UNIVERSITY OF TARTU
Institute of Computer Science
Data Science Curriculum

Allan Mitt

Creation of Digital Twin for Tartu

Master's Thesis (15 ECTS)

Supervisor: Tambet Matiisen, MSc

Tartu 2024

Creation of Digital Twin for Tartu

Abstract:

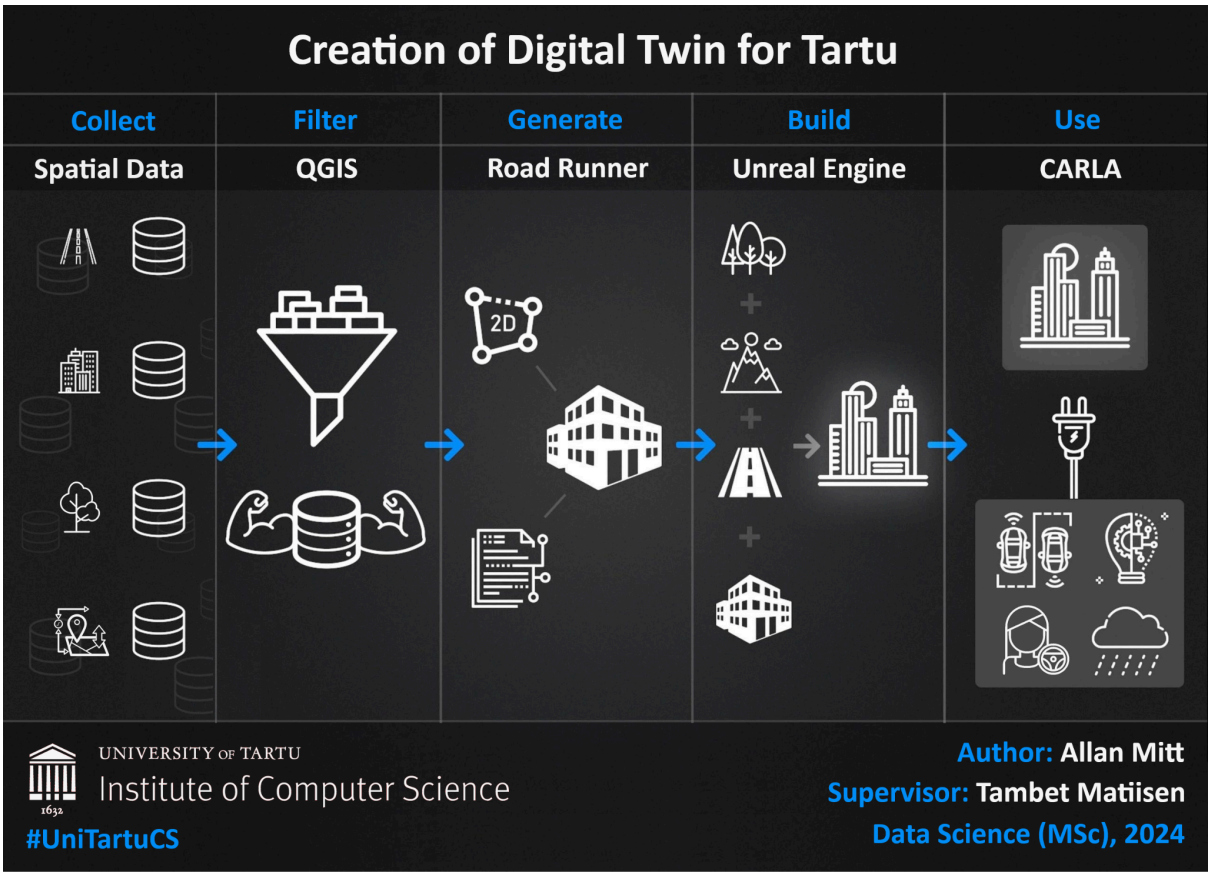
Before testing an autonomous vehicle in real-world conditions, it's more effective to conduct trials in a simulation that closely replicates the real world. This thesis outlines the creation of Tartu City's digital twin, utilising public spatial data and implementing various automation techniques to streamline the process. The final product is compatible with autonomous driving software and is very recognisable to residents of Tartu.

Keywords:

Digital Twin, Simulation, CARLA Simulator, Autonomous Vehicle, Road Runner, Unreal Engine, Blender, Photogrammetry, Agisoft Metashape, High Definition map, 3D Geospatial Data, Topographic Data, QGIS

CERCS: P170 Computer Science, P510 Cartography

Visual Abstract:



Tartu linna digikaksiku loomine

Lühikokkuvõte:

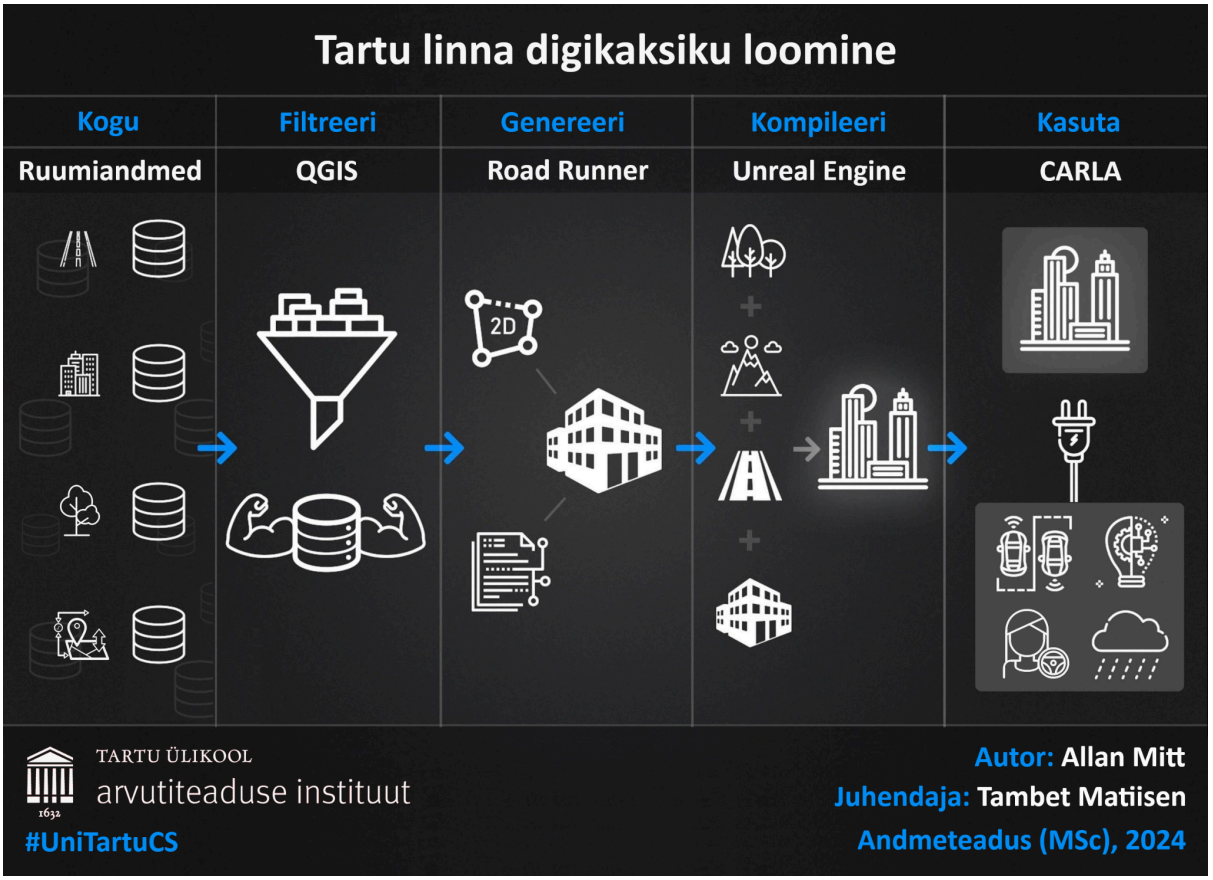
Enne isejuhtiva sõiduki testimist reaalses maailmas on tõhusam katsetada seda simulatsiooni keskkonnas, mis imiteerib pärismaailma. See lõputöö kirjeldab Tartu linna digikaksiku loomist, kasutades avalikke ruumiandmeid ja rakendades erinevaid automatiseerimistehnikaid tööprotsessi kiirendamiseks. Lõpptulemus on ühilduv isejuhtivate sõdukite tarkvaraga ning tartlastele väga äratuntav.

Võtmesõnad:

Digikaksik, Simulatsioon, CARLA Simulaator, Isejuhtiv sõiduk, Road Runner, Unreal Engine, Blender, Fotogrammeetria, Agisoft Metashape, Kõrglahutusega kaart, Ruumiandmed, Topograafilised andmed, QGIS

CERCS: P170 Arvutiteadus, P510 Kartograafia

Visuaalne kokkuvõte:



Tale of Contents

1. Introduction

The emergence of self-driving vehicle technology marks a significant change in city transportation, offering the potential to reshape how we move by incorporating sophisticated computer algorithms, artificial intelligence, and machine learning. This expanding field aims to enhance road safety by reducing human error and tries to streamline traffic management, reduce carbon emissions, and revolutionise the passenger experience. Central to these advancements is the development of digital twins—virtual replicas of urban environments that serve as dynamic testing grounds for autonomous vehicle systems. These sophisticated simulations enable researchers and developers to precisely analyse and refine the algorithms that govern autonomous navigation, interaction with traffic infrastructure, and response to diverse weather conditions.

This thesis describes the design process behind creating a digital twin of Tartu City within the CARLA[1] simulator, where autonomous driving technologies can be used to evaluate the test vehicle's performance in a virtual environment. This simulation environment facilitates testing autonomous driving systems under various conditions, replicating real-world scenarios. By accurately rendering the urban landscape of Tartu City, including its roads, traffic infrastructure, and natural elements, the digital twin allows for the exploration of vehicle behaviour in response to diverse environmental and traffic conditions. This level of detail ensures that autonomous vehicles can be tested against the many challenges they would encounter in actual operation, from navigating complex urban intersections to adjusting to weather and lighting conditions.

The thesis is structured into nine chapters and explores the intersection between autonomous vehicle technology and digital twin simulations. Chapter two, Background, provides a more in-depth look into autonomous vehicles, simulations, and the digital twin concept, providing contextual understanding. Chapter three, Software Used, describes the selection and application of specific software tools employed in developing the Tartu City digital twin, explaining the rationale behind their choice. Chapter four, Input Data, discusses the various data sources utilised in constructing the digital twin, highlighting the objectives and challenges associated with data integration. In chapter five, Methodology, the actual design process is described in detail, from initial planning to execution. Chapter six, Results and Discussion, presents the outcomes of the simulation tests conducted within the digital twin environment, offering insights into the performance and behaviour of autonomous driving systems. Chapter seven, Conclusion, reflects on the key achievements of the current project. Chapter eight, Future Work, proposes future research and development directions.

[1] https://carla.org/

2. Background

The concept of self-driving cars has rapidly transitioned from science fiction to a tangible, although complex, engineering challenge. Advances in artificial intelligence (AI), sensor technologies, and computational power have fueled innovation, but alongside these exciting developments come vital questions of safety, efficiency, and the complicated role of simulation in developing and deploying autonomous vehicles (AVs).

Historically, traffic safety concerns have focused on the frailty of human drivers. Research indicates that human error plays a critical role in most crashes [1]. While autonomous vehicles promise to address this issue, they introduce new risks inherent in any complex, AI-powered system. Expecting perfect behaviour from any technical system is unrealistic, and critics note that AVs must still navigate unpredictable human actions and environmental factors. The focus, therefore, shifts to whether AV systems can ultimately surpass human drivers in safety and how we can carefully validate such claims.

Potential advantages of AVs extend beyond mere accident reduction. A self-driving fleet could enhance transportation efficiency through optimised traffic flow, increased fuel economy, and expanded mobility options for those unable to drive themselves. As Fagnant & Kockelman [2] point out, autonomous vehicles have the potential to improve traffic safety, increase transportation accessibility for non-drivers, and even reshape the built environment. These societal implications demand careful consideration as researchers and policymakers grapple with the ethical responsibilities surrounding this potentially transformative technology.

The training of autonomous systems is where safety, technology, and data critically intersect. Unlike traditional software, an AV's ability to "drive" relies on complex AI models trained on massive datasets. This data-centric approach raises questions about the quality, representativeness, and accessibility of that data. Open data plays a crucial role here. In essence, open data is freely accessible, usable, and modifiable by anyone. In the context of AVs, this includes datasets containing sensor readings (such as LiDAR and camera data), GPS information, detailed traffic patterns, infrastructure (road networks, signage, etc.), weather conditions, and pedestrian behaviours [3]. Access to large, diverse, and high-quality open datasets is essential for training AI models capable of operating reliably. It expands the world and the scope of testing scenarios beyond what a single organisation could collect independently.

Simulation has become indispensable throughout AV development, as outlined in the survey "How Simulation Helps Autonomous Driving..." [3]. Simulators can rapidly generate diverse scenarios in a cost-effective and safe environment, from the predictable to the chaotic. Moving beyond early simulators focused solely on traffic flow and vehicle dynamics, modern simulators enable the "virtual training" of AI models, exposing them to experiences impossible to replicate safely or affordably in the physical world.

Notably, the survey article "Choose Your Simulator Wisely..." [4] underscores the range of specialised simulators available to developers. This diversity of tools is essential for subsystem testing. Simulators focused on sensor data generation allow for refining perception algorithms in varied conditions, while dedicated driving policy simulators test an AV's core decision-making logic. Open-source simulators such as CARLA further widen access, fostering a collaborative development environment and potentially accelerating AV technology.

At the same time, the inherent limitations of simulation pose a fundamental obstacle to developing reliable self-driving vehicles. Algorithms trained purely in idealised virtual environments can falter when exposed to the dynamic complexities of the real world – unexpected road conditions, unpredictable behaviours of other drivers, and subtle variations in sensor readings [3]. This discrepancy, known as the Sim2Real gap, necessitates strategies to enhance simulation fidelity and bridge the divide between controlled environments and reality.

Sim2Real encompasses various techniques designed to make simulations more realistic and robust. These techniques include:

- Domain Randomization: Introducing deliberate variations within the simulation, such as changes in lighting, weather conditions, object appearances, and even physics parameters. This forces algorithms to adapt to diverse scenarios, reducing their brittleness in real-world situations.
- Data Augmentation: Expanding the training dataset by manipulating existing data. Techniques like image rotation, noise injection, or synthetically varying sensor data generate new scenarios for algorithm training, increasing exposure to potential real-world conditions.
- Sensor Modelling: Faithfully replicating the imperfections of real-world sensors (camera lens distortion, LiDAR noise patterns) within the simulated environment. This ensures that algorithms learn to handle the inevitable limitations of physical sensors.

Parallel Intelligence (PI) provides a framework for harnessing the strengths of Sim2Real-enhanced simulations and strategically incorporating real-world data for continuous learning. PI emphasises an iterative process:

- Simulation to Real: Algorithms developed and refined in simulation (bolstered by Sim2Real techniques) are cautiously deployed onto real-world vehicles under controlled conditions.
- Real to Simulation: Data collected from vehicles navigating the physical world, reflecting real traffic patterns, pedestrian behaviour, and diverse sensor responses, is fed back to refine simulations. This feedback loop is critical to closing the Sim2Real gap as simulations become progressively more aligned with reality.

The power of this combined approach lies in its continuous improvement cycle. Sim2Real makes simulations a more effective starting point for algorithm training. Parallel Intelligence organises a repeating process, with real-world experiences fine-tuning simulations that, in turn, lead to the development of more robust algorithms. This iterative process promises to "promote a continuous improvement cycle" within autonomous vehicle development [3].

Testing all these scenarios in a simulator requires an environment to play them out, bringing the concept of a digital twin to the forefront. A digital twin is essentially a virtual model that mirrors a real-world environment, encompassing detailed replicas of roads, buildings, vegetation, and traffic infrastructure and ensuring that every element within the simulation precisely corresponds with its real-world counterpart in terms of location and physical properties. These models, often generated from open data sources, allow for extensive virtual testing within highly realistic environments.

The accuracy of the digital twin extends to the GPS coordinates used within the simulation, which match exactly with those of the actual geographic location. This level of precision ensures that simulated scenarios are as close to real-life conditions as possible, providing a

robust platform for comprehensive testing. The simulation environment also incorporates realistic traffic rules and dynamic variables such as the number of road users, weather conditions, and time of day. These variables can be manipulated to create scenarios, from everyday traffic conditions to rare or extreme situations.

By granting testers the ability to control various aspects of the road environment, including traffic density, traffic light behaviour, and other critical factors, the digital twin becomes an invaluable tool for simulating the behaviour of autonomous vehicles under diverse conditions. This controlled environment allows for the detailed analysis of the vehicle's responses to different challenges, enabling the identification and resolution of potential safety issues before the car is ever tested on actual roads.

The digital twin is an important element in the autonomous vehicle development pipeline, serving as a safe, efficient, and cost-effective method to assess vehicle performance and road safety. By employing comprehensive simulation testing, developers can obtain high-quality data on the vehicle's behaviour, substantially improving the development process and promote the creation of safer, more reliable autonomous driving systems before even a single kilometre is driven on the actual road. Although comprehensive virtual testing protocols are indispensable as a safety net before real-world deployment, they cannot fully replace the need for physical validation, emphasising the importance of integrating both virtual and physical testing approaches in the development process.

3. Software Used

The current project was developed by integrating diverse types of input data and specialised software to address different challenges. Consequently, a variety of software environments were employed throughout the project's execution. This chapter provides a detailed overview of the software programs used, outlining their functionalities and the reasons for their selection.

3.1 Blender

Blender[2] is a comprehensive, free, and open-source 3D creation suite that facilitates a wide range of digital content creation, including animation, modelling, simulation, rendering, compositing, and motion tracking. Known for its versatility in supporting the entire 3D pipeline, it is accessible to individual artists and small studios. Its integration of a Python API for scripting allows for extensive customisation and the development of specialised tools. Blender operates efficiently on Linux, Windows, and Macintosh computers as a cross-platform tool, offering a consistent user experience. The software's development is community-driven, under the GNU General Public License (GPL), ensuring it remains free and open to modifications by its user base. This model promotes continuous improvement and software adaptation to new technologies and user requirements. [5][6]

Educationally, learning Blender can offer a solid foundation in digital art and animation, providing transferable skills to numerous other digital content creation tools. Its wide adoption and active community also mean that users can access many tutorials, forums, and collaborative projects, facilitating a learning environment as robust as professional settings.[7]

Blender was chosen as the preferred modelling software for the project, a decision influenced by prior experience with the application. As the most popular free modelling tool available, Blender boasts wide accessibility, compatibility with nearly any machine, and user-friendly operation. Its selection was further justified by its versatility in handling various tasks essential for the project, such as splitting up or combining different objects and converting them between file formats.

3.2 RoadRunner

MathWorks's RoadRunner[3] is a versatile interactive editor designed for creating 3D scenes to simulate and test automated driving systems. It stands out for its user-friendly interface that allows the design of realistic road networks, complete with traffic signals, signs, and various roadway features. This tool is especially favoured for its compatibility with ASAM OpenDRIVE[4], enabling users to export and import detailed road environments for simulation use.[8]

Integrating GIS data enhances RoadRunner's ability to build accurate 3D scenes reflecting real-world locations. Users can import aerial images, elevation data, and lidar point clouds, making it possible to recreate physical locations with high fidelity. The software's comprehensive asset library, which includes customisable signs and models, aids in populating scenes with realistic objects, thereby increasing the immersion and realism of simulations.[8]

---

[2] https://www.blender.org/
[3] https://uk.mathworks.com/products/roadrunner.html
[4] https://www.asam.net/standards/detail/opendrive/

RoadRunner's ability to export scenes to various simulators and gaming engines, such as CARLA, aiSim, and Unreal Engine, underscores its flexibility. This feature ensures that scenes created in RoadRunner can be seamlessly integrated into various development workflows, facilitating broader applications in automated driving systems development. [8]

RoadRunner stands unparalleled in its capability to construct a road network from the ground up. This software is explicitly engineered for such purposes, offering an optimal solution for developing road networks. In the context of this project, RoadRunner was instrumental in designing the road infrastructure, designing land areas, and accurately positioning various structures within Tartu City, demonstrating its efficacy and precision in urban planning and development.

3.3 CARLA Simulator

CARLA[5] (Car Learning to Act) is an advanced open-source simulator designed for autonomous driving research, supporting development, training, and validating autonomous urban driving systems. Developed with a focus on flexibility, scalability, and realism, CARLA offers a comprehensive toolset for simulating complex driving environments and autonomous vehicle sensors, including LIDARs, cameras, and GPS. The simulator is distinguished by its server multi-client architecture, allowing for scalable simulations with multiple actors controlled by different clients, and its extensive API that provides control over various simulation parameters such as traffic, weather, and sensor configurations. [9]

Key features of CARLA include its ability to generate realistic traffic scenarios, integration with ROS (Robot Operating System) for enhanced robotics applications, and the provision of autonomous driving baselines for immediate testing and development. Moreover, CARLA supports the creation of custom maps following the OpenDrive standard, which is crucial for developing specific driving scenarios or testing autonomous vehicles in tailored environments. [9]

The simulator has been utilised in diverse research projects and applications, demonstrating its flexibility and capability. For example, it has been employed to simulate specific traffic situations, including lane switching and overtaking, in a controlled environment with custom maps reflecting specific regional road configurations and signs [10]. This adaptability underscores CARLA's utility in creating highly customised driving scenarios for detailed analysis and development.

CARLA's comprehensive feature set, open-source accessibility, and strong community support make it a powerful autonomous driving research and development platform. Its ability to simulate complex urban environments and integrate with advanced vehicle sensor suites has made it an invaluable resource for researchers, educators, and developers in autonomous driving.

