vehicles

MDPI

Article

3D Environment Generation from Sparse Inputs for Automated Driving Function Development

Till Temmen *, Jasper Debougnoux, Li Li, Björn Krautwig, Tobias Brinkmann, Markus Eisenbarth and Jakob Andert *

Teaching and Research Area Mechatronics in Mobile Propulsion, Faculty of Mechanical Engineering, RWTH Aachen University, Forckenbeckstraße 4, 52074 Aachen, Germany
* Correspondence: temmen@mmp.rwth-aachen.de (T.T.); andert_j@mmp.rwth-aachen.de (J.A.)

Abstract

The development of AI-driven automated driving functions requires vast amounts of diverse, high-quality data to ensure road safety and reliability. However, both the manual collection of real-world data and creation of 3D environments are costly, time-consuming, and hard to scale. Most automatic environment generation methods still rely heavily on manual effort, and only a few are tailored for Advanced Driver Assistance Systems (ADAS) and Automated Driving Systems (ADS) training and validation. We propose an automated generative framework that learns ground-truth features to reconstruct 3D environments from a road definition and two simple parameters for country and area type. Environment generation is structured into three modules—map-based data generation, semantic city generation, and final detailing. The overall framework is validated by training a perception network on a mixed set of real and synthetic data, validating it solely on real data, and comparing performance to assess the practical value of the environments we generated. By constructing a Pareto front over combinations of training set sizes and real-to-synthetic data ratios, we show that our synthetic data can replace up to 85% of real data without significant quality degradation. Our results demonstrate how multi-layered environment generation frameworks enable flexible and scalable data generation for perception tasks while incorporating ground-truth 3D environment data. This reduces reliance on costly field data and supports automated rapid scenario exploration for finding safety-critical edge cases.

Keywords: automated driving; synthetic data generation; 3D scene generation; generative models; AI generated content; computer vision

Check for updates

Academic Editors: Eva Michelaraki and George Yannis

Received: 30 December 2025
Revised: 17 February 2026
Accepted: 19 February 2026
Published: 2 March 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

1. Introduction

Traffic safety is one of the greatest challenges of our time, especially in the context of automated driving [1–3]. One of the fundamental issues is the substantial Operational Design Domain (ODD) within which these systems must operate. The vastness of the ODD arises partly from a significant reliance on perception functions, which must evaluate a diverse range of environments [4]. These environments vary greatly on multiple scales, including different regions, types of areas, different types of scene elements, and local interactions, such as reflective surfaces [5–7]. Consequently, the testing process becomes increasingly challenging due to the extensive nature of the ODD. Nevertheless, employed approaches must be capable of scaling easily to encompass as much of the ODD as possible and validation should be conducted using realistic data [3,4].

Concurrently, there is an increasing emphasis on structured development processes and early validation. This transition is precipitated by methodologies such as systems engineering, which support the frontloading of development efforts. It has been demonstrated that such approaches also aid in defining component-specific ODDs when utilizing model-based systems engineering (MBSE) or more advanced techniques, such as scenario-based systems engineering (SBSE). The construction of a foundation for scenario-based testing is thus facilitated [8,9]. Furthermore, the rise of AI applications in perception functions necessitates early access to large amounts of data—in the case of perception-based functions specifically, camera or lidar data with corresponding ground-truth—for effective training and validation. The data required must encompass the entire ODD—including both normal system states and critical scenarios, which may be infrequent or considered edge cases. It is imperative for the system to exhibit robust behavior even in challenging situations. Two fundamental requirements are deduced from this analysis: Data generation methods need to scale efficiently while producing realistic, broadly useful data [10,11].

Using real-world data is difficult to scale due to the process of ground-truth labeling. On the other hand, synthetic data generation is easier to scale but suffers from the sim-to-real gap [10,12–14]. In the existing literature, only a limited number of sources describe the process of automated synthetic 3D environment generation [15]. This trend is also reflected in [1,2,10], while Ref. [2] explicitly summarizes: "Identifying critical scenarios and automating scenario generation are among the two most prominent challenges in the domain of testing AVs".

