world driving dynamics.

6.3 Experimental Additions

As part of the effort to enhance immersion within the simulation, two additional features were explored: **Hand Tracking** and **Motion Compensation**. While both were initially considered promising, practical limitations led to their eventual exclusion from the final setup.

Hand Tracking was introduced as a means to increase realism by allowing users to interact with the virtual environment using their real hands. The potential benefits of this approach included a greater sense of presence, more intuitive interactions, and a reduction in the need for physical controllers. However, its implementation proved problematic due to several limitations related to both hardware and software constraints.

One major issue was the necessity of aligning the physical and virtual driving components. Specifically, in order to use Hand Tracking effectively, the positions of the virtual steering wheel and gear shifter had to match their real-world counterparts. This requirement introduced two significant challenges. The first issue stemmed from the racing-style seating position of the simulator, which features a relatively high steering wheel and a reclined posture. Aligning the virtual steering wheel with the physical one resulted in an elevated windshield position in the virtual scene, which significantly restricted the user's field of view.

The second problem was related to the gear shifter. In the physical setup, the shifter is positioned low and at nearly the same depth as the steering wheel, making it quite different from traditional vehicle layouts. To accommodate this layout in the virtual space, a highly unconventional car configuration would have been required. During its implementation, interaction with the UI menu was designed to

be controlled through direct hand gestures. However, the lack of haptic feedback in this context posed a challenge, as users could not receive tactile confirmation of their gestures, potentially reducing usability. Studies have also highlighted the difficulties of hand tracking in scenarios where the hands are outside the optimal tracking range, such as when they are positioned below the field of view of the headset's cameras.

Moreover, the issue of *"uncanny valley"* effects (Gavin Buckingham, 2024 [34]) arose when the virtual hands did not perfectly synchronize with the user's real movements, potentially leading to discomfort. Given the precision required in a driving simulation, even slight discrepancies in hand position and responsiveness could negatively impact usability.

Due to these constraints, Hand Tracking was ultimately removed from the setup.

Motion Compensation was also tested as a potential enhancement but encountered technical difficulties when used in conjunction with Hand Tracking. The primary issue stemmed from API conflicts: OpenXR relies on Oculus APIs for Hand Tracking, whereas SteamVR requires its own APIs to implement Motion Compensation. This incompatibility made it impossible to run both features simultaneously. After Hand Tracking was removed, Motion Compensation was tested separately using a Vive Tracker attached to the motion seat. However, the results were not as expected. Users reported that enabling Motion Compensation introduced continuous vibrations in the virtual vehicle, which significantly detracted from the experience. As a result, participants ultimately preferred to disable this feature.

This exploration of additional features highlights the challenges of integrating new immersive technologies into a VR driving simulation. While hand tracking and motion compensation showed potential, their practical limitations ultimately led to their exclusion. These findings emphasize the importance of balancing technological advancements with usability and realism to enhance the overall simulation experience.

Chapter 7

Testing the Generated Environments

To evaluate the maps generated using the workflows discussed in this study, a series of comprehensive tests were carried out involving 20 participants. These tests aimed to assess various aspects of the simulation experience, including the effectiveness of different navigation interfaces and the impact of environmental representation on both immersion and simulator sickness.

Given the focus on integrating large-scale, highly detailed virtual environments into a driving simulator, the tests were designed to evaluate how well these maps perform in a VR context and how they affect user experience, particularly in terms of navigation ease, comfort, and overall realism. By combining objective performance metrics with subjective user feedback, the study sought to gain valuable insights into the strengths and limitations of the generated maps, providing a more nuanced understanding of their viability for real-world applications in autonomous driving simulations and virtual reality experiences.

7.1 Test Structure and Methodology

Each participant began by completing a demographic form and a pre-simulation **Simulator Sickness Questionnaire** (SSQ) to establish their baseline condition before exposure to the virtual environment.

The first simulation took place in the City Sample environment, where participants were asked to drive from point A to point B while following a navigation system. Two different types of interfaces were tested: a **Head-Down Display** (HDD), which resembled a traditional GPS navigation screen, and a **Head-Up Display** (HUD), where a projected line was overlaid directly onto the scene to indicate the correct route. To ensure balanced exposure, participants were divided into two groups: one began with the HUD interface and switched to HDD in the second simulation, while the other followed the opposite order.

Fig. 7.1 The HDD navigation system showing the route and surroundings without any active hazard warnings.

Fig. 7.2 The HDD interface marks a detected hazard with a triangular warning symbol on the navigation display.

Fig. 7.3 The HUD navigation interface displaying standard information without any detected hazards.