The CARLA Simulator was selected for its distinction as the foremost open-source platform for autonomous driving research, powered by the Unreal game engine and seamlessly integrated with Autoware software. The simulator was used to compare the virtual setup of Tartu City against the real-life counterpart when running autonomous driving scenarios.

[5] https://carla.org/

3.4 Unreal Engine

Unreal Engine[6], developed by Epic Games, is a cutting-edge real-time 3D creation platform that empowers developers and creators across various industries to bring their visions to life. It's known for its robust features that cater to various applications, from game development to architectural visualisations, cinematic productions, and more. With Unreal Engine, users can create immersive and visually stunning experiences across PC, console, mobile, VR, and AR platforms. Moreover, the engine is free to download and use for creating linear content, custom and internal projects. [11]

Unreal Engine's visual scripting system, Blueprints, is particularly noteworthy. It offers a node-based interface that enables artists and designers to prototype and finalise interactive content without writing a single line of code. This changes the creation process, making it accessible to non-programmers while offering the depth and control required for complex projects. [11]

Unreal engine was selected as the project's default game engine because it was required for CARLA Simulator software. It was used to merge all the project's different parts and add some additional features. The result was a digital map of Tartu that was usable in the CARLA Simulator environment.

3.5 QGIS

QGIS[7] is a leading open-source Geographic Information System (GIS) known for its versatility and broad functionality. It enables users to create, edit, visualise, analyse, and publish geospatial information on various platforms, including Windows, macOS, Linux, and Android. The software supports many vector, raster, and database formats, making it a flexible tool for GIS professionals and enthusiasts. Its user-friendly interface and extensive documentation facilitate a relatively easy learning curve for newcomers. QGIS's accessibility is one of its most significant advantages. It is free to use, offering an affordable alternative to expensive proprietary GIS software without sacrificing quality or performance. QGIS stands out for its comprehensive features, strong community support, open-source nature, and commitment to accessibility and inclusivity. [12][13][14]

The software was employed to filter out specific location data of various objects within the city, subsequently converting this information into a coordinate format compatible with the Unreal Engine. This process ensured that the spatial details of city objects were accurately represented and could be seamlessly integrated into the Unreal Engine's development environment for further manipulation and visualisation.

3.6 Agisoft Metashape

Agisoft Metashape[8] stands out as a leading software in photogrammetry, providing a comprehensive suite for processing digital images into 3D spatial data. Known for its high accuracy and speed, it caters to various professional needs across GIS applications, cultural heritage documentation, and visual effects production. The software's capabilities extend from creating detailed 3D models and orthophotos to generating dense point clouds, making it a versatile tool for various scales of objects and environments. [15][16][17]

[6] https://www.unrealengine.com/en-US/
[7] https://qgis.org/en/site/
[8] https://www.agisoft.com/

The workflow in Metashape is designed to be user-friendly, enabling both novices and experienced users to achieve high-quality photogrammetric outputs. It supports a range of processing options, from photo alignment and masking to dense cloud building and mesh generation, providing flexibility in data processing and model creation. [15]

In this project, the technology was utilised to project aerial photographs onto pre-constructed city meshes accurately, enabling the creation of a highly detailed and realistic representation of the urban landscape. This approach enhanced the visual fidelity of the digital model by overlaying real-world imagery onto the 3D structures, enriching the simulation environment with authentic textures and details.

3.7 Python

Python[9] is a highly versatile and widely used programming language known for its simplicity and readability, making it an excellent choice for both beginners and experienced developers. Its broad standard library, support for multiple programming paradigms, and extensive ecosystem of third-party packages enable developers to tackle a wide range of tasks, from simple scripts to complex applications. Python's simplicity allows developers to focus on solving problems rather than syntax intricacies, speeding up the development process.

When combined with Blender and Unreal Engine, Python significantly enhances these platforms' capabilities. It can be used for scripting and automating various aspects of the development and content creation workflows. It allows asset management automation, streamlining processes, and integration with other tools and systems.

[9] https://www.python.org/

4. Input Data

The spatial data required for the project was compiled from multiple sources, including the Estonian Land Board[10], the geoHUB[11] of Tartu City, 3DI[12] and the OpenStreetMap[13]. This comprehensive dataset encompassed elevation data, topographic data, orthophotos, geospatial vector, and 3D geospatial data, all instrumental in the project's execution. For a detailed overview of the data types and their applications within the project, refer to Table 1.

Table 1: Different Input Data

Structure	Amount
Roads	https://www.openstreetmap.org/ (OSM), Estonian Land Board elevation map 2023 (GeoTIFF), Estonian Land Board Estonian topography database road outlines 2023 (ESRI Shape), Estonian Land Board orthophoto 2023 (GeoTIFF), Estonian Land Board point cloud 2023 (LAZ)
Ground	Estonian Land Board elevation map (GeoTIFF), Estonian Land Board orthophoto (GeoTIFF), Estonian Land Board point cloud (LAZ) Estonian Land Board 3D models (OBJ), 3DI 3D models (OBJ), Estonian Land Board Estonian topography database house outlines (ESRI Shape), Estonian Land Board point cloud (LAZ)
Buildings	Estonian Land Board 3D models (OBJ), 3DI 3D models (OBJ), Estonian Land Board Estonian topography database house outlines (ESRI Shape), Estonian Land Board point cloud (LAZ)
Road+ground+building textures	Tartu city drone photos (JPG)
Vegetation	Estonian Land Board tree database (GeoPackage), Estonian Land Board point cloud (LAZ)
Street lights	Tartu City geoHub (CSV)
Bus stops	Tartu City geoHub (CSV)
Trash containers	Tartu City geoHub (CSV)

4.1 The Elevation Data

Estonia's entire territory benefits from high-quality elevation data gathered through Airborne Laser Scanning (ALS). The initial two scanning phases utilised a Leica ALS50-II scanner at an altitude of 2400m, while the third phase (2017-2020) employed a Riegl VQ-1560i scanner at 2600m. Annually, low-altitude ALS data (1200m) is collected for Tallinn, Tartu, and Pärnu areas, with biennial collections for other towns. Additionally, summertime flights cover a quarter of the territory for forestry purposes each year. Figure 1 displays an instance of point

[10] https://geoportaal.maaamet.ee/eng/
[11] https://geohub.tartulv.ee/
[12] https://3di.ee/
[13] https://www.openstreetmap.org/

cloud data visualised within the RoadRunner environment, showcasing the detailed representation of the data in this specific setting. [18]

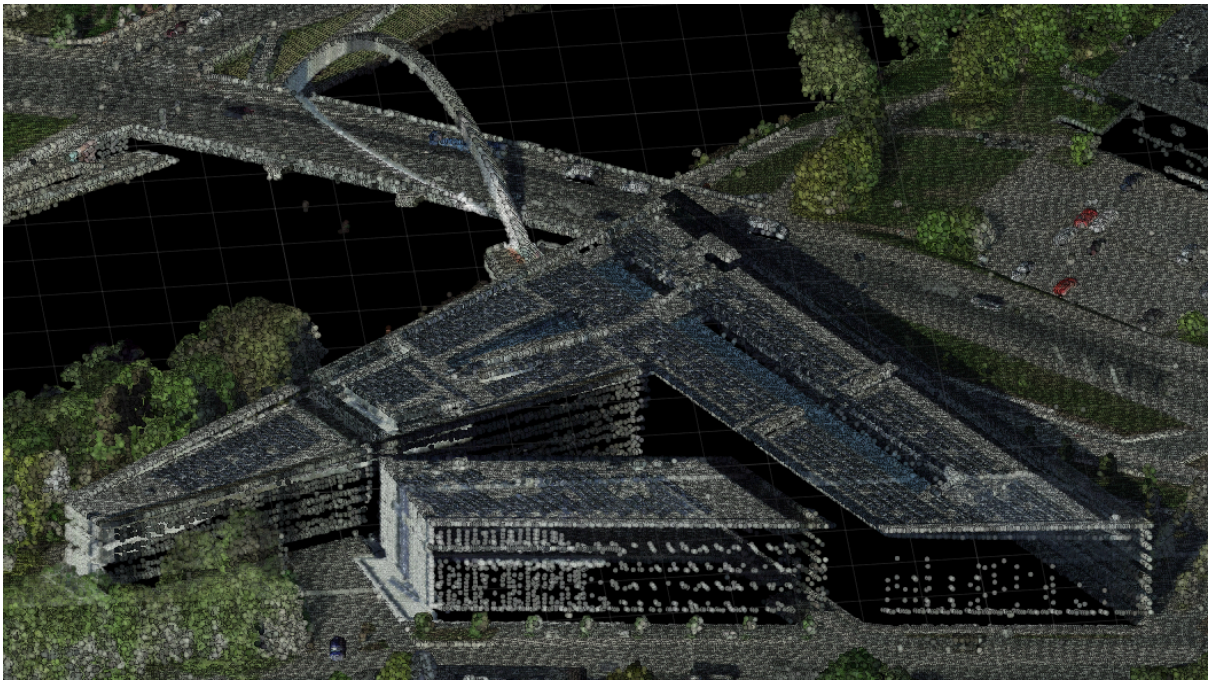


Figure 1: Point cloud structure in LAZ format, originating from the Delta Centre. Each dot within the structure denotes the elevation at its respective point, providing a detailed representation of the terrain's topography.

The raw LiDAR data is stored in LAZ 1.4 format, with Digital Elevation Models (DEMs) available in GeoTIFF and XYZ ASCII formats. These DEMs, which range in grid sizes from 1m and 5m, include Digital Terrain Models (DTM), Digital Surface Models (DSM), normalised DSM (nDSM), and Canopy Height Models (CHM), primarily derived from the latest 2017-2020 ALS data. The DTM utilises data from springtime flights, including low-altitude flights over towns, while DSM/CHM incorporates data from summertime flights. The types of data utilised in the project, and their individual properties are described in Table 2. [18]

Table 2: The Elevation Data

Data type	Sturcure	Scale
Raw Light Detection and Ranging (LiDAR)	LAZ 1.4	18 points/m² — 0,8 points/m²
Digital Terrain Models (DTM) format	GeoTIFF	DEM grid sizes 1m, 5m

LiDAR-derived point clouds were utilised to adjust the altitude of roads and buildings accurately. Digital Terrain Model (DTM) raster maps facilitated the precise calculation of elevation values for the terrain. Comprehensive coverage of the aforementioned data was ensured across the entire project area.

4.2 The Orthographic Data

Orthophotos are aerial photographs that have been processed to remove distortions caused by terrain relief, camera tilt, and camera projection at the time of capture. They feature specific pixel sizes or resolutions, indicating the Ground Sampling Distance (GSD), the smallest area on the ground represented by each pixel.

Nationwide, orthophotos offer coverage at scales between 1:5000 and 1:10,000, with pixel sizes ranging from 20 to 40 cm. The pixel size is finer in densely populated areas, ranging from 10 to 16 cm. Orthophoto maps consist of individual sheets derived from an orthophoto mosaic on a regular grid, with Estonia being covered by 2,074 orthophoto map tiles, each measuring 5 by 5 km. These maps align with the 1:10,000 scale Estonian Basic Map in terms of projection, tiling, and coding, with a GSD of 20-40 cm. Additionally, low-altitude flights achieve a GSD of 10-16 cm and are scheduled annually during springtime over towns such as Tallinn, Tartu, and Pärnu. The images employed in this project qualify under that category and feature a GSD of 10 cm, ensuring a high level of detail in the visual data. [19][20]

In aerial photography available for this project when using drones, the operational altitude significantly influences the GSD. Due to the lower flight altitudes characteristic of drone operations, the resultant GSD is markedly improved, achieving a resolution of approximately 2 cm. This enhanced resolution facilitates the capture of finer details in the imagery, underscoring the advantage of utilising drones for high-resolution aerial photography. An example of this can be seen in Figure 2.

Figure 2: Aerial photo taken by a drone with a DJI M3E camera at 130 meters.

The types of data used in the present project and their respective properties are displayed in Table 3.

Table 3: The Orthographic Data

Data properties		
Format	GeoTIFF	JPG
Data structure	raster	raster
Ground Sampling Distance (GSD)	10 cm	2 cm

4.3 The Topographic Data

Topographic data encompasses a detailed representation of the Earth's surface, including both natural and man-made features. Such data captures a broad spectrum of information, including elevations, vegetation types, bodies of water, built structures, and more. This project utilised comprehensive topographic data covering roads, water bodies, buildings, vegetation, streetlights, bus stops, and trash containers. This data was provided in various formats, including ESRI Shapefiles, GeoPackage (GPKG) files, and Comma Separated Value(s) (CSV) files, catering to a wide range of analytical and visualisation needs. An example of the topographic data utilised in this project is illustrated in Figure 3, while Table 4 elaborates on the specifics of the different file formats employed, showcasing the diversity and richness of the topographic information integrated into the project.

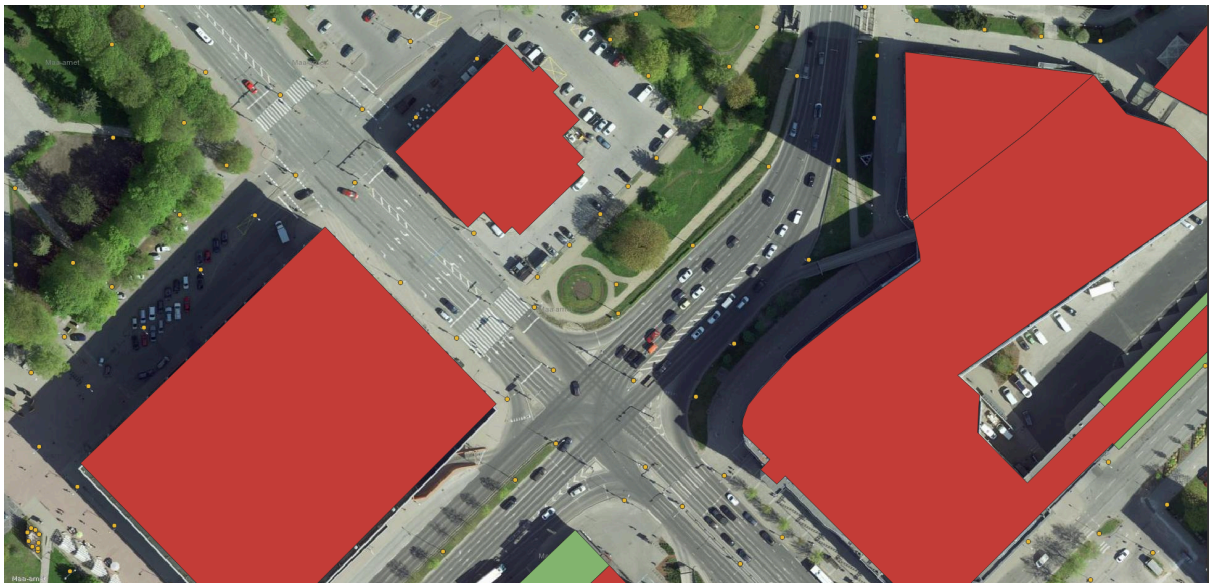


Figure 3: An example of topographic data illustrating buildings and street lights overlaid on an orthophoto map within the QGIS environment.

Table 4: The Topographic Data

Data properties			
Format	ESRI Shapefiles (SHP)	GeoPackage (GPKG)	CSV
Data structure	Vector data	Vector data	Point data
Ground Sampling Distance (GSD)	Roads, water bodies, buildings	Vegetation	streetlights, bus stops, trash containers

4.4 3D Geospatial Data

The construction of three-dimensional (3D) models is derived from Airborne Laser Scanning (ALS) data and building footprints obtained from the Estonian Topographic Database (ETD). The methodology for generating these building models by the Estonian Land Board is entirely automated, avoiding any manual intervention. The models are classified into two distinct datasets based on their level of detail (LOD):

- The LOD 1 dataset consists of building models characterised by uniformly flat roofs, which are situated at the pinnacle of each structure. This simplification facilitates a generalised representation of the buildings' massing without delineating roof details.

- Conversely, the LOD 2 dataset offers a more nuanced and accurate depiction of building roofs, presenting a detailed and realistic portrayal of their surfaces. This dataset aims to provide a higher fidelity representation suitable for applications requiring detailed architectural features.

In addition to the geometric representations, both datasets are augmented with attribute data, which encompasses information such as the building's address, type, and unique identifiers as recognised in various national registries. [21]

This project used three-dimensional (3D) models of buildings in Tartu, specifically those that meet the LOD 2 standard. Chosen for their more realistic roof structures and overall building mass, these models—formatted as OBJ files—play a crucial role in accurately and realistically rendering the urban landscape. Figure 4 presents an example of the exceptional level of detail attained through this automated methodology by the Estonian Land Board, highlighting the models' sophisticated representation.

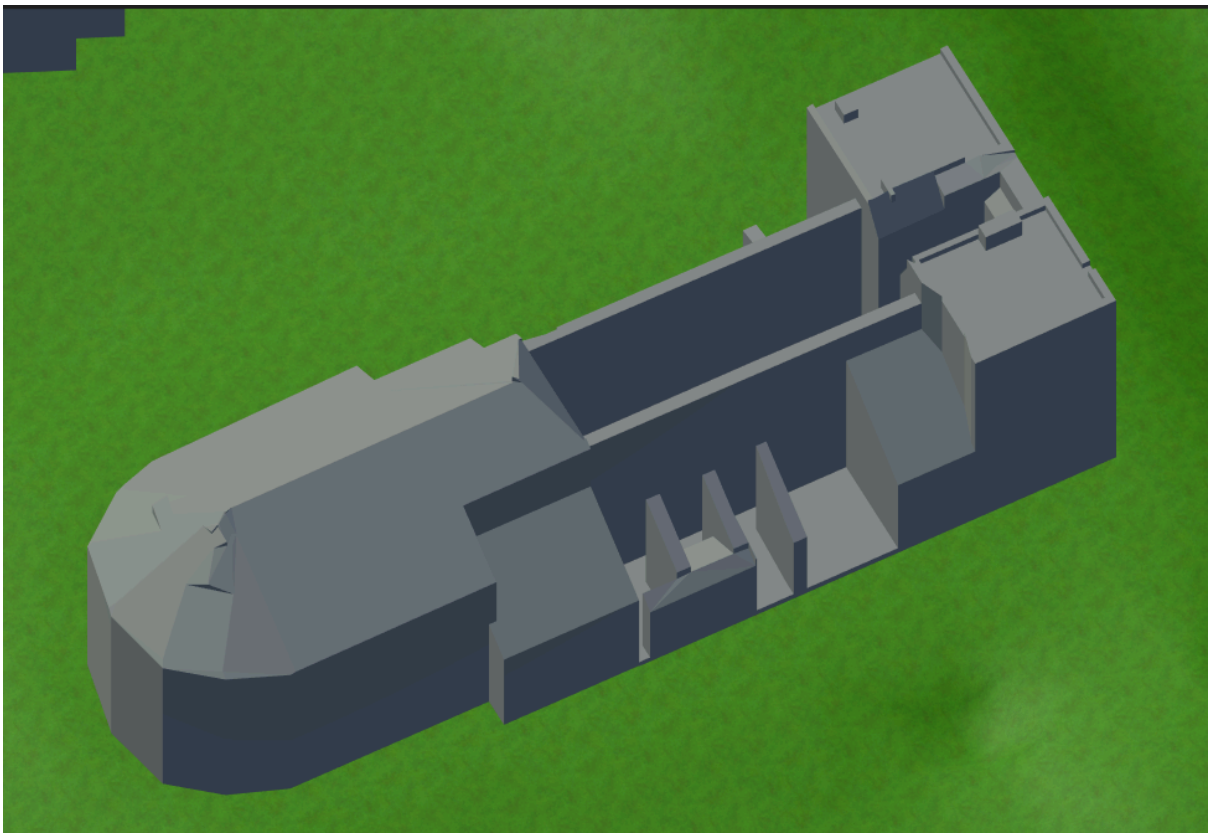


Figure 4: An example of the Ruins of Tartu Cathedral represented at LOD 2.

4.5 Digital twin of Tartu City Centre

The digital twin of Tartu City Centre, utilised as one of this project's primary input data sources, represents a comprehensive digital replica of the urban area, focusing on road network, buildings, vegetation and urban infrastructure. This model was developed during the author's undergraduate study as a project for the bachelor's thesis entitled 'Simulation of Tartu City Centre for Testing Autonomous Vehicles' (2021). Detailed documentation and the development process of the digital twin model are described in the aforementioned work [22]. The model, as mentioned earlier, is illustrated in Figure 5.

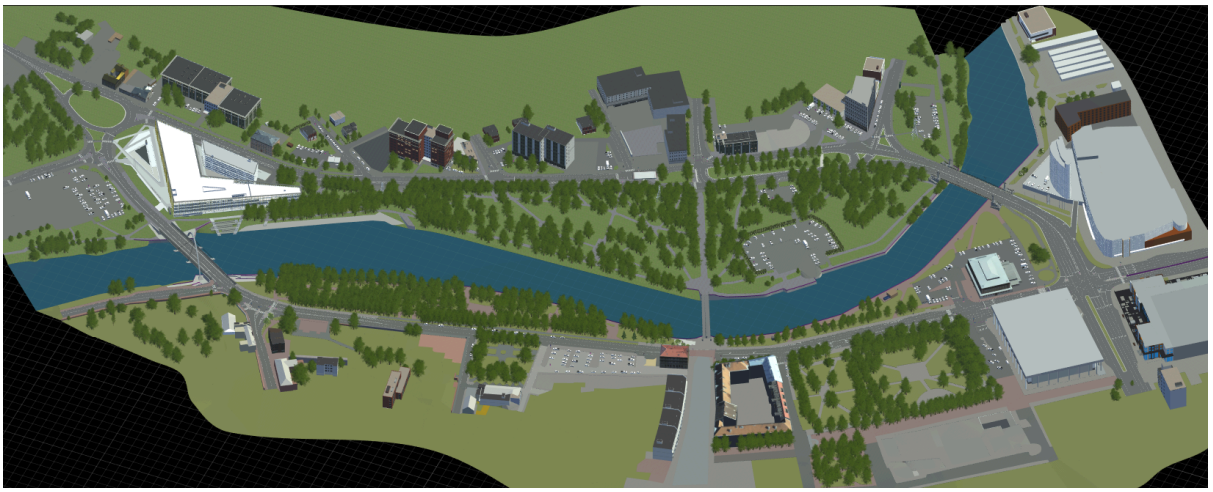


Figure 5: Tartu City Centre model in RoadRunner.