This work addresses this gap by developing a holistic approach for urban environment generation, specifically designed to construct scenes for the development and validation of autonomous vehicles. By focusing on scalability and narrowing the sim-to-real gap, our method addresses critical challenges faced by test engineers in simulation environments. We emphasize the importance of transforming logical scenarios—often characterized by numerous unknown or seemingly unimportant parameters—into practical concrete environments defined by only a handful of critical hyperparameters.

In this work, the terms Assisted Driving Systems (ADAS) and Automated Driving Systems (ADS) are used interchangeably, since the proposed framework for generating synthetic 3D environments is applicable to both. However, the intended use of the generated data differs depending on the level of automation. For ADAS, the framework can support the entire development and validation process. For ADS, it is intended for use in specific contexts, such as early stages of development and the identification of rare edge cases, which must subsequently be verified using real-world data. Accordingly, this work does not claim to replace real-world validation for higher levels of automation.

1.1. Background on Data Generation and Collection

Autonomous driving systems generally depend on a variety of sensors, including cameras, LiDAR, radar, and ultrasonic sensors. Different variants may be employed within each sensor category. For example, the camera category includes color and thermal cameras along with other variants. In this work, we focus on color camera data and the generation of corresponding ground-truth. However, it should be noted that the overarching concepts presented are also applicable to perception data from other sensor variants and categories. Henceforth, the general term "perception data" or simply "data" will be used in the following chapters. We start with a discussion of real-world data generation and collection techniques, followed by an examination of synthetic data approaches.

1.1.1. Real-World Data Collection and Validation

One prominent approach for data collection and validation involves using test tracks or specialized facilities designed to create diverse environmental conditions, such as varying wind effects or rain. These facilities offer a comparatively safe environment due to their controlled nature. They are also highly effective for generating high-quality data through controlled parameter changes, eliminating the unpredictability of real-world scenarios. Yet, this is also a limitation of the method since edge cases may not always present themselves, potentially leaving gaps in the validation process [16].

An alternative approach to perception data collection and system validation involves conducting tests on public roads. This method provides a variety of naturalistic scenarios and has the potential to reveal edge cases, although such instances may be limited.

Nonetheless, the aggregation of real-world data is generally a resource-intensive process, necessitating the deployment of high-precision sensor equipment and the allocation of significant time, among other factors, including personnel. The approach further demands the exhaustive generation of ground-truth data for the collected data (labeling) [17]. The labeling process is labor-intensive and therefore costly [13]. In the context of validating autonomous driving functions on public roads, there is a risk for safety. This is due to the possibility that the vehicle may not have been validated in all its ODD components, yet still has to navigate traffic and various driving conditions. While public roads offer a real-world environment, they are sometimes perceived as a somewhat uncontrolled environment for validation purposes without inherent systematics. Systematic approaches are viable for targeting specific conditions—however, this may be considered unethical in light of the safety risks outlined in [18].

It is evident that both approaches result in the generation of high-quality data in terms of realism. It must be noted that both methods are challenging to scale due to various factors, including cost, general effort, and the necessity for resources [1,16].

1.1.2. Synthetic Data as an Alternative to the Use of Real Data

The utilization of synthetic data during the development process represents a promising alternative. Simulation approaches generally constitute a more economical and secure alternative to developing solely in the real-world. Nevertheless, the utilization of simulated data is inherently limited by the discrepancy between simulated and actual outcomes—the sim-to-real gap. Moreover, it is even challenging to quantify the concept of "realness" using established metrics [15,16].