Fig. 7.4 The HUD interface highlights a detected hazard using a red bounding box, alerting the driver in real time.

Beyond navigation, the study also investigated how drivers reacted to hazard warnings integrated into each interface. In the HDD system, warnings appeared as a triangle symbol on the navigation screen when approaching an obstacle, such as a vehicle or a pedestrian. The HUD system, on the other hand, marked the obstacle directly in the environment using a red bounding box, making hazards more visually prominent. During their route, participants encountered two predefined dangers: an erratic car and a pedestrian crossing against the traffic light.

Upon completing the first simulation, participants filled out three questionnaires. The post-simulation SSQ measured any changes in simulator sickness symptoms, the **User Experience Questionnaire** (UEQ) assessed the usability and effectiveness of the navigation system they had just tested, and the **NASA-TLX** questionnaire provided insights into the perceived workload and cognitive effort required during the task.

The second simulation followed the same structure but inverted the navigation interface according to the participant's assigned group. Additionally, the route was altered to introduce a different driving scenario, ensuring that any observations were not tied to a single road layout. At the end of this session, participants were again asked to complete the SSQ, UEQ, and NASA-TLX, along with an **Igroup**

Presence Questionnaire (IPQ) to evaluate their sense of presence and immersion in the simulation.

Following these structured driving tasks, participants engaged in a third and final session within the point cloud-based map of Mappano. Unlike the previous tests, this session had no predefined objectives or navigation instructions; instead, participants were encouraged to freely explore the environment, allowing them to form impressions of the visual style and spatial coherence.

At the conclusion of this last session, they completed a final round of questionnaires, including another post-simulation SSQ, an IPQ specifically focused on the Mappano environment, and an additional survey designed to assess their sense of familiarity with the area and any discomfort caused by the point cloud representation.

Overall, each testing session lasted approximately 45 minutes, providing a comprehensive evaluation of navigation systems, environmental perception, and user experience across different virtual driving scenarios.

7.2 Results and Analysis

The results of the tests reveal several interesting trends regarding user adaptation, motion sickness, and environmental perception. Overall, most participants found the second simulation less discomforting than the first, suggesting that their brains were gradually adapting to the experience. This trend is further supported by the fact that many testers started with a certain level of discomfort, as indicated by the SSQ-Pre questionnaire, but concluded the sessions with lower SSQ scores. This could suggest the presence of mild pre-test anxiety or even a certain degree of engagement and appreciation for the simulation itself.

Despite this adaptation, some participants reported a decrease in perceived smoothness when navigating curves, which were the primary moments where motion sickness symptoms emerged. Additionally, some movements of the driving station, such as braking, were found to be overly pronounced compared to real-world driving dynamics. Nevertheless, the majority of testers considered the motion platform a valuable enhancement to the overall experience, despite these occasional inconsistencies.

Regarding the evaluation of the virtual environments, the City Sample was generally perceived as highly realistic. The sense of "Being There" received an average rating of 5.7 on a scale from 1 to 7, with relatively consistent responses across participants. However, when assessing the perceived consistency of the simulation with the virtual environment, the issues mentioned earlier played a significant role, resulting in a lower average score of 4.2. These ratings varied more significantly among testers, as their individual approaches influenced the experience. It is worth noting that, although the vehicle's speed was limited during the tests, some participants instinctively reduced their speed even further out of concern for potential discomfort, while others approached the simulation with a more playful mindset, leading to lower levels of motion sickness. This observation suggests that, beyond individual predisposition, the mental approach to the test itself may influence the perception of discomfort—where the fear of an experience can amplify its effects.

When shifting the focus to the Mappano point cloud map, the average "Being There" score was 5.1 out of 7. Despite the lower level of detail and realism in comparison to the City Sample, many testers still found this environment immersive, in some cases even more so than the highly detailed urban setting. One possible explanation for this lies in the psychological adjustment of expectations, participants seemed to lower their demands for realism when engaging with the point cloud, which, in turn, enhanced their enjoyment and sense of presence within the simulation. Furthermore, despite the unconventional aesthetic of the point cloud environment, testers reported minimal discomfort, with an average score of 2.4. Interestingly, the environment was also perceived as familiar, with several participants recognizing its distinctively Italian character.

Chapter 8

Conclusions and Future Work

The results obtained from user tests have confirmed the strong potential of procedural techniques in generating high-definition virtual environments for immersive experiences. The City Sample, in particular, demonstrated an impressive level of immersion, showcasing how procedural methods can bring virtual urban landscapes to life. Despite this success, the complexity inherent in these environments introduces challenges, particularly regarding performance optimization to ensure smoother interactions in virtual reality. Frame rate stability, although generally good, remains an important aspect for future improvements. By focusing on enhancing rendering efficiency and refining asset management, the usability of these virtual environments within CARLA can be significantly improved.