5. Methodology

The methodology chapter outlines the comprehensive process involved in developing the digital twin of Tartu City. This includes a detailed description of the steps and techniques employed throughout the project. The chapter aims to provide a clear framework for how the digital representation of Tartu City was conceived, designed, and executed, emphasising the selection of data sources, software tools, and modelling approaches. It will cover the integration of geospatial data, the creation of 3D models and the application of simulation technologies.

5.1 Advanced Road Network Generation

The road network development can be divided into 4 phases:

1. Creating an open street map (OSM) file that covers the whole of Tartu City.
2. Using the OSM file to generate the road network.
3. Using elevation and data to assign the correct road altitude.
4. Using orthophotos, topographic vector data and point clouds to correct the positioning of roads and their boundaries.

5.1.1 Creating an Open Street Map

The road network construction was initiated by procuring data in OSM format from the OpenStreetMap website[14]. Due to the large area of Tartu, the entire city's dataset could not be queried at once; it had to be divided into pieces (a total of 8 parts). Moreover, the retrieved OSM data contained a considerable amount of extraneous information, or "noise," including details of sidewalks, bike paths, and other elements not directly relevant to the road network. To address this challenge, a Python script[15] *clean_and_merge_osm.py* was deployed to combine the segmented OSM data and filter out the noise. This process culminated in creating a refined OSM file tailored to the project's requirements and free from unnecessary data.

5.1.2 Creating Road Network

The OSM file was imported into the RoadRunner environment, setting the road network's development stage. Initially, nodes within the city centre were removed to avoid duplication, as the road network in this area had already been constructed. Following this preparatory step, the 'build roads' feature of RoadRunner was employed. This critical phase transformed the two-dimensional road node data into a detailed three-dimensional model. This conversion process preserved the intricate layout of Tartu's roads and enhanced it by adding depth and dimensionality, as visually documented in Figure 6. This methodological approach ensured a comprehensive and accurate representation of the city's road infrastructure within the digital twin project.

[14] https://www.openstreetmap.org/
[15] https://github.com/burnout2k/Digital_Twin_of_Tartu

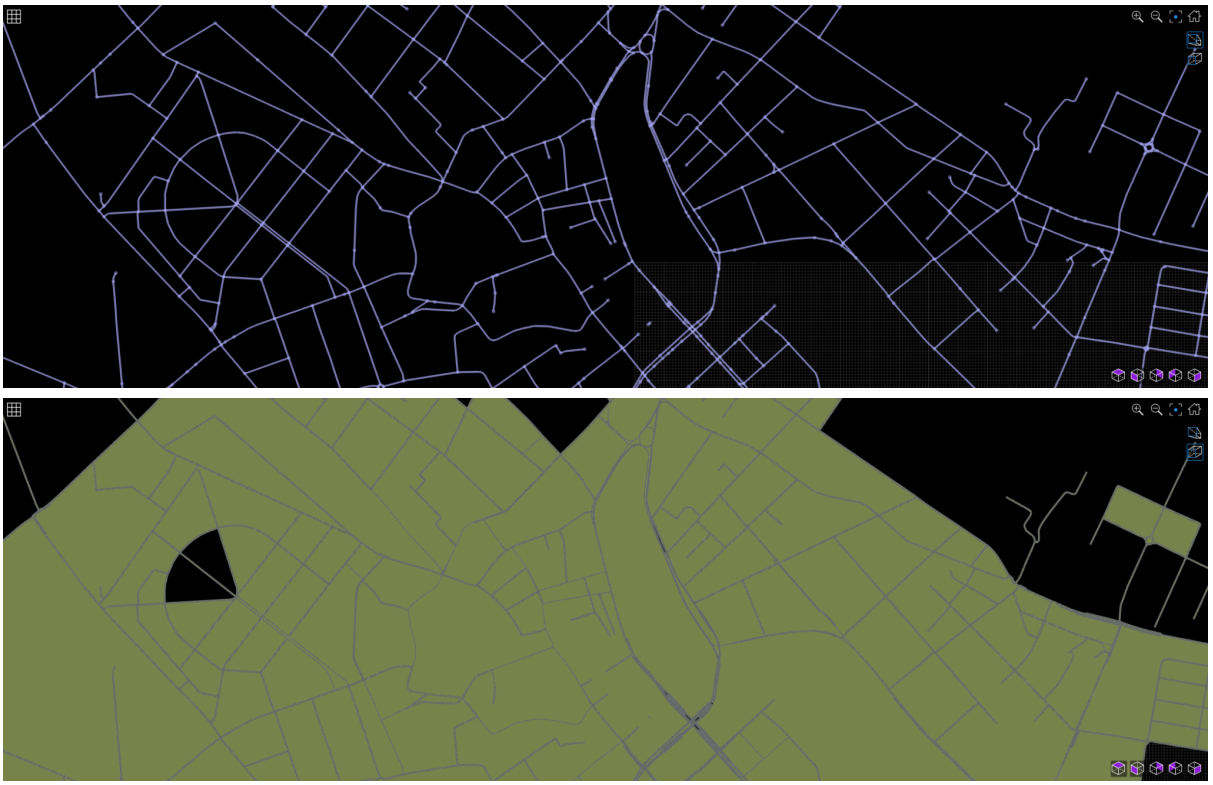


Figure 6: Imported OSM file (top) and automatically generated road system (bottom) in RoadRunner.

5.1.3 Assigning Elevation

The elevation profiles of the road network were automatically generated within RoadRunner using the "Project Roads" feature and utilising digital terrain models (DTM) provided by the Estonian Land Board in GeoTIFF format as the primary input. This method enabled the precise determination of road elevations, ensuring that the digital road network accurately reflected the real-world topography of Tartu. By integrating these elevation maps, RoadRunner could seamlessly incorporate the terrain's natural flow into the road design, enhancing the realism and accuracy of the digital twin.

5.1.4 Refining the Roads

The newly generated road network was seamlessly connected with the city centre counterpart. To ensure the accuracy and integrity of the generated road layouts, a comprehensive verification and correction process was undertaken for the main streets of Tartu within the RoadRunner environment. This process utilised a multi-step approach, incorporating various datasets the Estonian Land Board provided. Specifically, the road outlines were cross-referenced with the topographic database available in ESRI Shape format, providing a detailed outline and structure of the roads. Additionally, orthophotos in GeoTIFF format offered a high-resolution aerial perspective, enabling precise alignment and verification of the road layouts against real-world imagery. Furthermore, point cloud data in LAZ format was utilised to add another layer of accuracy, offering detailed elevation and terrain information. Together, these datasets played a crucial role in refining the road network model, ensuring it accurately reflected the physical layout and topography of the area. The work process is shown in Figure 7.

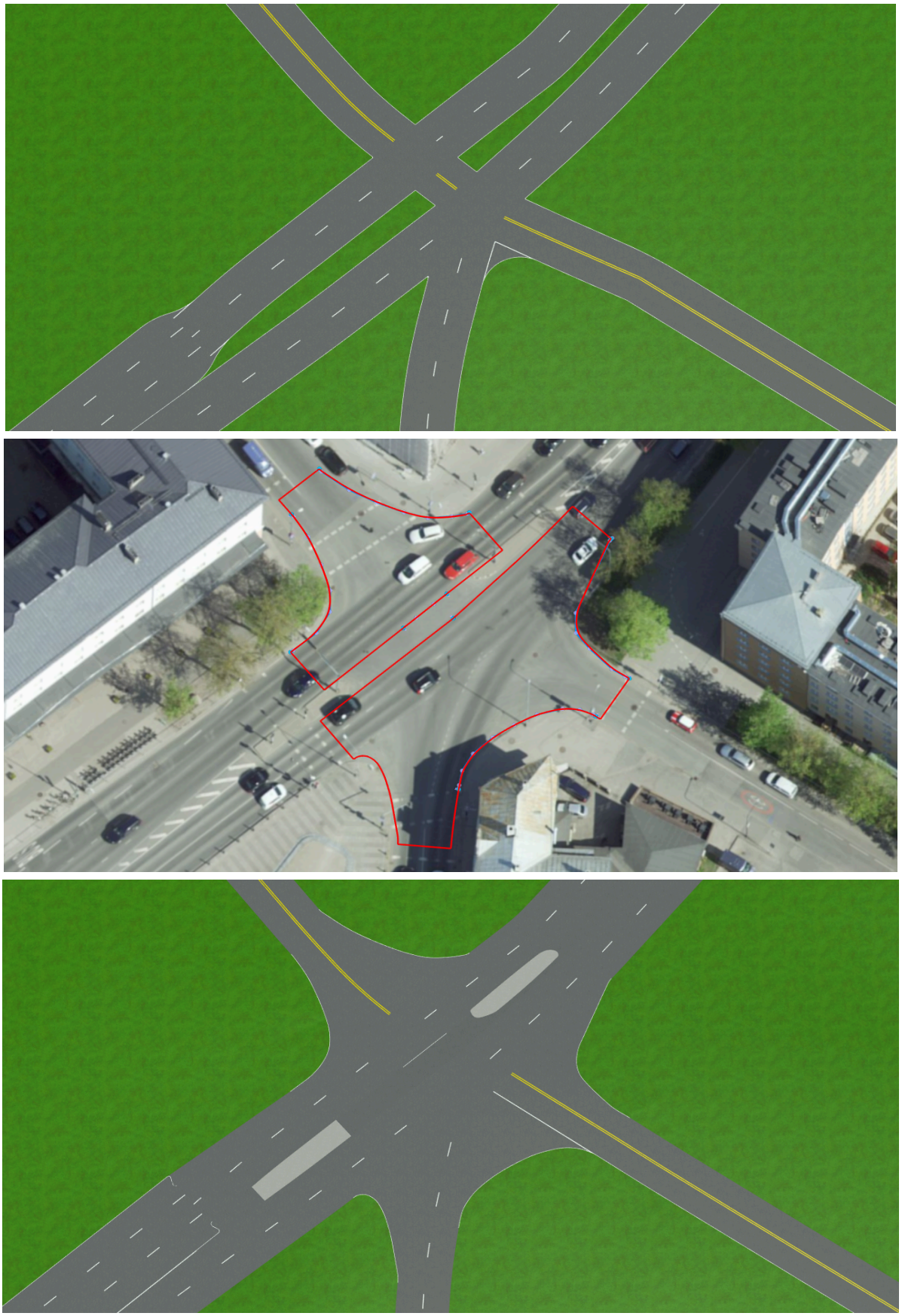


Figure 7: Changes applied to the Riia Hill's intersection (near the Estonian National Defence College) before (top), during the editing process while using orthophoto as a base (middle) and after (bottom). Marked with the red highlight are roadblocks that the "cornering tool" can be used on.

Alongside the modifications mentioned above, some further refinements were implemented to enhance the realism and structural integrity of the digital twin, mainly focusing on the bridges within Tartu City. These adjustments involved altering certain terrain altitude levels to ensure the bridges were accurately represented in relation to their surroundings. This process was crucial for achieving a realistic impression of bridge elevations and alignments, ensuring they integrated seamlessly with the city's topography.

Moreover, to further augment the authenticity and detail of the bridges, additional objects such as railings and columns were introduced in some places. Including these objects enhanced the visual fidelity of the bridges, contributing to a more immersive and accurate representation of the urban landscape in the digital twin.

5.2 Terrain Formation and Elevation Mapping

In the terrain generation process within RoadRunner, the gaps between roads were bridged to create a unified base layer. This foundational layer was then augmented with elevation values derived from the Estonian Land Board's Digital Terrain Models (DTM), ensuring a topographically accurate landscape representation. Using the "Sample Global Elevation" feature, RoadRunner could interpret and apply the elevation data from the raster files (in GeoTIFF format), moulding the terrain to reflect the accurate contours of the area. An illustration of this terrain transformation process, highlighting the enhanced realism and accuracy achieved, is presented in Figure 8.

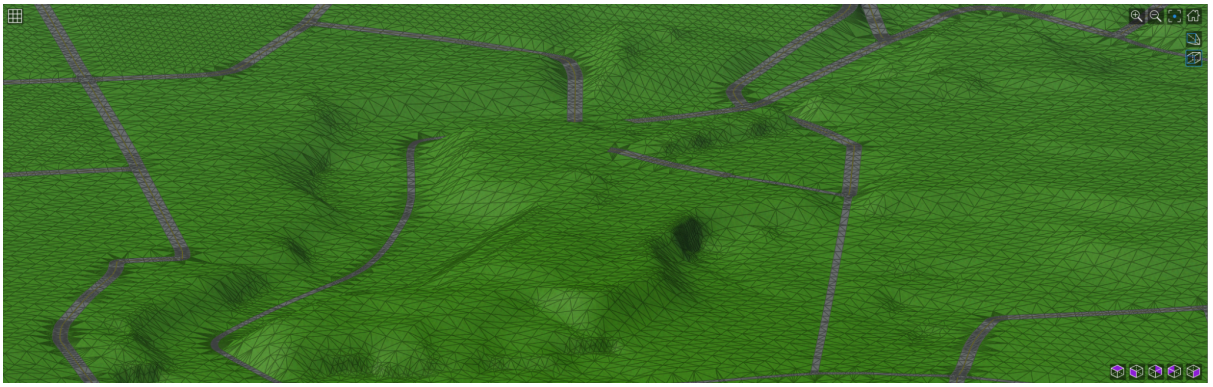


Figure 8: Observing Toome Hill from an aerial perspective reveals the terrain's elevation, shaped by leveraging digital terrain models.

5.3 City Building Construction

The construction of buildings within the project was predicated on foundational data sourced from the Estonian Land Board's 3D model of buildings in Tartu (in OBJ format). Buildings at Tartu's Tähe Street that 3DI had textured were also the Estonian Land Board models. These initial datasets underwent a comprehensive processing phase utilising Blender software, as illustrated in Figure 9. The preliminary step involved integrating the two distinct model sets, eliminating overlapping elements. Following this, object normals were recalculated to correct their orientation, and the models were divided into segments. This segmentation was essential to facilitate the management of the city's extensive building data by preventing the simultaneous loading of all structures.

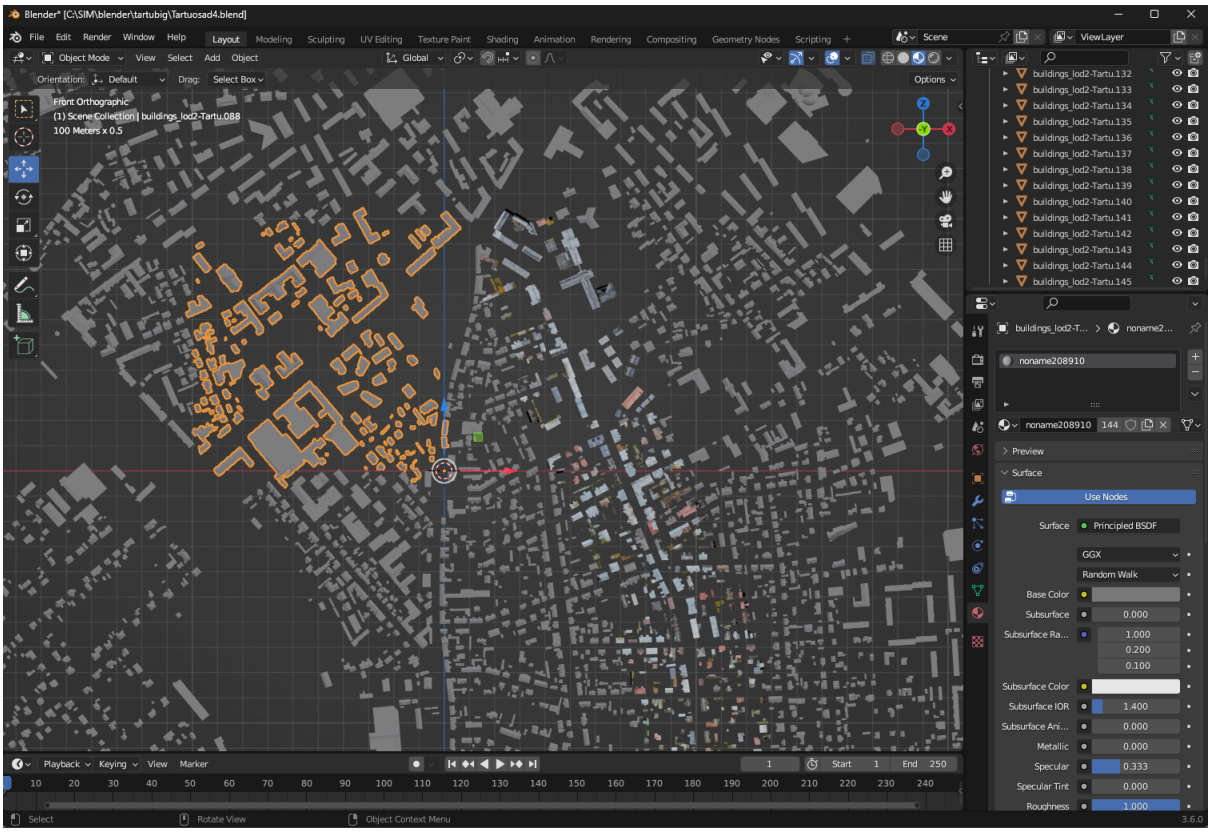


Figure 9: Working with Tartu's building models in Blender.

A custom Python script *export_blender_fbx.py* was developed and executed via Blender's Python API to export each segmented part as an individual file, resulting in 144 Filmbox (FBX) files. These segmented 3D models were then imported into RoadRunner, aligned and calibrated against the terrain. This calibration process leveraged the Estonian Land Board's topographic database of building outlines (ESRI Shape format) to ensure precise placement and orientation. Furthermore, the elevation for each house model was accurately determined using elevation data derived from the Estonian Land Board's point clouds (in LAZ format), ensuring that the structural models were not only spatially accurate in terms of location but also in elevation, enhancing the realism and fidelity of the digital twin of Tartu.

In enhancing the virtual representation of Tartu City, an addition was made by modelling the Tartu TV Mast, a notable landmark. This modelling process utilised Blender and point cloud data as a detailed blueprint for accuracy and precision. The point cloud data provided a robust foundation for creating a clear and accurate 3D model of the TV Mast. Including the Tartu TV Mast model into the digital twin of Tartu enriched the virtual environment with one of the key pieces of the city's skyline and contributed to the realism and depth of the simulation. The model can be seen in Figure 10.

Figure 10: Tartu TV Masts in Blender.

5.4 Implementation of Textures to Roads, Buildings and Ground Areas

The texturing process can be divided into 4 steps:

1. Importing the city model into Blender and splitting it into city plots.
2. Importing city plots into Agisoft Metashape and linking them with drone cameras.
3. Building textures for every city plot.
4. Importing textured city plots into RoadRunner and aligning them with the world.

5.4.1 Importing the City Model into Blender

The initial phase in the workflow involved exporting the entire city scene from RoadRunner as a Filmbox (FBX) model, laying the groundwork for subsequent texturing and modelling processes. Following this export, the comprehensive scene file was imported into Blender.

Given the objective of evaluating the efficacy of the applied texturing methodology, it was resolved that the analysis would focus on a specific portion of Tartu City. The area south of Aardla Street, recognised for its architectural and infrastructural diversity, including residential homes, industrial facilities, warehouses, and a varied network of small streets and highways, was selected for this purpose. This diversity presented a comprehensive testing ground for the texturing process.

Segmenting the territory into distinct city blocks was necessary to manage the computational demands, particularly concerning video memory during the city's loading phase. This segmentation allowed for selective loading of the model, mitigating the need to load the entire cityscape simultaneously and thereby optimising performance. The designated area was divided into nine separate parts, as detailed in Figure 11, facilitating focused analysis and processing. In the final step of this phase, the individually segmented city blocks were all exported in OBJ file format.

Figure 11: Nine city plots south of Aardla Street in Tartu City (the colours are just for illustration).

5.4.2 Importing City Plots into Agisoft Metashape

Following the initial preparation, the next phase involved the comprehensive import of drone-captured aerial photography taken by company 3DI into Agisoft Metashape, a critical step for ensuring that the texturing process was grounded in high-resolution, real-world imagery. This import laid the foundation for detailed photogrammetric analysis and texture mapping.

After importing the aerial imagery, the previously segmented city blocks, selected for their representative diversity, were imported into Agisoft Metashape. This step was crucial for aligning the 3D models with the corresponding aerial photographs, ensuring that the textures accurately reflect the actual terrain and built environment conditions.

The process then required identifying and utilising the drone flight plan specific to the selected city block. This flight plan, essential for understanding the spatial distribution and angles of the captured images, facilitated the precise alignment of the cameras within Agisoft Metashape. An illustration of this drone flight plan, showcasing its coverage of the selected block, is provided in Figure 12.