Altering and augmenting real-world sensor data is a common approach to enrich scene diversity without the cost of new data collection. Common methods include adding synthetic actors, applying weather or lighting tweaks, injecting sensor noise, and blending synthetic elements via domain randomization or generative models to create realistic new scenes [10,19]. The primary benefit of augmented data is its realism, which preserves physical consistency and sensor characteristics while facilitating coverage expansion and corner-case testing. However, this approach is not without challenges. Since the alteration is based on sensor data, maintaining spatiotemporal consistency between modalities is difficult [10,15]. This phenomenon can be observed when inserting synthetic actors into the scene. Maintaining consistency across multiple frames is challenging, as the output (effect) is altered, rather than the environment (cause). It is important to note that this may also result in the occurrence of quality degradation in the ground-truth, given that the original ground-truth may no longer be accurate. Finally, "closing the loop" with these techniques proves to be a challenging endeavor. This motivates additional approaches, which are presented in the next section.

1.2. Related Works on 3D Environment Generation for Autonomous Driving

As noted earlier, perception data can be generated by the utilization of simulation, incorporating scenarios that delineate both the static three-dimensional environment and the dynamic scene. The static environment describes everything remaining constant in the scene for the duration of the simulation. The dynamic environment describes all dynamic actors in the scene, which move around or change their representation [5,6,20]. In order to close the sim-to-real gap, it is necessary that both the simulation itself and the scenarios resemble reality as closely as possible. In the field of simulation, significant progress has been made in recent years, particularly in generating photorealistic visualizations and realistic physics, driven by game engines such as Unreal Engine [21,22]. Combined with advancements in the inclusion of material properties into simulations with physics-based rendering techniques and standards for representing physical properties of materials, a solid foundation for generating synthetic perception data is laid [7]. However, realistic simulations alone are insufficient. High-fidelity 3D environments are also required [23].

1.2.1. Digital 3D Environment Twins

A common data-driven strategy is the use of digital twins—for example, as demonstrated in [24], Virtual-Kitti 1 and 2 [25,26]. A digital twin of an environment is created by directly capturing and meticulously reconstructing every detail of a real-world landscape. Because of this the sim-to-real gap may be very narrow, especially when also encompassing material properties into the environment. However, the inherent limitations of real-world data collection are applicable in this context. The efficacy of this method is contingent on a number of factors. These include the particulars of the reconstruction process, which may stipulate target formats or augment the processing required, and the overarching objectives [10,24,27].

Digital twins are dependent on real-world scenes, thus naturally encompassing only specific concrete scenarios. During operation of an automated vehicle, this limitation is acceptable, as only the currently observed scene must be modeled, and the world model can be tailored to the requirements of that specific situation [28]. During development, digital twins are useful for edge-case testing, but the technique is challenging to scale. Synthetic edits have been found to be beneficial in this regard [10,15]. It is important to note that each modified scene still requires extensive processing to maintain consistency. Targeted test cases are difficult to produce as they either need a real-world counterpart or heavy processing.

1.2.2. Synthetic 3D Environments

An alternative to digital environment twins involves the usage of synthetic 3D environments to generate training data. In [14] the authors show that this kind of data is capable of replacing large portions of real data in the training of a perception function. The used 3D environments draw inspiration from real-world scenes but do not necessitate the utilization of real-world data as input. Environments explicitly created by human designers are one example. The designers model three-dimensional scenes according to specific requirements, without the necessity of these scenes exactly matching real-world locations. Environments created in this manner appear and feel realistic, leveraging the designer's internal understanding of how the world works. Since the structure and all elements of the scene are defined during the creation process itself, the groundwork for extracting ground-truth is laid. In this context some synthetic datasets exist, leveraging manually created 3D environments, for example from video games [13,29]. These datasets demonstrate the general usability of synthetic data in the training of perception neural networks [13,14,29–31]. The primary disadvantage associated with manual labor is the

absence of scalability, a feature that has been identified along with the challenging integration of material properties as one of the predominant deficiencies in purely synthetic scenarios, as outlined in [24]. This is countered by the introduction of automation into the generation process. This, in turn, could open up opportunities to further strengthen testing methods in the realm of autonomous driving. Examples of such methods include fuzzing, which is dependent on large numbers of scenarios. In this domain, several approaches already show great promise by intelligent alteration of visual and non-visual properties of the environment to discover software defects [32–36].