Another promising avenue for further work involves the integration of advanced Computer Vision models, such as SPIN, to automate the process of OpenDRIVE data generation. This automation could streamline the conversion of procedurally generated maps into CARLA, bypassing some of the limitations of the current manual workflows. Specifically, while RoadRunner's conversion of OpenStreetMap data into road networks is functional, there is still a noticeable lack of precision, necessitating manual intervention to correct inaccuracies. Incorporating AI-driven techniques for automated error correction could lead to a more efficient, scalable workflow for road network generation.

While progress has been made in various areas, the generation of Digital Shadows remains a particularly challenging aspect of the research. Initial tests have shown potential, but the lack of sufficient detail and realism in point cloud-based represen-

tations limits their applicability for driving simulations that require high levels of validity and realism. The absence of an efficient, automated workflow for Digital Shadows across urban environments is a notable barrier. Moreover, the quality of LiDAR scan data, often requiring extensive manual cleaning, further complicates this task. Future efforts should explore leveraging Computer Vision models for point cloud segmentation to automate part of the cleaning process, improving the feasibility of realistic Digital Shadow implementation.

Additionally, the user experience tests provided valuable insights into the importance of seamless interaction between physical and virtual elements. While the removal of hand tracking due to hardware limitations is a notable challenge, it highlights the need for a more integrated approach between real-world ergonomics and virtual representation. Similarly, motion compensation, despite its theoretical promise, was hindered by API conflicts and did not yield the desired improvements in immersion. Future work should focus on refining hardware-software integration and exploring alternative tracking methodologies to better align physical movements with virtual representations, thus enhancing user experience and immersion.

In summary, while this research successfully demonstrates the feasibility of creating immersive virtual environments using procedural techniques and integrating real-world road data into CARLA, there is still significant room for improvement. Future work should focus on addressing performance optimization, increasing the level of automation in data conversion, and developing more effective workflows for Digital Shadows. These efforts will contribute to the creation of more realistic, efficient, and scalable driving simulations, paving the way for future advancements in the field.

Bibliography

[1] Jonathan A Stevens and J Peter Kincaid. The relationship between presence and performance in virtual simulation training. *Open Journal of Modelling and Simulation*, 3(2):41–48, 2015.

[2] Gartner. Digital Twin. In Gartner IT Glossary. Retrieved October 10, 2024, from https://www.gartner.com/en/information-technology/glossary/digital-twin.

[3] Gartner. 5 Trends Drive the Gartner Hype Cycle for Emerging Technologies, 2020. Retrieved October 13, 2024, from https://www.gartner.com/smarterwithgartner/5-trends-drive-the-gartner-hype-cycle-for-emerging-technologies-2020.

[4] F. Tao, B. Xiao, Q. Qi, J. Cheng & P. Ji. Digital twin modeling. *Journal of Manufacturing Systems*, 64:372–389, 2022.

[5] L. Wright & S. Davidson. How to tell the difference beetween a model and a digital twin. *Advanced Modeling and Simulation in Engineering Sciences*, 7:1–13, 2020.

[6] F. Tao, W. Liu, M. Zhang, T. L. Hu, Q. Qi, H. Zhang, ... & X. Jin. Five-dimension digital twin model and its ten applications. *Comput. Integr. Manuf. Syst*, 25(1):1–18, 2019.

[7] F. Tao, H. Zhang, Q. Qi, J. Xu, Z. Sun, T. Hu, ... & C. Chen. Theorym of digital twin modeling and its application. *Comput. Integr. Manuf. Syst*, 27(1):1–15, 2021.

[8] Gartner. Big Data. In Gartner IT Glossary. Retrieved October 27, 2024, from https://www.gartner.com/en/information-technology/glossary/big-data.

[9] S. Ivanov, K. Nikolskaya, G. Radchenko, L. Sokolinsky & M. Zymbler. Digital Twin of City: Concept Overviw. *Global Smart Industry Conference (GloSIC)*, pages 178–186, 2020.

[10] V. V. Lehtola, M. koeva, S. O. Elberink, P. Raposo, J. P. Virtanen, F. Vahdatikhaki & S. Borsci. Digital twin of a city: Review of technoloy serving city needs. *International Journal of Applied Earth Observation and Geoinformation*, 114:102915, 2022.

[11] M. Batty. Digital twins. *Environment and Planning B: Urban Analytics and City Science*, 45(5):817–820, 2018.