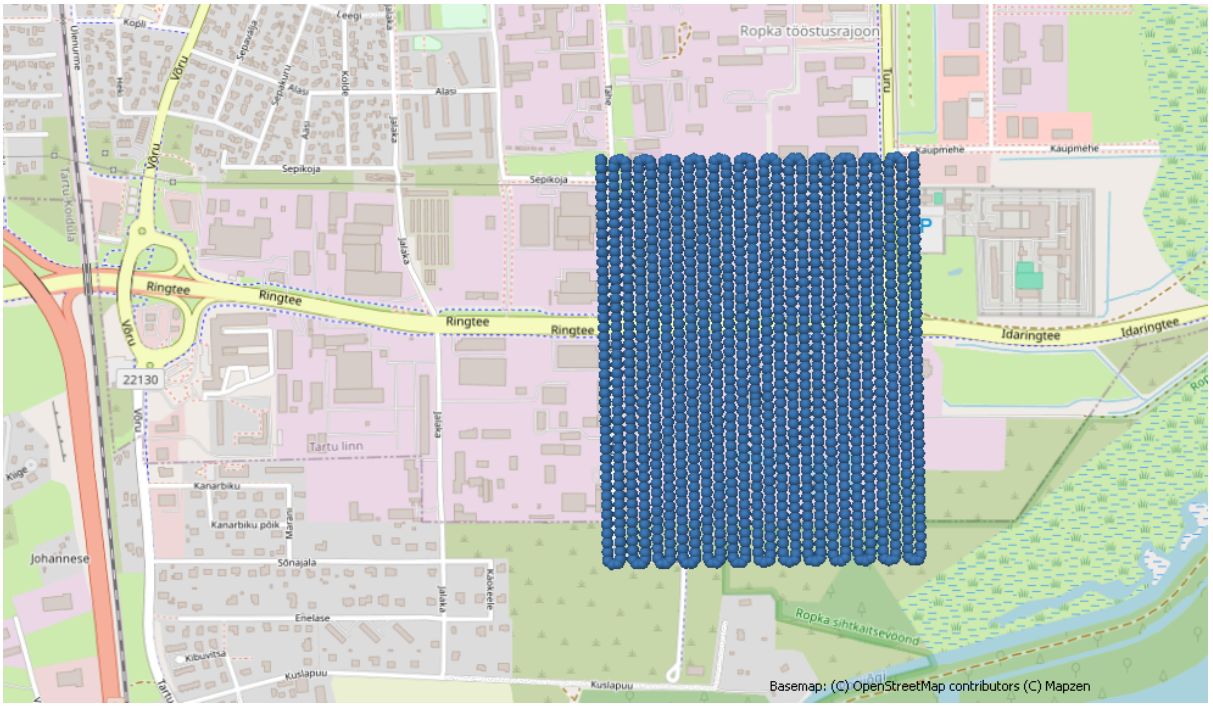


Figure 12: Drone flight path that surrounds the city block chosen. Each blue dot is a camera location when taking a picture.

Following the flight plan identification, aligning the cameras within the software was the next critical step. This alignment process, crucial for ensuring that each aerial photograph was correctly positioned in relation to the 3D model, was guided by specific setup options, as depicted in Figure 13. This alignment ensured that the photogrammetric software could accurately interpret and map the aerial imagery onto the 3D models, achieving a high level of detail and realism in the texture mapping process.

5.4.3 Building Textures for Every City Plot

The third step in the texturing workflow included the construction of textures for the imported model of the selected city block. This phase involved utilising Agisoft Metashape's texturing capabilities, with the setup options for this process detailed in Figure 13. The objective was to apply the aerial imagery as textures to the 3D model, enhancing its visual realism and detail.

To achieve optimal texturing results, a variety of options and methods within the software were rigorously tested. The configuration depicted in Figure 13 was identified as yielding the most accurate and visually appealing outcomes. This phase required careful alignment of the drone-captured photos with the city block model, ensuring that every texture accurately represented its corresponding real-world surface. The results of the texturing process can be seen in Figure 14.

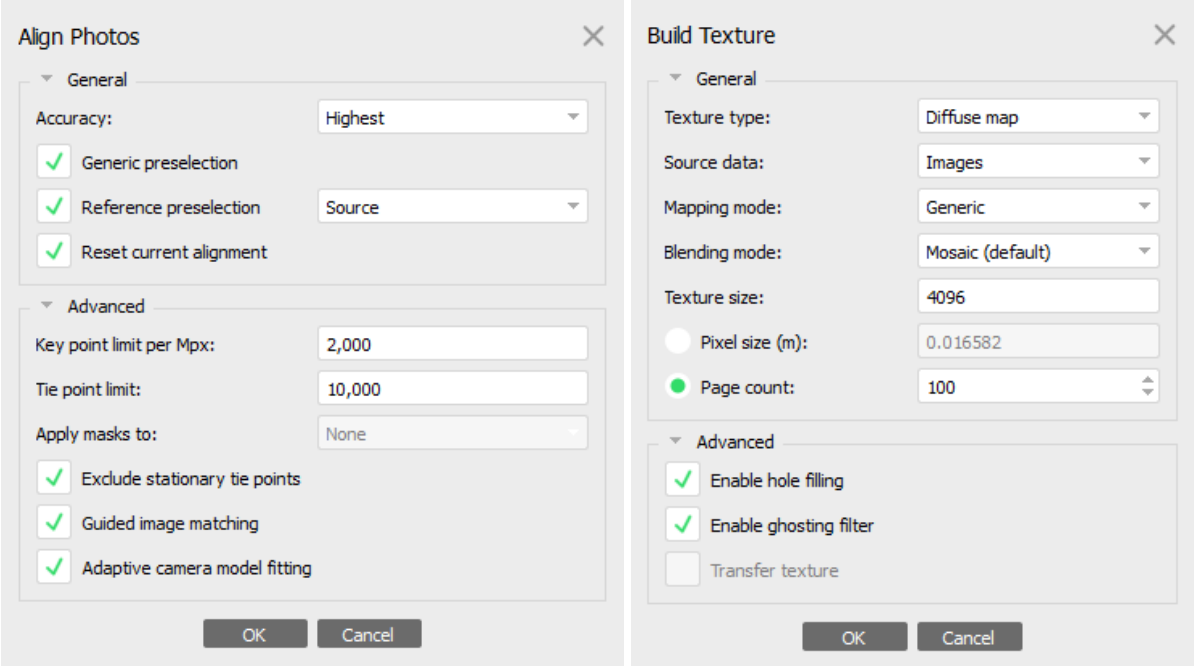


Figure 13: Metashape workflow options to align the cameras and build the model textures. The setup used for the camera alignment process is shown in the left picture, and the setup used for creating the textures is on display on the right.

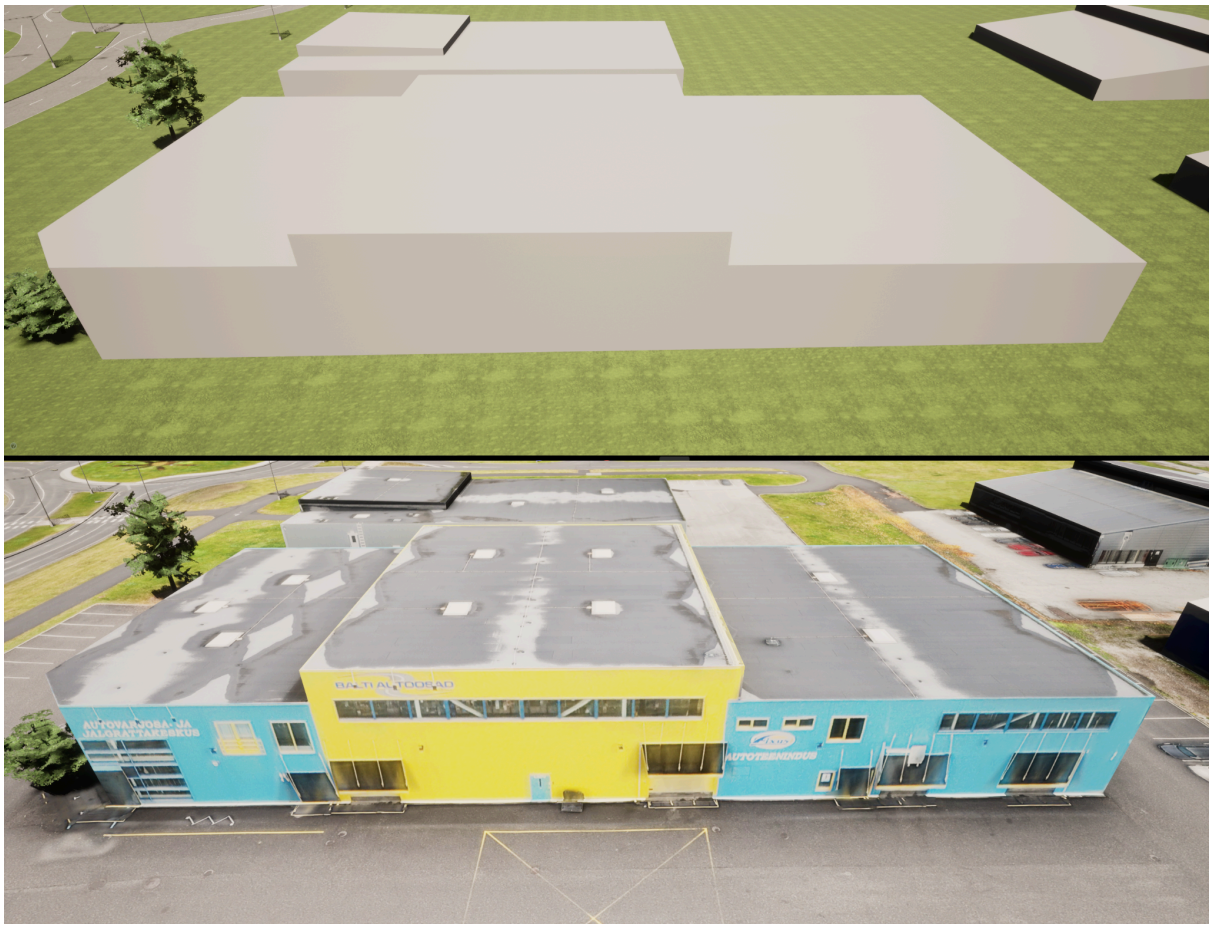


Figure 14: The same building and surrounding area before (top) and after (bottom) texturing process.

5.4.4 Importing Textured City Plots into RoadRunner

Following the successful alignment and texturing, the final step involved exporting the fully textured city blocks. This process included the individual models and the connected textures, ensuring that the entirety of the selected area was cohesively represented.

The culmination of the texturing process involved importing the fully textured city blocks into RoadRunner. This phase required precise alignment of the textured models with the existing digital map elements and each other, ensuring spatial coherence and continuity across the entire model.

A vital aspect of this integration was managing map areas, mainly removing any ground cover within the RoadRunner environment that overlapped with the footprint of the newly imported textured city blocks. This step was essential to prevent redundancy and ensure that the textured models accurately represented the city's surface without duplicating underlying terrain features. You can see the result in Figure 15.

Figure 15: Area south of Aardla Street before (top) and after (bottom) the textured city blocks were added to the scene. 3DI had already textured the buildings on Tähe Street.

Following the alignment and integration of the textured city blocks within RoadRunner, the scene was fully prepared for export, marking its readiness for use in the Unreal Engine. The chosen export format was Filmbox (FBX). All textures were embedded within the FBX file, ensuring the detailed surface information was preserved and seamlessly translated into the Unreal environment.

To optimise the scene for performance and manageability within the Unreal Engine, the export process included segmenting the world into manageable tiles, each measuring 2 km by 2 km. This tiling strategy created 21 tiles, facilitating efficient loading and rendering of the scene within the game engine. This approach enhances the user experience by improving performance and simplifying scene management and navigation in complex urban environments like Tartu.

5.5 Generation of Vegetation

The methodology for generating detailed vegetation information incorporated the use of a GeoPackage (GPKG) file from the Estonian Land Board, which provided comprehensive data on trees, including their longitude and latitude coordinates, altitude, radius, and classification as either coniferous or deciduous. Despite the richness of this dataset, it lacked elevation details, necessitating further processing to achieve a fully dimensional representation of each tree within the digital model of Tartu City.

The Estonian Land Board's approach to generating this comprehensive vegetation data involved an algorithm utilising point cloud data derived from low-altitude ALS operations [23].

Leveraging QGIS software, the elevation for each tree was calculated and appended to the dataset, mirroring the method applied to urban infrastructure elements. This processing phase ensured that each tree's placement within the digital model would accurately reflect its real-world elevation relative to the surrounding terrain.

The project also integrated additional spatial data layers into the QGIS environment, including an OpenStreetMap (OSM) file from road network construction via RoadRunner and topographic information detailing urban cleaning areas, water bodies, and buildings. To prevent unrealistic vegetation placement, a 1-meter buffer zone was established around these features, ensuring a naturalistic separation between trees and artificial structures or water bodies. This safeguard was instrumental in enhancing the model's realism, supported by filters that prevent the placement of trees on roads, within buildings, or in water bodies.

After applying these filters and adjustments, the detailed tree data—including coordinates, altitude, radius, and type—was compiled and exported into a CSV file. The World Geodetic System (WGS84) was chosen as the coordinate reference system for this export, ensuring compatibility with the Unreal Engine.

Given the initial over-density of vegetation in certain areas, a Python script *lesstrees.py* was developed to refine the tree distribution, mandating a minimum separation of 6 meters between individual trees. This adjustment ensured a more realistic and ecologically plausible representation of the urban forest. In the beginning, there were 149810 trees in the dataset. It was brought down to 92302 after all the filtering.

The QGIS editing process, including the application of these methodologies and the final vegetation adjustments, is exemplified in Figure 16. This figure illustrates the contrast between the original vegetation density and the modified layout, highlighting the effectiveness of the applied filtering and spacing criteria in achieving an accurate and visually appealing digital representation of Tartu's vegetative landscape.

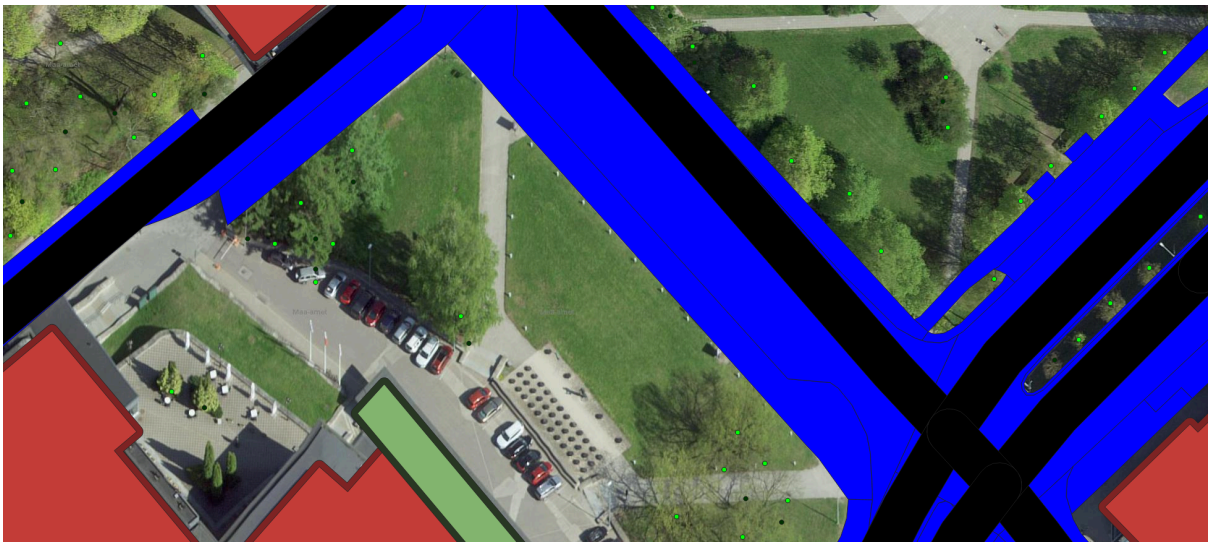


Figure 16: QGIS desktop application showing imported tree locations as green dots. After running the Python script to lower the vegetation density, the trees left out are marked as dark green vs. those used (light green). The picture also includes the road area (black), urban cleaning area (blue) and buildings (red+green).

5.6 Set Up for Urban Infrastructure

The acquisition of location data for street lights, bus stops, and trash containers was an initial step in developing Tartu City's digital twin, with the data being sourced from Tartu City's geoHub in CSV format. This dataset provided each object's precise latitude and longitude coordinates; however, the absence of elevation information presented a challenge for integrating these objects into the 3D environment in a manner that accurately reflected their real-world positioning relative to the terrain.

The QGIS software was employed to rectify this. The workflow commenced with importing the location data from the CSV files into QGIS, as seen in Figure 17. This step was followed by integrating Tartu's Digital Terrain Models (DTMs), available in GeoTIFF format, into the QGIS environment. These DTMs provided detailed elevation data across the city's landscape.

Figure 17: QGIS desktop application showing locations of imported street lights (yellow), bus stops (blue) and trash containers (red) as dots.

The elevation for each object was then calculated using the "Point Sampling Tool" in QGIS. This tool facilitated the precise determination of elevation values by referencing the DTMs and assigning accurate altitude information to each street light, bus stop, and trash container based on their geographic coordinates.

After successfully calculating and adding elevation data to each object, the enhanced dataset was exported back into CSV format. The World Geodetic System (WGS84)[16] was selected as the coordinate reference system for this export, ensuring the data would be compatible with Unreal Engine.

5.7 Generation of High-definition map

High-Definition (HD) maps represent a significant advancement in digital mapping technology, offering unparalleled detail and accuracy in depicting roads and their surrounding environments. These maps transcend the capabilities of traditional GPS-based navigation systems by providing comprehensive information that includes not only basic road layouts but also detailed attributes such as lane markings, road signs, traffic signals, and specific road features like geometry, curvature, and gradient. [24]

[16] https://gisgeography.com/wgs84-world-geodetic-system/

HD maps are utilised in the OpenDrive format within the context of the CARLA simulation environment. OpenDrive is an open-format specification that establishes a standardised data model for the logical representation of road networks. This format enables the detailed and structured description of road networks, capturing the physical geometry of roads and their topology and semantics. Such descriptions include lane configurations, road markings, traffic signs, and lighting signals, providing a comprehensive framework for simulating real-world traffic conditions.

In the CARLA simulation, vehicles (called "traffic cars") navigate the virtual environment by adhering to the road networks and traffic rules defined within the OpenDrive maps. These vehicles follow specific lane assignments, obey traffic signals—halting at stop lines when encountering red lights—and execute yielding manoeuvres as dictated by the map's logic. This dynamic interaction between the simulation's traffic participants and the detailed road network model underscores the critical role of HD maps in facilitating realistic and complex traffic simulations, offering a robust platform for testing and developing autonomous driving systems.

A novel approach was required to expand the digital twin project to encompass the entirety of Tartu beyond the already detailed HD map of the city centre. The city centre's HD map, crafted by hand in earlier stages, provided a robust foundation but could not feasibly be extended to the entire city due to the manual labour involved. Automation was needed to address this challenge and achieve comprehensive coverage.

5.7.1 OpenStreetMap to OpenDrive conversion

The solution emerged by leveraging a Python script[17] that the CARLA simulator's developers developed. This script facilitated the transformation of OpenStreetMap (OSM) data into an OpenDrive format, effectively converting the broad and accessible OSM road network data into the detailed, simulation-ready OpenDrive map. The downside was that the conversion did not include the info about the stop lines and traffic lights. Recognising the specific needs of the Tartu project, this script *osm_to_xodr.py* was adapted to incorporate the origin point of the project's geospatial projection, ensuring that the generated HD map would align accurately with the existing spatial data framework.

5.7.2 Integration of Elevation Data

The input for this conversion process was the OSM file previously generated and utilised for road network creation in RoadRunner. The primary challenge in utilising the OSM data was its lack of elevation information, which is crucial for creating a realistic and functional 3D simulation environment. To overcome this hurdle, a new Python script, *xodr_altitude.py,* was devised specifically for the project. This script utilised Tartu's Digital Terrain Models (DTMs) to enrich the OSM data with precise elevation information for each map node. By embedding elevation data directly into the map nodes, the script ensured that the resulting OpenDrive map would accurately reflect the city's road geometry, layout, and terrain elevations.

5.7.3 Integration of Two Maps

An existing high-definition OpenDrive map, crafted for ADL by Karl-Johan Pilve[18], already encapsulated the city centre and additional surrounding areas of Tartu, complete with elevation data, serving as a foundational element for the project. The overarching aim was to

[17] https://github.com/carla-simulator/carla/blob/master/PythonAPI/util/osm_to_xodr.py
[18] karljohan30@gmail.com

seamlessly integrate this pre-existing map with a newly generated OpenDrive map, incorporating elevation details added through a Python script, thereby creating a comprehensive and unified road network representation for the entire area.

The integration process commenced with importing the newly created OpenDrive map into RoadRunner. This map served as the groundwork for generating an initial road network. Roads that duplicated those present in Karl-Johan Pilve's map were identified and subsequently eliminated to avoid redundancy and ensure coherence in the road network layout.

After removing overlapping roads, Karl-Johan Pilve's map was introduced into the same RoadRunner scene. Utilising this map as input, a road network was generated, reflecting the detailed and accurate representation established by Pilve's work. The critical task of merging these two road networks involved the construction of new intersections at junctures where the respective roads from each map converged, ensuring a smooth and logical integration of the two datasets.

5.7.4 Finalisation and Export

The final phase of the process entailed refining the elevation data for the combined road network. By employing Digital Terrain Models (DTMs) and RoadRunner's "Project Roads" feature, new elevation values were assigned to the entire unified map. This step achieved a topographically accurate and continuous representation of the road network, aligning with the real-world undulations and gradients of Tartu's landscape.

Upon completing these steps, the OpenDrive file encapsulating the comprehensive road network was generated and exported from RoadRunner, ready to be used with the CARLA simulator.

Figure 18 illustrates the intricate process of editing the Riia-Turu intersection in RoadRunner, showcasing the level of detail and precision in combining and refining the road networks to achieve a seamless and accurate urban model.