A comprehensive survey of recent advancements in the overarching domain of automated synthetic environment generation is found in [15]. The survey shows that a significant proportion of approaches concentrate on the reconstruction of indoor spaces. Conversely, only a limited number of approaches focus on nature or urban scenes, such as [37,38]. The survey further shows that a variety of methodologies and data presentation formats are used, each possessing distinct advantages in terms of usability, scalability, and realism. However, these approaches are frequently implemented as isolated solutions, not as a cohesive framework for the development of autonomous vehicles. Only a few works exist that propose synthetic datasets specifically designed for autonomous driving. The works of [39,40] feature a procedural modeling approach for the generation of the simulation environment. A ruleset is defined manually featuring a set of hyperparameters to construct the environment with a mixture of predefined models and procedural generated geometry. The work of [30] featuring the SYNTHIA dataset does not specify in detail how their world generation approach works and if the process can be automated or scaled. In the case of the SHIFT dataset, the corresponding paper does not give details about the 3D environments used in the generation of the dataset [41].

Despite progress in individual methods, holistic 3D environment generation for autonomous driving remains scarce. Current techniques address separate aspects in isolation, highlighting the need for a systematic framework that integrates them. The following sections present our contribution toward such an approach.

1.3. Contribution

The main contribution in this context is a framework that decomposes the challenge of transforming sparse input data into a high-fidelity 3D environment into sub-problems. Each sub-problem is addressed in an individual module, with each module implemented as an example to demonstrate continuity. We do not claim optimality in this, as part of the idea of the framework is that each method of generation may offer different advantages and disadvantages on the specific layers.

The multi-layered framework defines modules for

- Generating map-based data such as topology and land use information;
- Generating semantic city representations;
- Detailing environments to be used in photorealistic simulations.

Each layer builds upon the previous layer to define a different vital part of the environment.

The methodology employed in this work is a two-step process:

1. Establish (re-)generation rules based on ground-truth data;
2. Use the established rules to generate environments with minimal input data.

This approach makes the methodology scalable, as it does not require additional ground-truth data in the second step. The ground-truth used in this paper is fictional, but it is designed to be replaced by real-world data in the future. Using real-world data as ground-truth is beyond the scope of the current study.

Our modular framework enhances interpretability and adaptability, enabling an efficient scenario or, more specifically, scene generation and accommodating diverse user

needs. Although frontloading demands significant effort at the outset, it ultimately facilitates the quick and cost-effective production of scenarios. This addresses the previously mentioned research gap concerning the generation of synthetic 3D environments. Automated generation is generally overlooked in this context, and when it is introduced, it is often not well described. Existing research typically focuses on specific generation algorithms without exploring additional possibilities and raising the question of how to introduce ground-truth for rule generation in the generation process. We address this by highlighting various opportunities for integrating ground-truth into the generation of 3D environments, identifying multiple avenues for further exploration.

The efficacy of the generated environments is demonstrated by utilizing the simulation tool CARLA [22] to navigate through the scenes and collect camera images as well as ground-truth segmentation data. The generated image data is then used in conjunction with the real-world dataset, KITTI-360 [42], to train a YoloV8 segmentation network [43]. The objective of this training is to ascertain the extent to which real-world data samples can be supplanted by synthetic data samples without compromising performance.

The results indicate that our approach produces high-quality synthetic sensor data, demonstrating its potential for rapid generation of 3D environments and flexibility in employing various techniques tailored to user preferences. Furthermore, addressing different domains (such as topology and the semantic city) individually makes it easy to extend the framework. The validation results indicate that a significant portion of data may be replaced with synthetic data without significant quality degradation. Therefore, this research contributes valuable insights into improving simulation frameworks for autonomous vehicle testing.