[12] Ekaterina Povkh. 10 digital twins of cities, 2020. Retrieved October 27, 2024, from https://realty.rbc.ru/news/5e297b079a79478024d54ff6.

[13] Geospatial World. Virtual singapore - building a 3d-empowered smart nation. Retrieved from https://geospatialworld.net/prime/case-study/national-mapping/virtual-singapore-building-a-3d-empowered-smart-nation/.

[14] E. Shahat, C. T. Hyun & C. Yeom . City digital twin potentials: A review and research agenda. *Sustainability*, 13(6):3386, 2021.

[15] Wingtra. Wingtra creates stunning digital twin of zurich. Retrieved from https://wingtra.com/case_studies/wingtra-creates-stunning-digital-twin-of-zurich/.

[16] W. Kritzinger, M. Karner, G. Traar, J. Henjes & W. Sihn. Digital Twin in manufacturing: A categorical literature review and classification. *Ifac-PapersOnline*, 51(11):1016–1022, 2018.

[17] Andras Kemeny and Francesco Panerai. Evaluating perception in driving simulation experiments. *Trends in cognitive sciences*, 7(1):31–37, 2003.

[18] Alexey Dosovitskiy, German Ros, Felipe Codevilla, Antonio Lopez, and Vladlen Koltun. CARLA: An open urban driving simulator, 2017.

[19] Salvador Bayarri, Marcos Fernandez, and Mariano Perez. Virtual reality for driving simulation. *Communications of the ACM*, 39(5):72–76, 1996.

[20] Sandro Ropelato, Fabio Zünd, Stéphane Magnenat, Marino Menozzi, and Y van Dinther. Adaptive tutoring on a virtual reality driving simulator. *ETH Zurich-International SERIES on Information Systems and Management in Creative EMedia*, 2017(2):12–17, 2018.

[21] Andras Kemeny. From driving simulation to virtual reality, 2014.

[22] CARLA Simulator. Carla unreal engine 4 documentation. https://carla.readthedocs.io/en/latest/.

[23] R. Wang, J. Peethambaran & D. Chen. LiDAR Point Clouds to 3-D Urban Models: A Review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 11(2):606–627, 2018.

[24] D. Griffiths & J. Boehm. SynthCity: A large scale synthetic point cloud, 2019.

[25] CloudCompare. Houghnormals (plugin). https://www.cloudcompare.org/doc/wiki/index.php/HoughNormals_(plugin).

[26] Epic Games. City sample. https://dev.epicgames.com/documentation/en-us/unreal-engine/city-sample-project-unreal-engine-demonstration#collaborativelyconstructinglargeworlds.

[27] Epic Games. Introducing the matrix awakens: An unreal engine 5 experience, 2021. https://www.unrealengine.com/en-US/blog/introducing-the-matrix-awakens-an-unreal-engine-5-experience.

[28] Epic Games. City sample quick start - generating a city and freeway using houdini. https://dev.epicgames.com/documentation/en-us/unreal-engine/city-sample-quick-start-for-generating-a-city-and-freeway-using-houdini%3Fapplication_version%3D5.0.

[29] Epic Games. City sample quick start - generating a city and freeway in unreal engine 5. https://dev.epicgames.com/documentation/en-us/unreal-engine/city-sample-quick-start-for-generating-a-city-and-freeway-in-unreal-engine-5.

[30] Wele Gedara Chaminda Bandara, Jeya Maria Jose Valanarasu, and Vishal M Patel. Spin road mapper: Extracting roads from aerial images via spatial and interaction space graph reasoning for autonomous driving. *arXiv preprint arXiv:2109.07701*, 2021.

[31] A. Liu and S. Chang. Force feedback in a stationary driving simulator. *1995 IEEE International Conference on Systems, Man and Cybernetics. Intelligent Systems for the 21st Century*, 2:1711–1716, 1995.

[32] Diomidis I. Katzourakis, David A. Abbink, Riender Happee, and Edward Holweg. Steering force feedback for human–machine-interface automotive experiments. *IEEE Transactions on Instrumentation and Measurement*, 60(1):32–43, 2011.

[33] Mohammadali Ghafarian, Matthew Watson, Navid Mohajer, Darius Nahavandi, Parham Mohsenzadeh Kebria, and Shady Mohamed. A review of dynamic vehicular motion simulators: Systems and algorithms. *IEEE Access*, 11:36331–36348, 2023.

[34] Gavin Buckingham. Hand tracking for immersive virtual reality: opportunities and challenges. *Frontiers in Virtual Reality*, 2:728461, 2021.