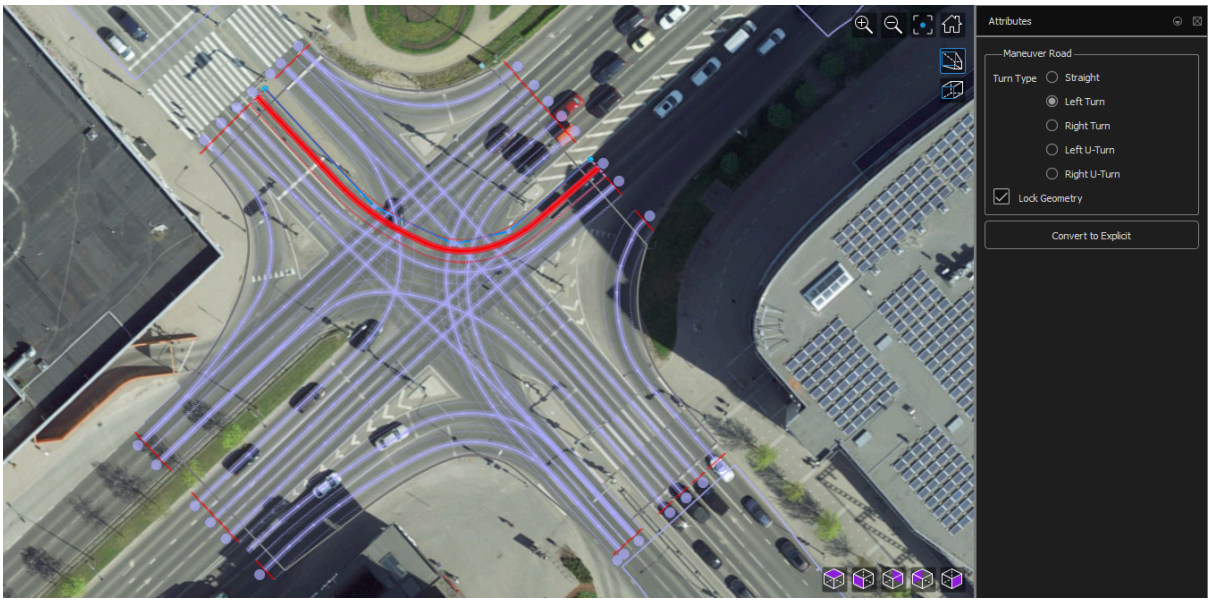


Figure 18: Editing Riia-Turu intersection using aerophoto as a guideline.

5.8 Project Assembly in Unreal Engine

The transition of Tartu's digital twin into the Unreal Engine marked a significant phase in the project, involving the import of 21 tiles from the RoadRunner environment, formatted in Filmbox (.fbx). This importation followed the structured guidelines provided by the CARLA simulator's official webpage[19], ensuring a smooth and accurate integration of the detailed city model into the Unreal Engine.

Following the import, a specific focus was placed on enhancing the realism of the map segment south of Aardla Street. This area initially featured original textures derived from RoadRunner, which, while accurate, did not leverage the advanced textural capabilities provided by the CARLA simulator. To rectify this and enhance the visual fidelity of the road surfaces, these original textures were replaced with new ones. These replacement textures were equipped with specific material properties, enabling them to interact more naturally with environmental effects, such as lighting and weather conditions, thereby enhancing the overall realism of the simulation environment.

However, an issue was identified with the drone textures in the area south of Aardla Street; they appeared overly bright when subjected to the scene's lighting conditions. To address this challenge, a Python script *fix_textures.py* was developed and executed, facilitating the adjustment of many texture settings. This script enabled precise calibration of the textures' appearance within the scene, ensuring they blended with the existing lighting and atmospheric conditions, thus correcting the brightness issue and significantly improving the visual coherence and realism of the simulation environment.

The Level of Detail (LOD) system was employed for various objects to enhance the simulation's performance within the Unreal Engine. This system is designed to optimise gaming and simulation environments by dynamically adjusting the complexity of 3D models based on their proximity to the camera. As an object becomes more distant, its detail level decreases, reducing the computational resources needed for rendering. This optimisation is essential for sustaining high performance without compromising the richness and expansiveness of the 3D environment.

The Unreal Engine's LOD generation mechanism automatically reduces polygon counts in static meshes, creating multiple LODs for a single object. This process employs quadratic mesh simplification, which evaluates the visual impact of merging vertices and collapsing edges within the mesh. It prioritises modifications that minimise visual discrepancies, ensuring the simplification process preserves the object's overall appearance. The simplification continues until the mesh reaches a predetermined target for the number of triangles, effectively generating LODs that maintain visual integrity while significantly reducing complexity. A key factor in the LOD system's effectiveness is using "screen size" as a metric for determining when to switch between different LODs. Screen size, calculated as the object's size as a percentage of the screen's total area, serves as a threshold for LOD transitions. By adjusting the screen size values, the system can finely tune when to switch to a less detailed model as the object moves further from the viewer's perspective. [25]

In practice, objects within the simulation environment were assigned different LODs, typically ranging from LOD 0 (full detail) to LOD 2 (minimal or no detail), based on their distance from the camera. LOD 0 displayed objects at 100% detail when viewed up close, while LOD 1 significantly reduced the triangle count to just 10% for medium-distance viewing. LOD 2, intended for objects far from the camera, often reduced the detail to zero,

[19] https://carla.readthedocs.io/en/latest/tuto_M_add_map_alternative/#alternative-methods-to-import-maps

effectively turning off rendering for distant objects and further conserving processing resources.

This LOD strategy aimed to progressively decrease the detail of objects as they recede from the camera's view and, in some cases, to cease rendering them entirely when they are too far away to impact the user's experience. The optimal balance between visual quality and performance efficiency was achieved by testing different screen size values, ensuring a smooth, immersive experience in expansive, detailed 3D environments.

5.8.1 The Implementation of Vegetation

For incorporating vegetation into the Unreal Engine, a similar approach to urban infrastructure was adopted, involving a series of Python scripts designed to prepare and optimise the data for simulation.

Initially, two scripts (*trees_clean.py, trees_unreal.py*) were employed: one for cleaning the data to ensure its accuracy and relevance and another for converting the geographic coordinates from the World Geodetic System (WGS84) to the coordinate system used by the Unreal Engine.

The third script, *trees_shuffle.py,* managed the vegetation data by shuffling it and dividing it into six distinct segments. This division resulted in three sets of deciduous trees and three sets of coniferous trees. Each was saved as a separate CSV file.

The segmented data sets were then imported into the Unreal Engine as data tables. This step converted the static data into a dynamic format that could be manipulated within the Unreal Engine, setting the stage for the next implementation phase.

For each data table, a blueprint file was created (check Figure 19). These blueprints served as templates for generating instances of trees within the simulation environment. An advanced optimisation technique, hierarchical instanced static meshes (HISM), was utilised to populate the scene efficiently with these trees.

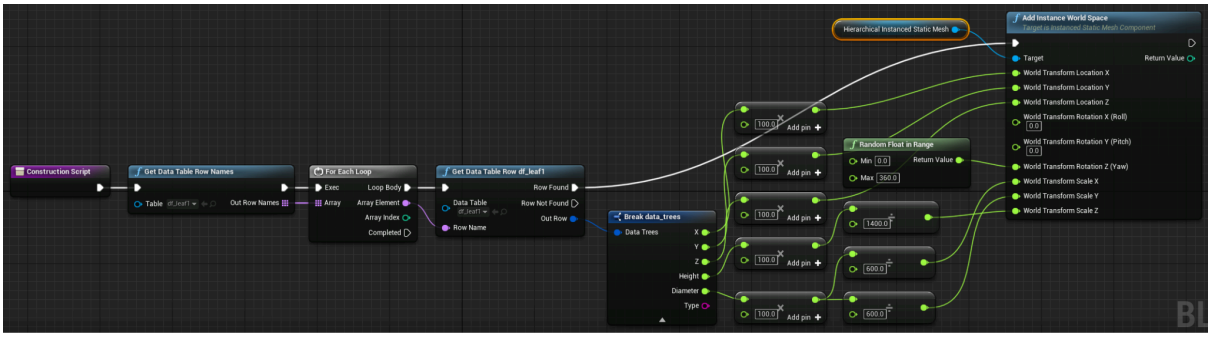


Figure 19: Blueprint setup for spawning trees using hierarchical instanced static meshes.

HISM is an evolution of the Instanced Static Mesh (ISM) concept, which allows for rendering multiple instances of a mesh through a single draw call, significantly reducing the computational resources required compared to rendering each instance separately. HISM builds upon this by introducing a hierarchical Level of Detail (LOD) system and spatial organisation. This sophisticated system enables the Unreal Engine to cull instances not visible to the camera and dynamically adjust the LODs for each visible instance based on its distance and angle relative to the camera. As a result, only those instances within the viewer's line of sight and at an appropriate level of detail are rendered. [26]

HISM in the Unreal Engine offers substantial benefits for rendering large-scale environments with repetitive elements, such as forests or urban landscapes. This optimisation technique not only leads to significant performance improvements by minimising draw calls and optimising memory usage but also dynamically enhances the visual quality of the scene by adjusting the detail level of instances based on their proximity to the viewer.

The tree models utilised within the simulation were sourced from the CARLA simulator's asset pack and were suitable for enhancing the realism of the virtual environment. In line with the approach adopted for urban infrastructure, these tree models' Level of Detail (LOD) settings were configured to optimise the simulation's performance while maintaining visual quality.

The LOD configuration was tailored to ensure that trees were rendered with full detail when viewed up close (LOD 0), gradually reducing in quality as the distance from the camera increased. This gradual reduction in detail helped to minimise the computational load on the system, as rendering high-quality models for distant trees that contribute minimally to the scene's visual fidelity is unnecessary. Moreover, the setup included provisions to stop rendering trees far beyond a certain distance from the camera, further enhancing performance by reducing the number of objects the engine needed to process.

This LOD strategy for the vegetation, akin to that used for urban infrastructure, played a crucial role in achieving an efficient balance between rendering quality and performance. By dynamically adjusting the detail level of tree models based on their proximity to the viewer, the simulation could handle densely vegetated areas and extensive urban landscapes without compromising on frame rates or visual appeal. This approach ensured a seamless and immersive experience, essential for simulations intended for autonomous vehicle testing or urban planning analyses, where detail and performance are the key. An example of the scene where it all is being put together in Unreal Engine can be seen in Figure 20.

Figure20: Unreal Engine's world scene, where the implementation of Level of Detail (LOD) settings and visibility optimisations is evident through the selective rendering of objects based on their distance from the viewer. Trees, street lights and ground areas are not rendered when they are far from the camera.

5.8.2 The Implementation of Urban Infrastructure

To seamlessly integrate street lights, bus stops, and trash containers into the scene, CSV files containing the geographic coordinates and other relevant data extracted from QGIS were utilised. This data transfer and integration process was facilitated by developing three distinct Python scripts, each designed to perform specific tasks for placing and rendering these urban infrastructure elements within the Unreal Engine’s environment.

The first scripts (*bins_clean.py, bus_clean.py, lights_clean.py*) were tasked with cleaning the data, ensuring any inconsistencies, errors, or irrelevant information in the CSV files were rectified or removed.

The second batch of script (*bins_unreal.py, bus_unrealpy, lights_unreal.py*) addressed the need to convert the geographic coordinates from the World Geodetic System (WGS84) used in QGIS to the coordinate system employed by the Unreal Engine.

The final scripts (*bins_add.py, bus_add.py, lights_add.py*) handled the import of the cleaned and converted data into the Unreal Engine, placing each street light, bus stop, and trash container model within the scene according to the geographic coordinates specified in the CSV files.

The models used to represent these infrastructure elements were sourced from RoadRunner's and the CARLA simulator's asset packs or were already included within the Tartu City Centre's digital model. To optimise performance without compromising the visual quality of the scene, Level of Detail (LOD) settings were adjusted for each model. These settings ensured that as the camera moved away from an object, the object's details would progressively diminish until it eventually disappeared from view. This approach not only improved the simulation's performance by reducing the rendering load on distant objects but also maintained a high level of detail for those within the immediate vicinity of the viewer, thereby achieving a balance between visual fidelity and computational efficiency.

In addition to these three items, one additional object was added to test the animation possibilities in the scene. The underground parking lot's door in the Delta Centre building was animated, making it only open if it is being approached by a UT Lexus vehicle.

5.8.3 Build Process

Before initiating the build process for the simulation environment, final adjustments were made to ensure optimal visual quality and performance. These adjustments included configuring the default weather settings to achieve the desired atmospheric conditions within the simulation. A crucial modification was made to the lighting settings for objects within the scene: the "Affect Distance Field Lighting" option was turned off. When enabled, this lighting feature can sometimes introduce visual artefacts that detract from the scene's overall realism and visual coherence. Turning off this option helped to eliminate such artefacts, ensuring a cleaner and more consistent visual presentation. Finally, the OpenDrive High-Definition map created for Tartu City was moved to the required location.

With these adjustments, the scene was fully prepared for the build process. The next step was compiling and finalising the simulation, enabling it to be deployed for testing, analysis, or demonstration purposes within the CARLA simulation platform.

6. Results and Discussion

This chapter describes how realistic and authentic the virtual world was compared to the real-life version. What kind of different objects existed, and how many were added to the virtual Tartu. It also shows how successful the texturing process was using the drone photos and whether Autoware Mini[20] autonomous driving technology could control the vehicle in the simulation and reach the navigation goal. This chapter also includes the results from the in-person test, where participants had to re-orientate themselves in the virtual world and finally, the benchmark tests.

6.1 Virtual World Characteristics

The area covered by the digital twin of Tartu City was around 46 square kilometres. Figure 21 provides a visual comparison, illustrating the designated area on a map alongside its virtual representation in the RoadRunner environment.

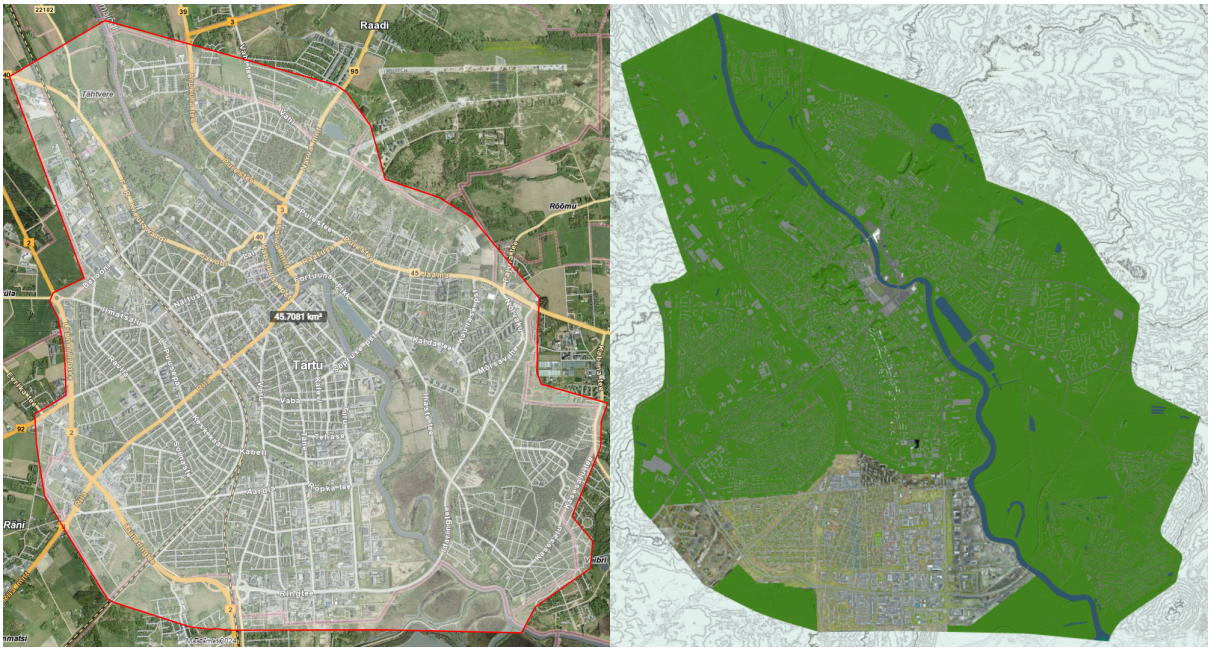


Figure 21: Area covered by digital twin on the map (left) and in RoadRunner (right).

The digitised road network of Tartu City encompasses a total length of 518 kilometres. Implementing a dynamic time-of-day cycle added the realism of the simulation, with street lights automatically illuminating the roadways as darkness fell. Figure 22 captures the Kaubamaja junction under various lighting conditions and during rainfall, showcasing the simulation's capability to reproduce natural lighting and weather effects with high fidelity.

Furthermore, the simulation's attention to detail is evident in its handling of weather conditions, such as rainfall. The introduction of rain visually alters the environment and affects the road textures, allowing for realistic puddles on the road surface. Figure 23 exemplifies this feature, displaying raindrops creating ripples in a puddle, enhancing the immersive experience of the simulation by replicating the visual and physical effects of rain on urban surfaces.

[20] https://github.com/UT-ADL/autoware_mini

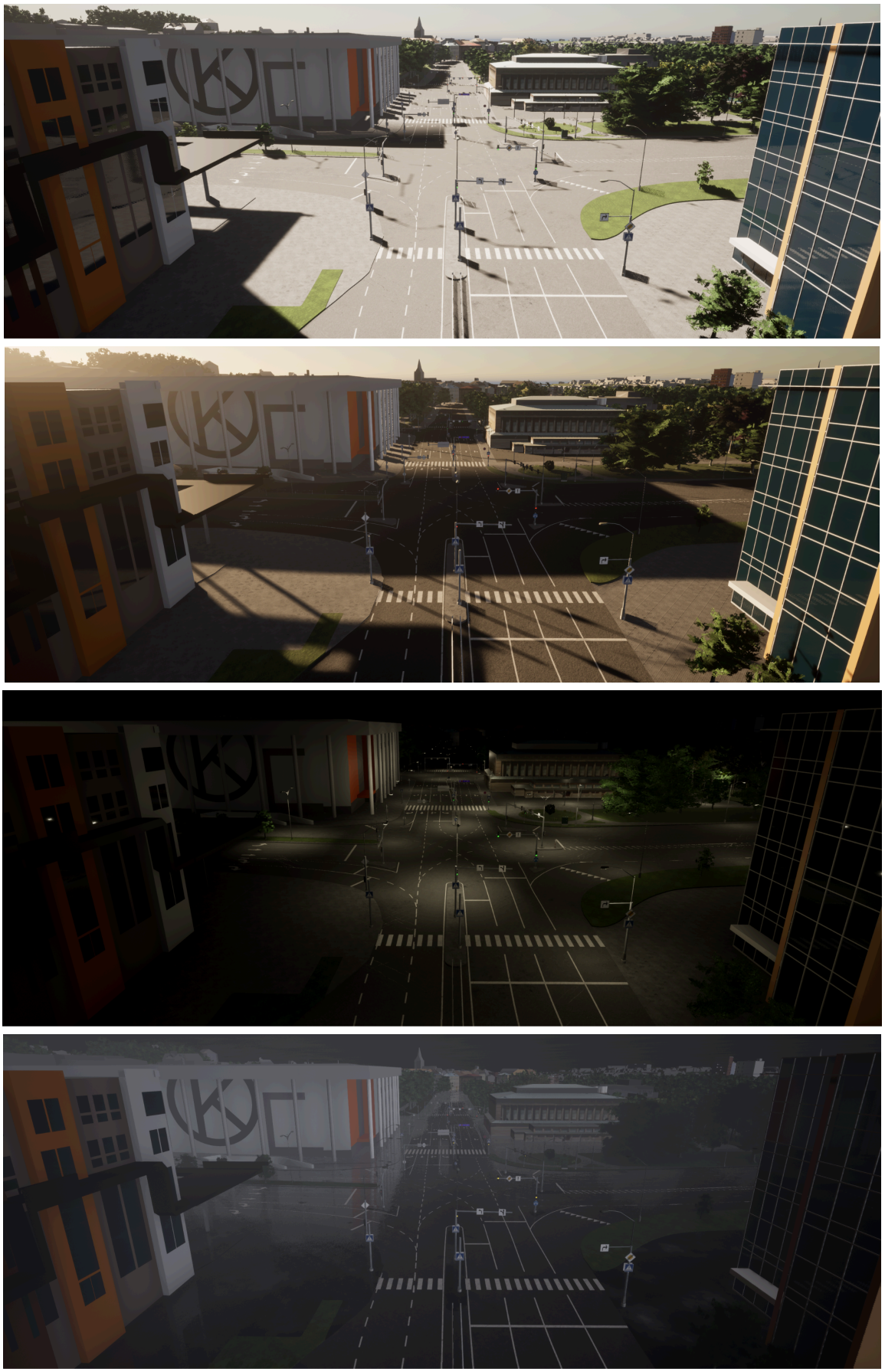


Figure 22: From the top: Kaubamaja junction during the day, in the evening, during the night and with heavy rainfall.

Figure 23: Raindrops hitting the puddle at the side of a road.

Table 5 provides a comprehensive overview of the different structures built and integrated into the digital Tartu City. This table enumerates the various elements placed within the simulation, such as the road network, vegetation, and the city's infrastructure.

Table 5: Amount of structures created within the virtual world

Structure	Amount
Road network	518 kilometres
Bus stops	325
Trash containers	245
Street lights	14699
Trees	92302
Structures built with Blender	2

6.2 Results from Using Drone Photos for Texture Mapping

The use of drone photography for generating textures for roads, buildings, and terrain in the simulation of Tartu City yielded a range of outcomes, as illustrated by Figure 24 and Figure 25. These visual examples highlight the successes and challenges inherent in the texturing process. While mapping these high-resolution aerial images onto the 3D models was not always flawless, introducing these textures significantly enhanced the visual appeal and realism of the virtual environment.

Incorporating drone-derived textures brought the simulation a new level of detail and authenticity, offering nuanced representations of surface materials and urban landscapes. This approach captured real-world variability and imperfections, creating a more immersive and convincing virtual space. Despite occasional inaccuracies in texture mapping, the overall effect was a noticeable improvement in the atmospheric quality of the simulation. The added textures provided a richer, more dynamic backdrop to the simulated Tartu, effectively capturing the essence and aesthetic of the actual city.