2. Materials and Methods

First, we would like to introduce the general idea behind the proposed framework. This will be achieved by first examining the formalized inputs and outputs and the generalized parts. Following this, the modules will be examined in depth together with an exemplary use case that will be used for the first implementation. Subsequently, the validation of the proposed framework will be demonstrated.

In order to achieve scalability, a number of actions must be taken. The problem is to be considered as comprising multiple sub-problems. This approach facilitates integration and allows for better targeting of specific problem areas using dedicated technologies while also enabling the reuse of established techniques. The decomposition is oriented around the logical composition of real environments and their semantic hierarchy. Each layer is built semantically upon the previous one, as illustrated in Figure 1. The process begins with the reconstruction of the general topology/terrain. Based on this foundational understanding, we determine how the terrain is utilized semantically in two dimensions (2D semantics). Consequently, this information is instrumental in determining the representation of elements in three-dimensional space from a semantic perspective. Finally, based on the semantic 3D representation, detailed rendering specifications are applied to enhance visual fidelity. The generated environment can then be utilized to generate synthetic sensor data for training and validation purposes. The generation process is organized using a chunk system. This system involves the division of large-scale environments into chunks of equal size, which are then reconstructed individually. This approach facilitates the management of substantial data volumes necessitated by large environmental models, concurrently defining the output dimensions of reconstruction functions.

The sparse input data consists of ODD descriptions that can be formalized using specialized formats such as the ASAM OpenDRIVE (ODR) [6] and OpenSCENARIO (OSC) [5] standards. These formats are chosen as input into the framework due to their widespread

use in the automotive industry [20]. The present study will exclusively address the static component of the scene, which constitutes the predominant aspect of the environment in the visualization. The static part of the environment is solely defined in OpenDRIVE.

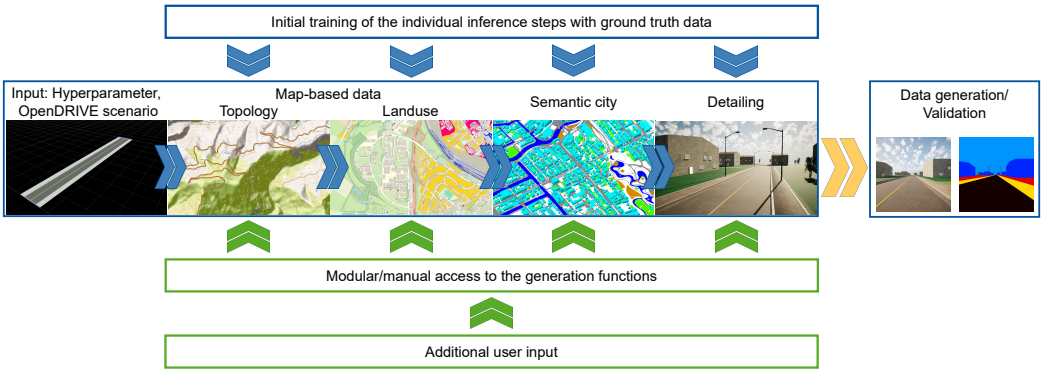


Figure 1. Concept of the proposed modular framework for the generation of static 3D environments.

The input data—i.e., the hyperparameters and the OpenDRIVE scenario—provides valuable information that is used for reconstruction. For instance, the geographical characteristics of the environment, such as the topography, can be deduced from the observed road network. A winding nature of roads may indicate a hilly terrain, while the presence of large industrial facilities typically suggests a flat, well-developed area. While these insights may be implicit, the road layout also provides explicit information, such as variations in elevation and land use at specific locations along the route. Through the analysis of both implicit and explicit cues derived from the input data and from any subsequent layer, neural networks and other techniques should be capable of making informed decisions during the reconstruction process. The wide variety of techniques that are used to reconstruct the different layers raises the problem of choice. However, this diversity also presents the prospect of conducting reconstruction in multiple ways. Rule-based, data-driven or hybrid approaches may be used at each of the individual layers. The choice is dependent upon the selected output format and the potential for the approach to be generalized, thereby rendering the reuse of the architecture of reconstruction functions viable.