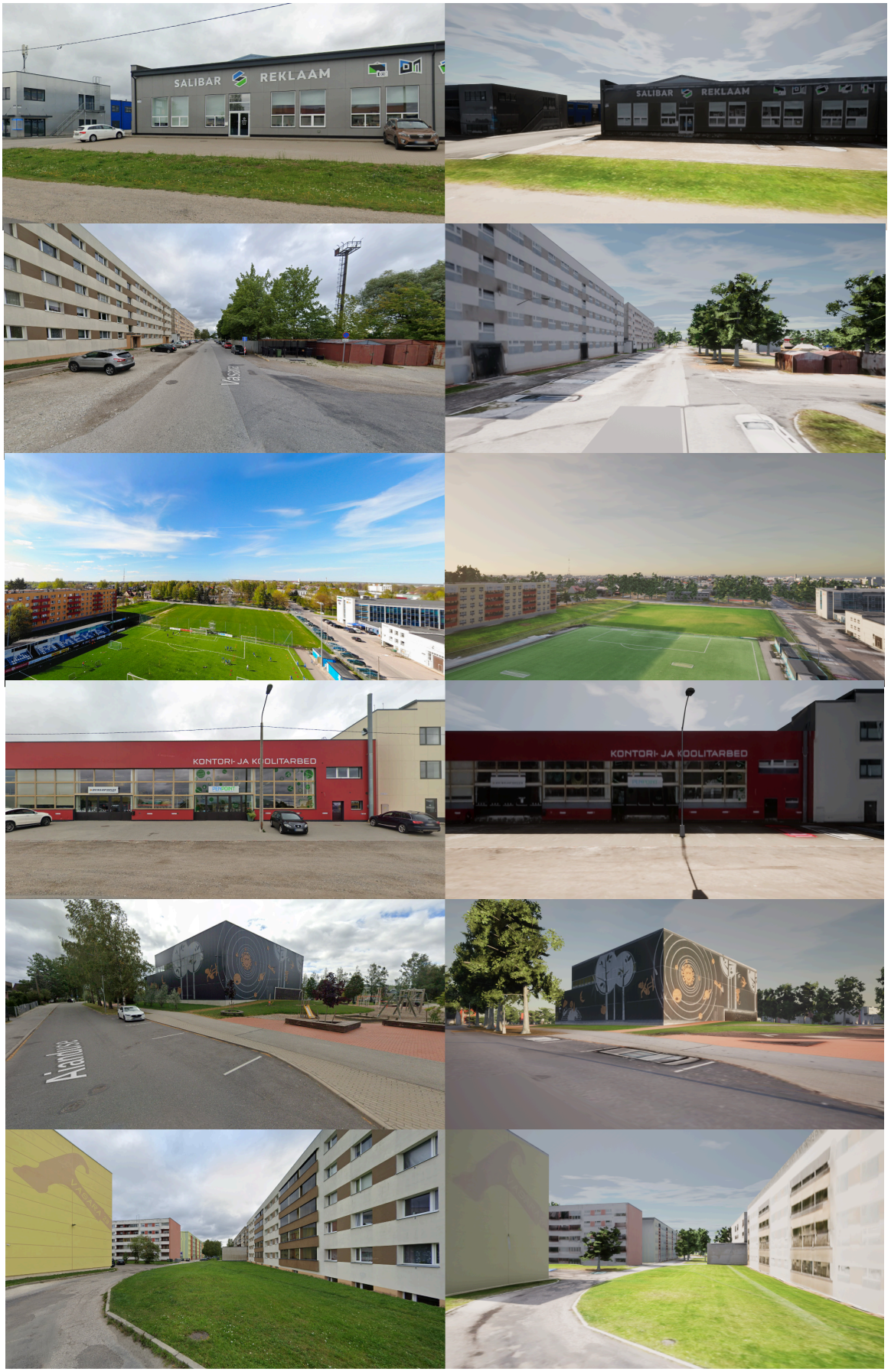


Figure 24: Comparing photos taken in the real world (left) versus the ones in the simulator. This batch of pictures is an example of where drone photography worked well.

Figure 25: This array of pictures is an example of drone photography that had challenges with texture mapping.

However, despite these advancements, the main problems encountered with texture mapping onto prebuilt models are shown in Figure 24. These issues illustrate the complexities of accurately applying real-world textures to virtual structures and underscore the importance of precise model geometry and image processing techniques.

A frequent issue observed was the misapplication of roof textures onto building walls, creating the false impression of incorrect building heights. This problem often stemmed from the high altitude at which drones captured photos and the absence of eaves in the house models. This discrepancy led mapping software to project textures meant for eaves onto adjacent wall surfaces incorrectly. The solution to this problem involves refining the 3D

models to include the missing geometric details, ensuring that textures align correctly with their intended surfaces. The top part of Figure 24 illustrates this specific challenge.

Another common problem was the projection of vegetation textures onto the walls of buildings. This usually occurred when trees or bushes were very close to buildings, obstructing the drone's ability to capture clear images of the wall surfaces. Such instances resulted in textures that inaccurately included elements of the adjacent vegetation. The middle part of Figure 24 demonstrates this issue. Employing masking technologies to digitally remove vegetation from the images before texture application is a potential solution to this challenge, ensuring that only the appropriate architectural textures are applied to building surfaces.

Misalignments and visible seams in the texture mapping often arose from incorrect stitching of images. These errors could be attributed to insufficient overlap between photos, low-quality images, or the mapping software's inability to interpret complex geometric structures accurately. Ensuring a comprehensive dataset with ample photographic coverage from multiple angles can mitigate these issues by providing the software with a richer context for accurately stitching images. The bottom part of Figure 24 highlights examples of stitching errors and their impact on the visual integrity of the 3D model.

Addressing these texture mapping challenges requires a sophisticated approach, including improvements to 3D model geometry, advanced image processing techniques, and the collection of more comprehensive photographic datasets. Refining the methodology behind texture application, the simulation gains depth and detail that closely mirrors the real world.

This effort to overcome technical hurdles underscores the significant value of integrating drone-captured imagery into virtual environments. Even when the technical execution falls short of perfection, the strategic use of such textures can substantially elevate the user's experience. Offering a more engaging and believable representation of the environment, this balance between technical precision and atmospheric impact is crucial. It highlights the importance of accurately replicating the physical world and evoking a sense of place and realism in developing digital twins and other advanced simulations.

6.3 Using Autoware Mini Autonomous Driving Software

Autoware Mini (AM), developed by Tartu University's Autonomous Driving Lab[21], represents a streamlined Python-based autonomy software that is open-source and designed for both real-world and simulated applications. This software represents an innovative approach to autonomous driving technology, offering a versatile and accessible solution for navigating vehicles through diverse environments. Figure 26 shows AM driving a car in the CARLA simulator.

[21] https://adl.cs.ut.ee/

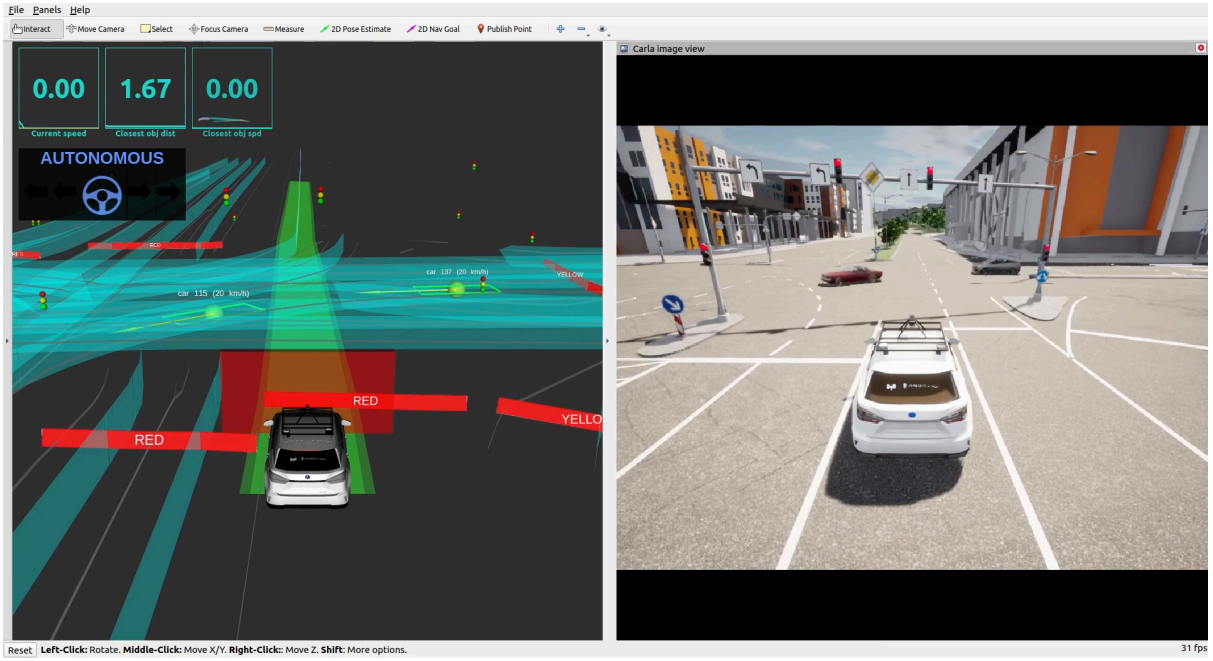


Figure 26: An EGO vehicle driven by Autoware Mini stopped at the red light Riia-Turu intersection.

AM was deployed to navigate the virtual streets of Tartu using the CARLA simulator, demonstrating its capability to handle various navigation goals. The software's successful completion of assigned routes showcases its ability to accurately follow driving lanes and respond appropriately to traffic lights, ensuring a realistic and safe driving experience within the simulated environment. The whole process can be seen in video[22].

Integrating other traffic cars into the simulation further tested AM's functionalities, particularly its object detection and distance measurement capabilities. The EGO vehicle, controlled by AM, exhibited advanced situational awareness by detecting surrounding vehicles, maintaining a safe following distance, and adhering to its intended path without compromising the mission objectives. The successful test was covered in video[23].

The test utilising AM within the digital twin of Tartu City yielded compelling evidence of the virtual environment's realism and accuracy, particularly regarding the generation of sensor data for autonomous driving applications. The similarity between the digital twin and the real-world Tartu in terms of sensor data production underscores the high fidelity of the simulation, affirming its value as a tool for developing and testing autonomous driving technologies.

The ability of Autoware Mini to navigate both the real and virtual Tartu using the same setup highlights the digital twin's effectiveness in replicating real-world conditions. This seamless transition between actual and simulated environments indicates that the digital twin accurately captures the essential characteristics and dynamics of Tartu's road network, traffic systems, and urban layout. Such a high degree of correspondence ensures that algorithms and systems developed and tested within the virtual environment directly apply to real-world scenarios.

[22] https://www.youtube.com/watch?v=BnHtTW02rFY
[23] https://www.youtube.com/watch?v=0nVbxaEhDPs

6.4 RACE to the Delta Centre Time Trial

6.4.1 Introduction

This test aimed to evaluate the navigational intuitiveness and realism of the virtual Tartu City within the CARLA simulator, focusing on participants' ability to re-orient themselves and navigate to the Delta Centre from various predetermined locations around the city. The underlying objective was to assess how well the digital twin of Tartu City facilitates user orientation and navigation, mirroring the challenges and cues one would encounter in the real world.

6.4.2 Method

This unique test aimed to assess participants' ability to navigate and re-orient themselves within a virtual representation of Tartu City, utilising the CARLA simulator. The test involved a diverse group of five individuals who have resided in Tartu for their entire lives, providing them with a deep familiarity with the city's layout. Four participants held driver's licenses for over five years, indicating substantial real-world driving experience. Additionally, four participants were identified as casual gamers, suggesting varying levels of comfort and familiarity with video game controls, which could influence their navigation efficiency in the virtual environment.

The input device for the simulation was the Logitech G920[24] steering wheel, a popular choice for driving simulators due to its realistic feedback and control features, which enhance the immersive experience of virtual driving.

Participants were challenged to locate and travel to the Delta Centre from 10 different starting points scattered throughout the virtual Tartu City, as depicted in Figure 27. The spawn points were selected to be near one of the more known landmarks in Tartu. The primary objective was to achieve the shortest possible time to the destination, with a stopwatch measuring the duration from the moment of spawn. Importantly, participants were informed that adherence to traffic rules was not required, allowing them to focus solely on speed and directness of route.

To ensure a fair and informed starting point for all participants, each was afforded the opportunity to acclimate to the simulator's controls and the digital environment before the commencement of the time trial. This preliminary round included locating their actual homes within the virtual city, an exercise designed to enhance their spatial awareness and comfort level with the simulator setup.

[24] https://www.logitechg.com/en-us/products/driving/driving-force-racing-wheel.html

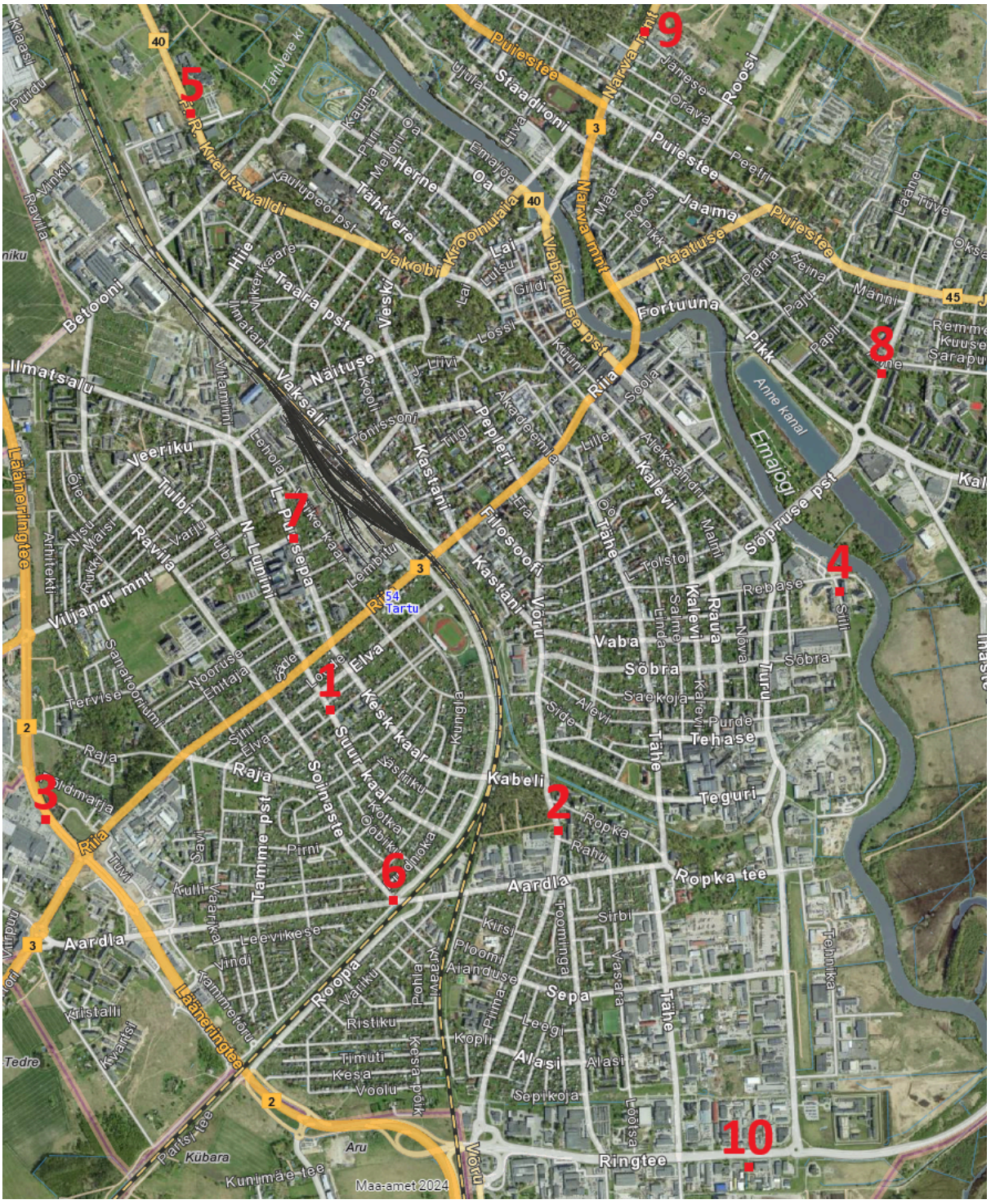


Figure 27: Different starting points in Tartu for RACE to Delta Centre time trial. The number next to each point is the number of a test.

6.4.3 Results

Table 6 shows the results for all the participants.

Table 6: Time trial results

Test nr/Tester	Tester 1	Tester 2	Tester 3	Tester 4	Tester 5
1.	2:50	**1:48**	3:10	3:30	2:55
2.	5:55	4:40	3:50	**1:44**	4:02
3.	7:25	3:50	3:37	**2:00**	2:44
4.	2:45	**1:23**	1:40	1:24	1:32
5.	2:55	2:30	2:00	**1:51**	3:17
6.	3:00	3:55	2:45	**2:06**	3:46
7.	**2:40**	4:11	3:26	3:08	4:41
8.	6:30	1:25	2:50	**1:12**	3:33
9.	1:45	0:30	0:37	**0:25**	0:31
10.	3:40	**2:00**	3:40	2:51	4:10

Explanations for Table 6:

- Test nr column shows the number of the test
- Tester column shows the number of a tester
- Times in bold are the fastest times for the corresponding tests

Visualisation of the results can be seen in Figure 28.

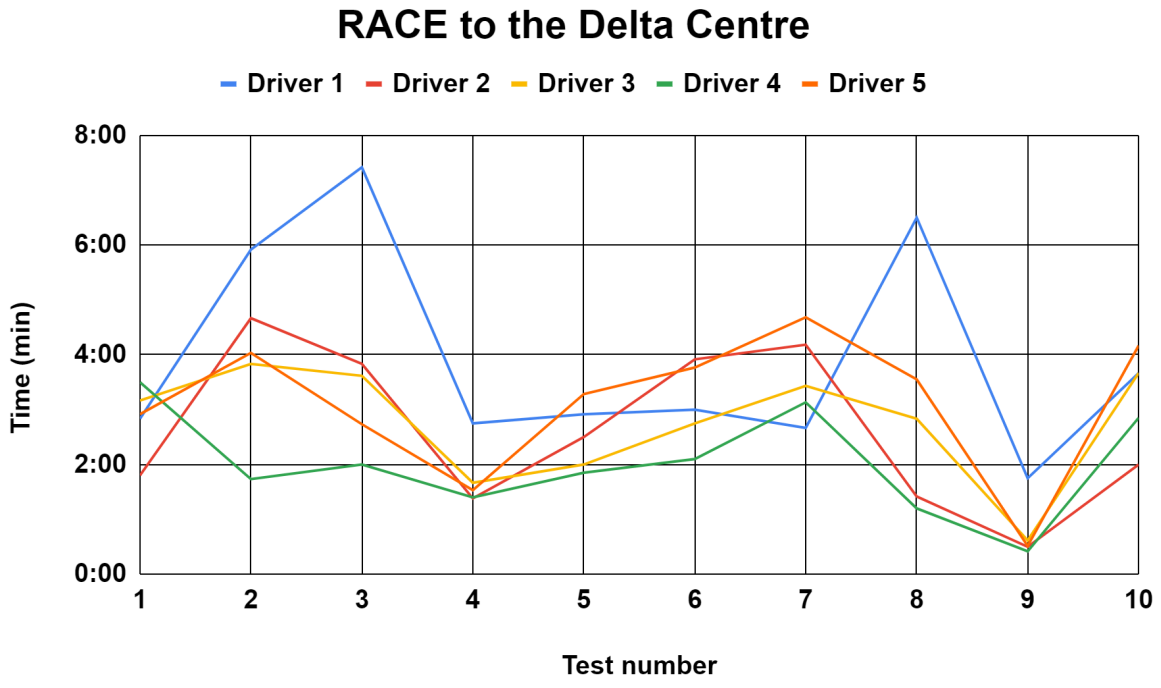


Figure 28: Visualisation of time trial results to compare drivers against each other easily.

6.4.4 Discussion

The analysis of participant performance in the navigation test within the CARLA simulator, as detailed in Table 6 and illustrated in Figure 27, reveals a relatively narrow time variance among the participants, except for Tester 1. This individual, who neither held a driver's license nor engaged regularly in gaming, emerged as an outlier in the test results. Despite occasionally demonstrating quicker reorientation abilities than his peers, his reduced driving speed within the simulator impacted Tester 1's overall travel times to the Delta Centre. This observation suggests that the interplay between real-life driving skills and gaming experience may significantly influence one's ability to navigate efficiently in a simulated environment. However, the exact contribution of each factor to Tester 1's performance remains unclear, pointing to the complexity of translating real-world skills and knowledge into a virtual context.

Tester 4's performance warrants special attention. Notably, he recorded the fastest times in most tests, adopting an innovative approach to the instruction that adherence to traffic laws was unnecessary. By exploiting the simulator's environment creatively, Tester 4 frequently sought shortcuts through unconventional routes, including train tracks, parks, and even the Emajõgi River's surface. While effective in minimising travel time, this strategic use of game mechanics highlights the need for regulatory considerations in future tests to ensure a balanced and realistic assessment of navigational strategies.

Moreover, Tester 4's approach to planning routes to the Delta Centre sometimes resulted in sub-optimal decisions, indicating that the pursuit of shortcuts did not consistently yield the intended time-saving benefits. This outcome underscores the importance of strategic route planning and the potential pitfalls of overly aggressive attempts to exploit the simulation environment.

These findings illuminate the diverse strategies and skill sets participants bring to simulated navigation tasks. They also emphasise the significance of balancing the realism of the virtual environment with the need to establish clear guidelines and limitations for test scenarios.

The feedback from testers regarding their re-orientation experience within the virtual Tartu City offered valuable insights into the digital twin's navigational aids and potential areas for improvement. Here's a summary of their general comments:

- City Centre Quality: The high-quality representation of the city centre was universally praised, significantly aiding in easy navigation and providing a positive initial impression of the virtual environment.
- Lack of Traffic Infrastructure: The absence of detailed traffic infrastructure, particularly traffic lights, in more prominent streets made these areas less recognisable and potentially hindered realistic navigation.
- Generic Building Textures: Testers noted that generic buildings, such as supermarkets represented by untextured grey squares, would serve as better orientation points if textured, suggesting that more detailed textures could improve recognisability.
- Importance of Textured Areas: Texturing, in general, was found to aid significantly in identifying locations by providing recognisable patterns, enhancing the ability to self-locate within the city.
- Missing Recognisable Junctions: Specific traffic features, like the Riia-Vaksali traffic junction with tunnels, were missed.
- Excessive Vegetation: Some areas were noted to have an overabundance of trees.

- Distinguishable Building Contours: Testers observed that some larger buildings with unique contours were easily recognisable without needing detailed textures, indicating that shape and silhouette can be sufficient for identification.
- Residential Textures and Orientation: Textures on private houses in residential areas were deemed less helpful for orientation, suggesting that focus on texturing might be better placed on more distinct and commonly known landmarks or buildings.

The general observations from the navigation and orientation test within the digital twin of Tartu offer insightful reflections on the methodology and the simulation's design. These observations suggest avenues for refining the testing approach and enhancing the virtual environment's realism and utility:

- Re-orientation vs. Route Planning: It may be more effective to focus the test on the initial re-orientation phase rather than the complete journey to a destination. This adjustment would better assess the digital twin's recognisability and navigational aids without conflating results with individuals' route planning skills or gaming proficiency. Such a focus emphasises the simulation's accuracy in replicating real-world landmarks and layouts.
- Use of Landmarks for Orientation: The Tartu TV mast's frequent use as a navigational beacon underscores the importance of incorporating prominent landmarks into the digital twin. These features greatly aid orientation, suggesting that other significant landmarks should be similarly emphasised or included to enhance navigability.
- Enhancing Recognisability of Major Streets: Streets with high traffic volumes, such as Riga Street, should be more distinguishable within the simulation. Adding specific traffic infrastructure, like traffic lights, could improve their recognisability, reflecting their significance within the city's real-world navigational framework.
- Impact of Gaming Proficiency: The test highlighted the influence of gaming experience and familiarity with the control setup on participants' movement speed within the simulation. This observation suggests the need for a standardised orientation or training session on the simulator controls to ensure that navigation tests measure spatial awareness and orientation skills rather than gaming ability.
- Setting Clear Test Regulations: The necessity of establishing comprehensive test guidelines beforehand is evident, mainly to prevent unrealistic behaviours such as driving on water. Clear rules will ensure that the test outcomes reflect realistic navigation and orientation capabilities, contributing to a more accurate evaluation of the digital twin's effectiveness as a navigational tool.

These observations provide valuable feedback for future iterations of navigation and orientation tests within digital Tartu. Addressing these points ensures that the virtual environment accurately mirrors the real-world counterpart and effectively serves its intended purpose.

6.4.5 Conclusion

The Race to the Delta Centre time trial, conducted within the digital twin of Tartu City via the CARLA simulator, has provided invaluable insights into the navigational intuitiveness and realism of the virtual environment. Through the participation of individuals with varying degrees of familiarity with Tartu's geography, driving experience, and gaming proficiency, the study has highlighted the critical aspects of virtual navigation and orientation.

The findings from the test underscore the importance of a realistic, high-fidelity representation of urban environments in digital twins for effective navigation. The detailed feedback from participants pinpointed areas of excellence, such as the accurate depiction of Tartu City Centre, and identified opportunities for improvement, including the enhancement of traffic infrastructure and the texturing of generic buildings. These observations are instrumental in refining the digital twin to serve its users better, whether for entertainment, educational purposes (driving schools), or urban planning simulations.

Moreover, the experiment demonstrated the complex interplay between real-world knowledge, gaming skills, and adaptive problem-solving in virtual environments. Tester 4's inventive navigation strategies, while highlighting the potential for creative engagement with digital twins, also emphasised the need for clear guidelines to ensure that such simulations remain realistic and grounded in the physical world's constraints.

6.5 Measuring the Simulation Performance Using TELECARLA

6.5.1 Introduction

The primary objective of this test was to evaluate the performance of the digital twin of Tartu City by comparing it against other simulated environments. This comparison aimed to highlight the strengths and potential areas for improvement in the Tartu simulation, particularly in computational efficiency. TELECARLA [27], a specialised software framework, was employed to facilitate this assessment. TELECARLA is an extension of the CARLA simulator for teleoperated driving. It uses GStreamer[25] for the compression and transmission of camera data. ROS[26] (Robot Operating System) acts as an interface between its framework and CARLA. It was selected as the preferred tool for driving simulations within the CARLA simulator due to its alignment with the ADL's real-life teleoperation methodologies. Teleoperation, the remote control of vehicles, is a critical component of ADL's approach to autonomous vehicle research and development, providing a hands-on method to navigate vehicles from a distance, whether for testing or operational purposes. Opting to pilot a car within the CARLA simulator using TELECARLA instead of directly manipulating the vehicle without the ROS bridge enhances the simulation's realism and puts the hardware under a more substantial load.

6.5.2 Method

In this comprehensive test, three distinct maps were evaluated to assess their performance, particularly in terms of Frames Per Second (FPS), a critical measure of rendering efficiency and visual fluidity in simulations. The maps included "Town 10," a standard environment from the original CARLA package serving as the benchmark, and two maps representing different scales of Tartu: the "Tartu demo" map, focusing on the city centre, and the "Tartu large" map, which encompasses the entire city and represents the culmination of this master's

[25] https://gstreamer.freedesktop.org/
[26] https://www.ros.org/

thesis project. The testing framework was designed to cover various environmental conditions and graphical settings, ensuring a thorough assessment of the digital twin's adaptability and rendering efficiency. By conducting each test in both Low and Epic quality levels[27], the evaluation sought to understand how different graphical fidelity settings impact the performance, specifically regarding FPS, which is crucial for maintaining a smooth simulation experience.

Including day and night cycles, clear sky, and rainy weather conditions further diversified the testing scenarios. These variations are significant as they not only affect the visual appearance of the simulation but also introduce different rendering challenges. Nighttime and rainy conditions, for example, require the simulation to render additional lighting effects, reflections, and potentially particle effects for rain, which can be more demanding on the system's resources.

Therefore, each scenario underwent 8 test runs, encompassing the following combinations:

1. Low quality during daytime with clear sky
2. Low quality during daytime with rain
3. Low quality at night with clear sky
4. Low quality at night with rain
5. Epic quality during daytime with clear sky
6. Epic quality during daytime with rain
7. Epic quality at night with clear sky
8. Epic quality at night with rain

The decision to disable traffic simulation across all tests was strategic, aimed at isolating the impact of environmental rendering on performance without the additional computational load introduced by simulating dynamic traffic patterns. This approach ensures that the FPS measurements accurately reflect the efficiency of rendering the static environment, providing a more straightforward comparison between the maps.

The first test was for Town 10. TELECARLA was used to manually drive the vehicle, using the Logitech G920 steering wheel, around the block for one complete lap. This route provided a baseline for performance metrics within a standard CARLA environment.

For the Tartu demo and Tartu large maps, the evaluation process was designed to automate vehicle navigation, thereby standardising the test conditions across these environments. By spawning the vehicle at the Delta Centre and activating the monitoring mode, the vehicle autonomously traversed the map at a constant speed. This method ensured consistency in vehicle behaviour and objectively compared performance metrics such as Frames Per Second (FPS) across different areas of the digital twin. The monitoring mode, which relies on the Open Drive map lanes for navigation, allows the vehicle to follow a predetermined path that exemplifies a typical journey within the city's digital representation. This path, illustrated in Figure 29, was chosen to compare the performance of the Tartu demo and Tartu large maps. The vehicle's journey around the whole lap concluded at the Delta Centre, where it had initially started.

[27] https://carla.readthedocs.io/en/0.9.13/adv_rendering_options/#quality-levels

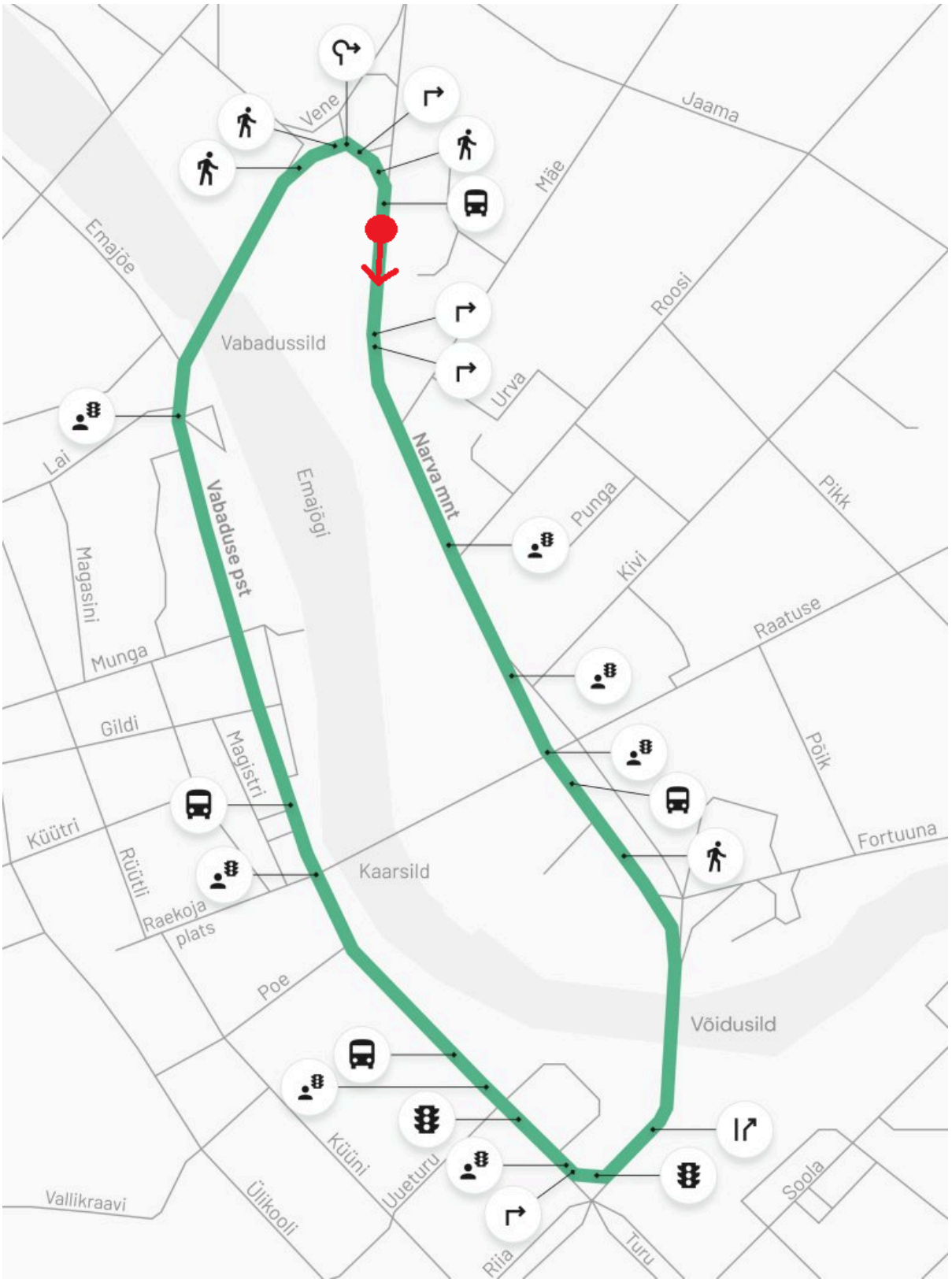


Figure 29: The starting point for scenario number one is marked with a red dot on the map. The arrow points in the direction the car starts moving, and the green line is the vehicle's path during the test.

The final test focused exclusively on the Tartu large map, utilising two distinct versions to assess the impact of texturing on performance. The first version of the map was devoid of

textures generated by drone photography, presenting a more basic visual representation of the environment. The second version included these textures, offering a detailed and visually rich depiction of the area. This comparative approach aimed to quantify the performance differences between a textured and a non-textured environment within the same urban landscape.

The test commenced at the Aardla-Soinaste-Raudtee intersection, a starting point that allowed for a comprehensive evaluation of the vehicle's journey through various parts of the city towards Turu Street. With the monitoring mode activated, the vehicle autonomously navigated the predetermined route, adhering to the lanes defined in the OpenDrive map. This automated driving mode ensured vehicle speed and behaviour consistency, providing a controlled environment for measuring performance metrics.

The path taken by the vehicle, detailed in Figure 30, was designed to traverse diverse urban settings, offering insights into how different areas of the map, with varying levels of detail and complexity, affected the simulation's performance. The conclusion of the test upon the vehicle's arrival at Turu Street marked the end of the assessment, enabling a direct comparison of the two map versions under identical conditions.

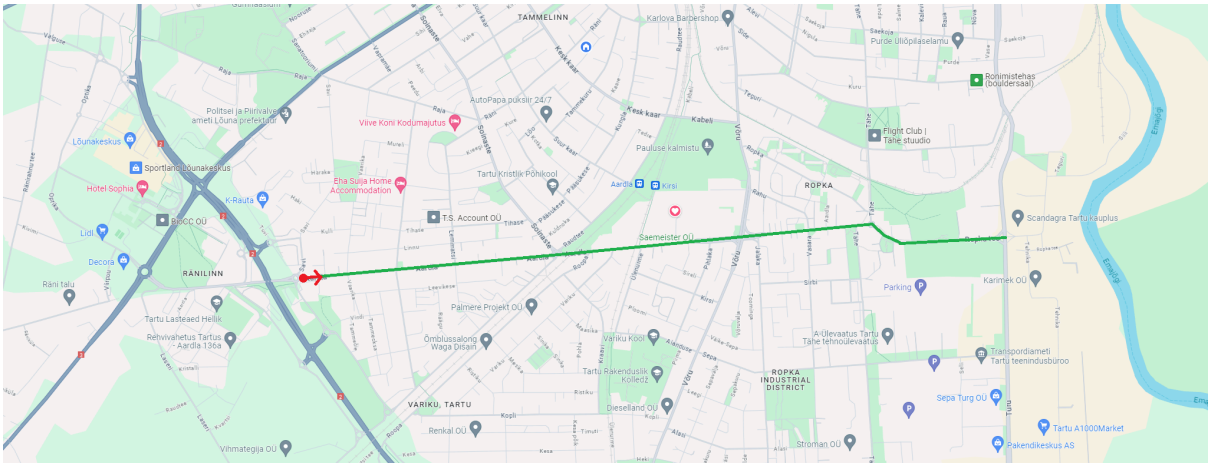


Figure 30: The starting point for scenario number two is marked with a red dot on the map. The arrow points in the direction the car starts moving, and the green line is the vehicle's path during the test.

The primary metric for evaluating performance in this test was the CARLA rendered FPS. This metric was monitored through the teleoperation client's graphical user interface (GUI), displayed explicitly in the top left corner, providing real-time feedback on the simulation's performance. The experimenter used their judgment to estimate an average FPS value. The FPS rate displayed by the server to TELECARLA was relatively stable throughout the test, so determining an average value was straightforward. While somewhat subjective, this approach allowed for a practical assessment of the simulation's performance under different conditions, including textured versus non-textured environments.

The computer used for testing had these specifications:

CPU: Intel Core i7-11700F 2.5G 8c

GPU: 8GB GB RTX3070 Gaming OC LHR

RAM: 64 GB 3200 MHZ

6.5.3 Results

Table 7 shows results from running the lap in Tartu City centre for the Tartu demo and Tartu large maps. It also includes the results from the Town 10 run. All the numeric values in the table are FPS scores.

Table 7: Demo Lap Results

Environment:	**Town 10**		**Tartu demo**		**Tartu large**	
Quality:	**Low**	**Epic**	**Low**	**Epic**	**Low**	**Epic**
Day	31	23	36	35	30	30
Rainy day	31	22	35	34	30	28
Night	28	15	33	22	24	17
Rain at night	28	15	33	21	24	15
Average	29.5	18.75	34.25	28	27	22.5

Table 8 shows results from running the texture test on Aardla Street. All the numeric values in the table are FPS scores.

Table 8: Aardla Street Results

Environment:	**Tartu large (no texture)**		**Tartu large**	
Quality:	**Low**	**Epic**	**Low**	**Epic**
Day	37	36	33	33
Rainy day	37	34	33	32
Night	30	27	29	25
Rain at night	29	26	29	24
Average	33.25	30.75	31	28.5

6.5.4 Discussion

The performance metrics presented in Table 7 offer insightful comparisons between the Tartu large map and the smaller-scale Tartu demo and Town 10 maps within the CARLA simulator. Despite the significantly larger size and potentially greater complexity of the Tartu large map, its performance remained competitive, challenging initial expectations about the correlation between map size, complexity, and rendering efficiency.

Key Observations:

- Unexpected Performance in Epic Quality Mode: The relatively low FPS observed for Town 10 in Epic quality mode was surprising, especially considering its smaller size and lower polygon count than the Tartu maps. This outcome suggests that factors other than map size and polygon count, such as texture resolutions, the number of dynamic objects, or specific rendering optimisations, may significantly impact performance in high-quality settings.
- Minimal Impact of Quality Settings on Tartu Large: The observation that reducing the quality from Epic to Low had little effect on the FPS for the Tartu large map, particularly during daytime, suggests that the map's performance bottlenecks may not be directly related to the factors adjusted by these quality settings. However, the noticeable improvement in FPS at night when reducing quality levels, attributed to the

disabling of shadows, indicates that lighting effects, especially those generated by streetlights, significantly impact performance.

- Rain Effects: The minimal impact of activating rain on FPS across all maps was notable regardless of the time of day. This suggests that the simulation's handling of weather effects, such as rain, is optimised to minimise additional computational strain. The rain effects, including changes to road textures and the addition of raindrops, are likely implemented so they do not significantly increase the rendering workload.

These results show that it's not easy to guess how well virtual environments will perform just by looking at their size and detail. It's important to think about many different things, such as how they're made, how the environment changes, and how moving parts are improved.

Table 8's comparison between the textured and untextured versions of the Tartu large map reveals a relatively minor difference in performance, with the textured version being only 7.32% slower than its untextured counterpart when running in Epic quality mode. This marginal performance decrease highlights the efficiency of the simulation's handling of textures and the effectiveness of optimisation techniques used in developing the digital twin.

The slight reduction in FPS associated with adding textures is a small trade-off for the significant benefits textures bring to the simulation. Textures play a crucial role in enhancing the realism and immersive quality of the virtual environment, making it more visually appealing and lifelike. Furthermore, textures can significantly aid navigation and orientation within the digital twin. They provide users with familiar visual cues that mirror the real world, making it easier for individuals to recognise specific locations, landmarks, and routes.

The data from Table 8 underscores the successful integration of detailed textures into the Tartu large map without significantly compromising the simulation's performance. This balance between visual quality and efficiency is essential for creating practical and usable digital twins. It ensures that users can benefit from a richly detailed and realistic virtual environment without experiencing detrimental impacts on smoothness or responsiveness.

6.5.5 Conclusion

In conclusion, the performance testing of Tartu City's digital twin has not only affirmed its viability as a tool for simulation and research but has also shed light on the complicated balance between visual fidelity and computational efficiency.

6.6 Benchmarking Performance

6.6.1 Introduction

As with TELECARLA, the primary objective of this test was to evaluate the performance of the digital twin of Tartu City by comparing it against other simulated environments. This test was using CARLA simulator's own benchmarking script[28] called *performance_benchmark.py*. It enables users to analyse the performance of CARLA quickly in their environment. The script can be configured to run several scenarios that combine different maps, sensors and weather conditions. It reports the average and standard deviation of FPS under the requested scenarios.

6.6.2 Method

The script was run with the default setup, besides enabling the rendering mode. The maps tested with TELECARLA (Town 10, Tartu demo, Tartu large) were also used here.

[28] https://carla.readthedocs.io/en/latest/adv_benchmarking/

The test used seven sensor setups:

1. Low resolution (300x200)
2. Medium resolution (800x600)
3. High resolution (1900x1080)
4. Dual low-resolution cameras (300x200, replicated setup)
5. 100k points (representing a lower-density LiDAR scan)
6. 500k points (a mid-range density)
7. 1M points (a high-density scan)

And three different weather profiles:

1. Clear Noon (optimal visibility conditions)
2. Cloudy Noon (reduced visibility)
3. Soft Rain Sunset (challenging visibility and additional rendering for rain effects)

The same computer setup used with TELECARLA ensured consistency in hardware performance across all tests, allowing for direct comparisons of software-induced performance variations.

6.6.3 Results

Table 9 shows the performance results from running performance_benchmark.py and only using the --render_mode flag, besides selecting the map.

Table 9: performance_benchmark.py results

Test nr/Sensor	Sensor	Weather	Town 10 Mean FPS	Tartu demo Mean FPS	Tartu large Mean FPS
Test 1	cam-300x200	Clear Noon	51.23	49.02	39.64
Test 2	cam-800x600	Clear Noon	27.96	13.56	20.87
Test 3	cam-1900x1080	Clear Noon	17.92	15.48	12.59
Test 4	cam-300x200 cam-300x200	Clear Noon	14.46	8.56	10.22
Test 5	LIDAR: 100k	Clear Noon	351.78	323.49	117.14
Test 6	LIDAR: 500k	Clear Noon	347.11	306.11	114.31
Test 7	LIDAR: 1M	Clear Noon	328.90	296.34	108.18
Test 8	cam-300x200	Cloudy Noon	51.20	51.13	39.61
Test 9	cam-800x600	Cloudy Noon	27.91	26.92	20.96
Test 10	cam-1900x1080	Cloudy Noon	17.89	17.06	12.59
Test 11	cam-300x200 cam-300x200	Cloudy Noon	14.49	12.30	10.20
Test 12	LIDAR: 100k	Cloudy Noon	361.40	328.99	119.25
Test 13	LIDAR: 500k	Cloudy Noon	349.81	306.06	113.62
Test 14	LIDAR: 1M	Cloudy Noon	329.74	298.64	107.47
Test 15	cam-300x200	Soft Rain Sunset	45.57	39.28	32.77
Test 16	cam-800x600	Soft Rain Sunset	25.78	22.48	18.08
Test 17	cam-1900x1080	Soft Rain Sunset	16.91	14.70	11.22

Test 18	cam-300x200 cam-300x200	Soft Rain Sunset	13.71	10.61	9.01
Test 19	LIDAR: 100k	Soft Rain Sunset	358.22	324.01	118.97
Test 20	LIDAR: 500k	Soft Rain Sunset	353.07	312.92	112.92
Test 21	LIDAR: 1M	Soft Rain Sunset	330.69	298.11	107.64

The Figure 31 shows the visualisation of Table 9 results.

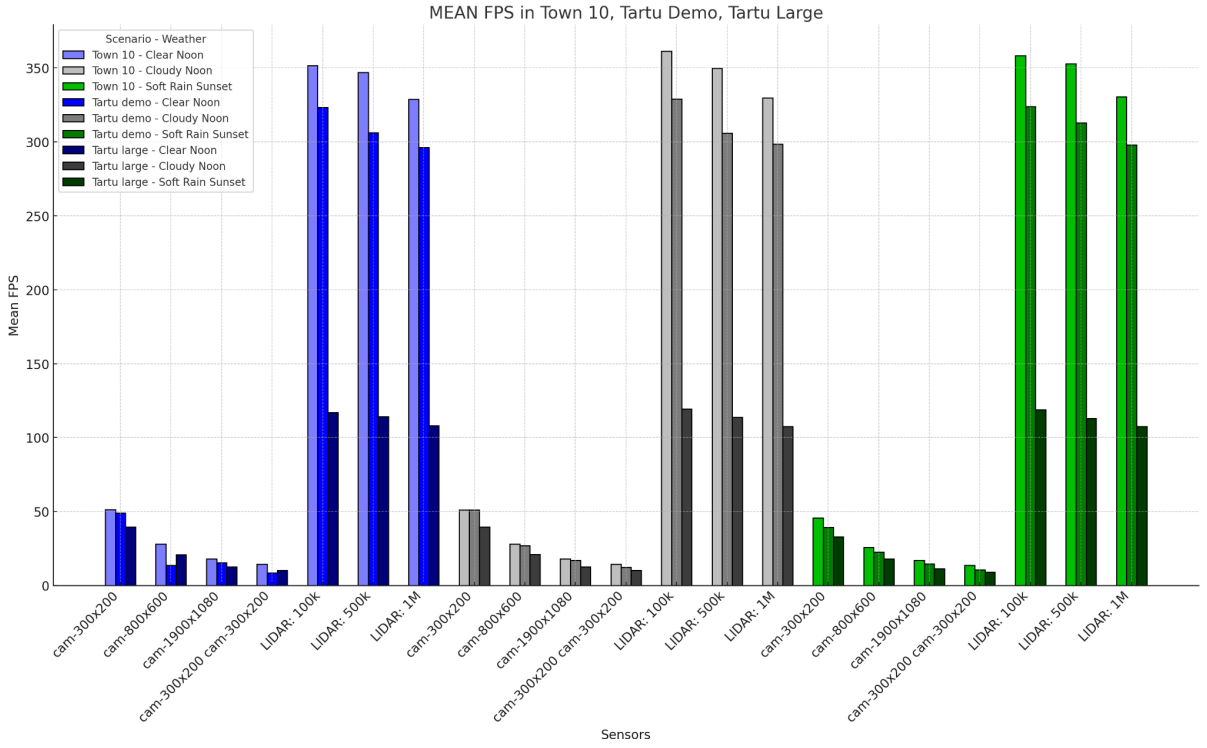


Figure 31: Results are visualised as a bar chart from running performance_benchmark.py script.

6.6.4 Discussion,

The analysis of simulation performance, using visual cameras and LiDAR sensors across three maps—Town 10, Tartu demo, and Tartu large—reveals insights into how different data acquisition methods impact simulation efficiency, as depicted in Figure 30.

Visual Cameras Performance:

- The relatively close performance results for all three maps under various camera and weather conditions indicate a high optimisation level within the CARLA simulation environment. It demonstrates the capability of the simulation to maintain consistent rendering performance, even as variables such as camera resolution and environmental effects are altered.
- A significant shift was observed in the performance rankings between the TELECARLA experiment and this test. Town 10, which had previously recorded the lowest mean fps in the TELECARLA experiment, emerged with the best mean fps across every test scenario. This reversal highlights how different testing setups, camera configurations, and weather conditions can dramatically influence a map's performance, underscoring the complexity of simulation environments and the impact of rendering technologies.

- In scenarios utilising clear noon weather with visual cameras, Tartu large unexpectedly outperformed the Tartu demo map. This was surprising given the larger scale and presumably higher complexity of the Tartu large map. Such an outcome suggests that implementing and optimising textures and other environmental and graphical elements in Tartu large may have been more efficiently handled than in the Tartu demo map.

LiDAR Sensors Performance:

- The implementation of LiDAR sensors introduced a significant differentiation in performance, particularly affecting the Tartu large map. While Town 10 maintained the highest mean fps across all tests, Tartu large experienced a marked decrease in performance when LiDAR was used, showing a 2.5 times lower mean fps compared to the Tartu demo.
- This performance disparity with LiDAR can be attributed to Tartu large's expansive environment, which presents more surfaces for LiDAR beams to interact with. The increase in surface interactions necessitates more complex calculations by the physics engine, accounting for the bounce and reflection of LiDAR beams, which impacts the simulation's rendering efficiency.
- These observations highlight the impact of simulation scale and sensor type on performance. While visual rendering appears uniformly efficient across different maps, LiDAR sensors introduce significant performance challenges, particularly in larger, more detailed environments like Tartu large.

6.6.5 Conclusion

The performance benchmark tests conducted on the digital twin of Tartu City provide valuable insights into the factors influencing simulation efficiency within the CARLA environment. Key findings from this analysis include:

- Performance Consistency with Visual Cameras: The simulation demonstrates a high optimisation level for rendering with visual cameras. Consistent performance was maintained across maps, camera resolutions, and weather conditions.
- Map-Specific Performance Shifts: Unexpected variations in performance rankings between the TELECARLA experiment and these tests emphasise the complex interplay of simulation elements. Factors like rendering techniques, map features, and testing setups can significantly alter a map's performance profile.
- LiDAR Performance in Large Areas: LiDAR sensors introduced a notable performance loss in more extensive and complex environments than in smaller areas.

7. Conclusion

The digital twin creation for Tartu City was an ambitious endeavour that required a diligent approach to accurately replicate the city's infrastructure, including roads, buildings, vegetation, and other critical components. At the heart of this process was a commitment to automation and precision, leveraging a combination of high-resolution drone photography, sophisticated software tools, and custom scripting to streamline the development process and enhance the twin's realism.

One of the thesis's key methodological highlights is the automated generation of the road network and terrain. Utilising open data and advanced mapping techniques, the project automatically converted geospatial information into a detailed and navigable road network within the digital twin. This automation extended to terrain formation, where elevation data from digital terrain models were applied to accurately represent Tartu City's topography. This level of detail was crucial for simulating driving conditions that AVs would encounter in the actual city, providing a robust foundation for testing and development.

The texturing process represented another significant methodological advancement. High-resolution drone photographs were used to map textures onto the 3D models of roads, buildings, and the ground, giving the digital twin a visually rich and authentic appearance. This process involved sophisticated image processing techniques to ensure that textures were accurately aligned with their corresponding models, thereby enhancing the virtual environment's overall realism.

Incorporating urban infrastructure into the digital twin—such as streetlights, trash containers, and bus stops—was achieved through scripting and data integration, automating the placement of these elements based on geospatial data. Similarly, the integration of vegetation was refined through filtering processes, ensuring that the digital twin's green spaces accurately reflect their real-world counterparts in terms of density and distribution. These automated methodologies for adding infrastructure and vegetation underscore the digital twin's attention to detail, significantly enhancing its effectiveness as a simulation environment for AVs.

Furthermore, creating a High-Definition (HD) map in OpenDrive format was essential for integrating autonomous driving functionalities. It simulates realistic driving scenarios, tests the AV's responses to various traffic conditions, and ensures its compatibility with real-world navigation systems.

The user experience test, the "Race to the Delta Centre" time trial, offered valuable insights into the digital twin's navigational intuitiveness and realism. Participants' ability to re-orient themselves and navigate through the virtual Tartu City highlighted the simulation's success in accurately replicating the city's layout and landmarks. The feedback emphasised the significance of detailed texturing and the strategic placement of recognisable urban features in enhancing spatial awareness and navigation.

In conclusion, this master thesis on developing a digital twin of Tartu City within the CARLA simulator has successfully demonstrated the integration of comprehensive urban data, automated road and terrain generation processes, and advanced texturing techniques to create a realistic virtual environment. This project enhances the testing capabilities for autonomous driving technologies and offers insights into the potential of digital twins in urban planning and transportation research. Through methodological rigour, performance benchmarking, and user experience evaluation, the thesis underscores the importance of digital twins in bridging

the gap between theoretical models and practical applications in the evolving landscape of autonomous mobility.

8. Future Work

The completion of this project marks not the end but rather a milestone in the journey of the Autonomous Driving Lab at the University of Tartu, with ambitions to integrate Tartu City's digital twin into various further projects. This necessitates several enhancements, given that the current project is a conceptual framework or prototype rather than a comprehensive solution. It has been an informative experience, revealing the current capabilities and the potential for future developments. Notably, this project has laid the groundwork for the possibility of creating digital twins for all major cities in Estonia, using the established workflow as a blueprint. This endeavour has not only tested our current understanding but also expanded our vision of what can be achieved in the realm of digital twinning.

The initial steps towards enhancing the project involve applying the texture method, which was successfully trialled south of Aardla Street across the entire city. This approach received highly positive feedback from users, and performance evaluations confirmed it would not significantly impact the simulation's efficiency. The method notably enhances the visual immersion, creating a remarkably vivid experience. Practically, it addresses many road marking issues, correcting those that are missing or inaccurately laid out. However, there are challenges with texture mapping, as highlighted in the chapter "Results from Using Drone Photos for Texture Mapping." The issue lies in the alignment; not everything matches perfectly when the textures are stitched together. Potential solutions include experimenting with various Agisoft Metashape techniques or using alternative photogrammetry software to refine the mapping process.

A crucial improvement to the project would be integrating traffic lights on major roads throughout Tartu. The absence of traffic lights was a primary concern among testers, leading to difficulties in navigation. Currently, traffic light data is not publicly accessible, but there is optimism that this information will become available in the near future, enabling a more realistic and functional simulation environment.

Incorporating additional elements into the cityscape, it's evident that the Estonian National Defence College building on Riia-Hill is absent. Since the Estonian Land Board does not supply the model directly, a viable and effective strategy would be constructing this 3D model utilising photogrammetry. This method allows for the creation of detailed and accurate representations of structures, thereby enhancing the overall realism and completeness of the digital twin.

Transitioning to a novel methodology for enriching the scene, leveraging machine learning algorithms in combination with orthophotos presents an interesting approach to automatically populate the city with various objects such as parked cars, benches, and garden fences. Including these items would significantly contribute to the authenticity and vibrancy of the urban environment, making it feel more alive and dynamic.

Several strategies are available for optimisation, primarily focusing on adopting the configuration recommended by CARLA for managing large[29] maps. While it has limitations, this approach promises to improve frame rates. It involves dynamically enabling and disabling traffic cars within a certain range, optimising performance without compromising the simulation's dynamic and complex traffic patterns. This could potentially allow for the inclusion of a significant number of traffic participants, making the simulation even more realistic and engaging.

[29] https://carla.readthedocs.io/en/latest/large_map_overview/

Another good strategy would be to use hierarchical instanced static meshes (HISM) in as many places as possible. Implementing HISM across the simulation could significantly increase its performance and visual fidelity. By utilising HISM technology, adding repetitive yet essential elements like 3D rails for train tracks, garden fences, and 3D grass becomes remarkably efficient in terms of performance. This approach capitalises on the fact that these common objects in the scene share the same base mesh. Leveraging HISM allows mass replicating these objects without the typical performance penalties of rendering multiple instances of complex models. This method not only enhances the detailed realism of the environment but also optimises the simulation's overall performance, enabling the inclusion of diverse and intricate details that contribute to a more immersive and lifelike city experience.

In refining the texturing process, it's crucial to conduct further testing to determine the optimal number of textures per city block that balances visual quality with performance efficiency. Presently, areas south of Aardla Street have been detailed using a hundred 4K textures. There's potential to reduce this number without significantly compromising visual fidelity. Finding the sweet spot would streamline the rendering process and enhance the simulation's performance, allowing for smoother operation and potentially enabling the simulation to run on a broader range of hardware. This optimisation could make the digital twin more accessible and improve the user experience while maintaining high realism and aesthetic appeal.

Given the project's significant artistic component, it inherently lacks a definitive conclusion. Artistic endeavours within such projects are perpetually evolving, reflecting that there is always room for enhancement and refinement. Even when all essential functionalities have been incorporated, the project's creative aspect offers improvement opportunities. This ongoing process underscores the dynamic nature of art in technology, where innovations and artistic excellence perpetually push the boundaries of what can be achieved.

9. Acknowledgements

I would like to acknowledge the assistance of ChatGPT, an AI-based tool developed by OpenAI, for its invaluable help in editing and refining the text of this thesis. The use of this advanced technology has significantly contributed to the clarity and coherence of the presented material.

10. References

[1]	Critical reasons for crashes investigated in the National Motor Vehicle Crash Causation Survey. (Traffic Safety Facts Crash Stats. Report No. DOT HS 812 506), *Singh, S.*, 2018 https://crashstats.nhtsa.dot.gov/Api/Public/Publication/812506 (07.03.2024)
[2]	Preparing a nation for autonomous vehicles: opportunities, barriers and policy recommendations, *Fagnant, D. J., Kockelman*, K., 2015 https://www.researchgate.net/publication/277025982_Preparing_a_nation_for_autonomous_vehicles_Opportunities_barriers_and_policy_recommendations
[3]	How Simulation Helps Autonomous Driving: A Survey of Sim2real, Digital Twins, and Parallel Intelligence, *Hu, X., Li, S., Huang, T., Tang, B., Huai, R., Chen, L.*, 2023 https://ieeexplore.ieee.org/abstract/document/10242366 (01.03.2024)
[4]	Choose Your Simulator Wisely: A Review on Open-source Simulators for Autonomous Driving, *Li, Y., Yuan, W., Zhang, S., Yan, W., Shen, Q., Wang, C, Yang, M.*, 2023 https://arxiv.org/abs/2311.11056 (01.03.2024)
[5]	About — blender.org, *Blender Foundation* https://www.blender.org/about/ (20.02.2024)
[6]	What is Blender, and Is It Right for You?, *School of Motion* https://www.schoolofmotion.com/blog/what-is-blender-and-is-it-right-for-you (20.02.2024)
[7]	What is Blender Used For?, *University of Silicon Valley,* 2022 https://www.usv.edu/blog/what-is-blender-used-for (20.02.2024)
[8]	RoadRunner, *Mathworks* https://uk.mathworks.com/help/roadrunner/index.html (20.02.2024)
[9]	CARLA, *CARLA Simulator Official Website* https://carla.org (20.02.2024)
[10]	What is CARLA and How to Use it To Simulate Autonomous Driving?, *Rocketloop*, 2023 https://rocketloop.de/en/blog/carla-autonomous-driving-simulator/ (20.02.2024)
[11]	18. Features, *Unreal Engine* https://www.unrealengine.com/en-US/features (20.02.2024)
[12]	A Gentle Introduction to GIS, *QGIS Documentation* https://docs.qgis.org/3.28/en/docs/gentle_gis_introduction/index.html (20.02.2024)
[13]	QGIS: An Introduction to an Open-Source Geographic Information System, *Mississippi State University Extension Service,* 2021

	https://extension.msstate.edu/publications/qgis-introduction-open-source-geographic-information-system (20.02.2024)
[14]	10 Advantages of QGIS Software, *Grind GIS* https://grindgis.com/software/10-advantages-of-qgis-software (20.02.2024)
[15]	Use Agisoft Metashape to create 3D models for Dynamics 365 Guides and for mixed-reality components included in apps created with Power Apps, *Microsoft Learn* https://learn.microsoft.com/en-us/dynamics365/mixed-reality/guides/3d-content-guidelines/agisoft-metashape (20.02.2024)
[16]	Using Agisoft Metashape, *University of North Carolina at Chapel Hill* https://guides.lib.unc.edu/metashape (20.02.2024)
[17]	Discover intelligent photogrammetry with Metashape, *Agisoft* https://www.agisoft.com/ (20.02.2024)
[18]	Elevation Data, *Estonian Land Board* https://geoportaal.maaamet.ee/eng/Spatial-Data/Elevation-Data-p308.html (19.03.2024)
[19]	Orthophotos, *Estonian Land Board* https://geoportaal.maaamet.ee/eng/Spatial-Data/Orthophotos-p309.html (19.03.2024)
[20]	Orthophoto Map Sheets, *Estonian Land Board* https://geoportaal.maaamet.ee/eng/Spatial-Data/Orthophotos/Orthophoto-Map-Sheets-p349.html (19.03.2024)
[21]	3D Data, *Estonian Land Board* https://geoportaal.maaamet.ee/eng/Spatial-Data/Geo3D/3D-Data-p836.html (19.03.2024)
[22]	Simulation of Tartu City Centre for Testing Autonomous Vehicles, *Mitt A.,* 2021 https://hdl.handle.net/10062/92171 (19.03.2024)
[23]	Geo3D Ärianalüüs, *Estonian Land Board, 2023* https://geoportaal.maaamet.ee/data/files/Geo3D_arianalyys_lopparuanne.pdf (19.03.2024)
[24]	Vardhan H. HD Maps: New age maps powering autonomous vehicles, *Geospatial World*, 2017. https://www.geospatialworld.net/article/hd-maps-autonomous-vehicles/ (19.03.2024)

[25]	Setting Up Automatic LOD Generation, *Unreal Engine Documentation* https://docs.unrealengine.com/4.27/en-US/WorkingWithContent/Types/StaticMeshes/HowTo/AutomaticLODGeneration/ (19.03.2024)
[26]	Unreal* Engine 4 Optimization Tutorial, Part 2, *Intel*, 2017 https://www.intel.com/content/www/us/en/developer/articles/training/unreal-engine-4-optimization-tutorial-part-2.html (19.03.2024)
[27]	TELECARLA: An Open Source Extension of the CARLA Simulator for Teleoperated Driving Research Using Off-the-Shelf Components, *Hofbauer, M., Kuhn, C.B., Petrovic, G., Steinbach, E.*; 2020 https://www.researchgate.net/publication/341293636_TELECARLA_An_Open_Source_Extension_of_the_CARLA_Simulator_for_Teleoperated_Driving_Research_Using_Off-the-Shelf_Components (4.03.2024)

Appendix

- **CARLA Launch Guide**

According to CARLA's official webpage, CARLA is a performance-demanding software. At the very minimum, it requires a 6GB GPU or, even better, a dedicated GPU capable of running Unreal Engine.

Adding Tartu demo track and UT Lexus to CARLA

- Download Carla 0.9.13 and extract it. We will call this extracted folder <CARLA ROOT>.

- Download tartu_large.tar.gz

- Copy tartu_large.tar.gz inside the Import folder under the <CARLA ROOT> directory.

- Run ./ImportAssets.sh from the <CARLA ROOT> directory. This will install the tartu_large map. (You can now delete the tartu_large.tar.gz file from the import folder.)

- Since we often refer to <CARLA ROOT>, let's export it as an environment variable. Make sure to replace the path where Carla is downloaded.

  export CARLA_ROOT=/path/to/your/carla/installation

  Now, enter the following command. (**NOTE:** Here, we assume that CARLA_ROOT was set from the previous command.)

  export PYTHONPATH=$PYTHONPATH:${CARLA_ROOT}/PythonAPI/carla/dist/carla-0.9.13-py3.7-linux-x86_64.egg:${CARLA_ROOT}/PythonAPI/carla/agents:${CARLA_ROOT}/PythonAPI/carla

  Note: It will be convenient if the above variables are automatically exported whenever you open a terminal. Putting the above exports in ~/.bashrc will reduce the hassle of exporting every time.

Running tartu_large map with UT Lexus and traffic enabled

Enter the following commands from the command line.

Run CARLA:

- $CARLA_ROOT/CarlaUE4.sh

Select tartu_large map:

- $CARLA_ROOT/PythonAPI/util/config.py -m tartu_large

Enable Traffic:

- $CARLA_ROOT/PythonAPI/examples/generate_traffic.py

Drive with UT Lexus:

- $CARLA_ROOT/PythonAPI/examples/manual_control.py --filter utlexus

To modify the game window size open

$CARLA_ROOT/PythonAPI/examples/manual_control.py and look for “resolution”

change the default value to your liking.

Running tartu_large map with Autoware Mini and CARLA Simulator

Follow the official guide to install all the prerequisites and Autoware Mini. CARLA installation part can be ignored besides installing system dependencies:

- sudo apt install libomp5

Once the installation is done, use the following commands to launch the tartu_large map with Autoware Mini.

- $CARLA_ROOT/CarlaUE4.sh
- roslaunch autoware_mini start_carla.launch map_name:=tartu_large

License

Non-exclusive licence to reproduce thesis and make thesis public

I, **Allan Mitt**,

1. grant the University of Tartu a free permit (non-exclusive licence) to:

reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright, my thesis

Simulation of Tartu City Centre for Testing Autonomous Vehicles,

supervised by Tambet Matiisen.

2. I grant the University of Tartu the permit to make the thesis specified in point 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 4.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work from 08/03/2024 until the expiry of the term of copyright,

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I confirm that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Allan Mitt

07.03.2024