In the following section, we first outline the underlying assumptions (Section 2.1) and introduce an exemplary use case for the first implementation of the framework, before discussing the individual generation steps in detail (Sections 2.2–2.4). For clarity, the chapters are named based on the corresponding modules shown in Figure 1.

2.1. Prerequisites and Assumptions

Given the broad ODDs of autonomous vehicles, the initial implementation and evaluation of the framework is carried out under simplified boundary conditions. In this initial study, three fundamental variables are selected as inputs:

- A road definition in OpenDRIVE;
- A parameter to switch between two imaginary countries—Country 1 and Country 2;
- A parameter to switch between two types of areas—industrial and residential.

Based on these inputs, the framework shall generate four different types of synthetic environments, each with distinct properties corresponding to its parameter combination. These properties are defined manually in advance and are loosely inspired by real-world examples. In the context of this preliminary implementation, it is imperative to note that all characteristics of the generated environments are entirely fictional. At this stage, no real-world data is utilized—the framework is developed and tested exclusively with synthetic ground-truth data. In future, it is envisaged that the environmental and landscape

attributes will be learned directly from real datasets. This will necessitate the alignment of the training data currently in use with the actual data sources available. Consequently, the subsequent application signifies a scaling step.

A short overview of chosen scene parameters is displayed in Table A1. Country 1 is modeled to represent a country with a higher population density, where space is used more efficiently. Country 2 is modeled to represent a more rural country. This overarching theme is further exemplified by the selection of materials employed. In this instance, area typical materials are utilized, which may be different but also equal, in order to demonstrate the flexibility of the framework. It is modeled that all countries are to be represented as completely flat. The question of how topology reconstruction may be incorporated will still be addressed.

An conceptualized exemplary implementation is described in the following subchapter, although we not claim that this is the optimal approach, as different interchangable techniques may have certain advantages and disadvantages.

2.2. Map-Based Data Generation—Topology and Land Use

Two prominent approaches for representing map-based terrain data and 2D semantics are vector-based mapping techniques and image/matrix-based representations. Each option has its own set of advantages and disadvantages. A vector-based approach has been demonstrated to offer a storage-efficient representation, which is advantageous when representing the structured elements of a map, such as building plots or even building footprints. A notable disadvantage of this approach is the challenge of constructing vector representations in which every area is defined exactly once, without the presence of redundancy. As a result, additional postprocessing may be needed to identify and fill gaps between polygons. This issue does not arise with matrix-based map representations, where every part of the image is discretely defined. Such representations have the capacity to encode both simple and complex shapes without affecting output size. Consequently, they are easily adaptable to different use cases. Furthermore, the integration of these systems with neural networks is seamless, given the inherent matrix structure of the data. The primary disadvantage associated with matrix-based maps is their increased storage requirements. Additionally, the level of detail that can be represented is constrained by the selected resolution of the image. In this instance, a matrix-based representation is chosen because of the higher flexibility and ease of integration to build a general foundation for all map-based data reconstruction.

In order to implement a reconstruction function, it is necessary to have access to data that serves as a ground-truth data for the desired appearance of the reconstructed landscape. Based on the assumption of locally flat terrain, the reconstruction is confined to two-dimensional semantics. Based on the environment properties defined in Table A1, a rule-based data generation tool is implemented, creating a total of eight different chunks (straight and curve, two countries, two area types). Samples of this generation technique are shown in Figure 2. The individual segments act as semantic classes and are colored differently for visualization purposes only. Industrial plots are displayed in red, residential plots in blue and grass in green. The roads are grey, with each shade of grey representing a different affiliation to country and area type combination. Images measuring 256 by 256 pixels are used at this point. As indicated in the work's objective, no real-world data is employed as the baseline so far. Nevertheless, it is still possible to obtain data for this purpose from sources such as OpenStreetMap [44].

In principle, the rule-based generator could be used to rebuild this layer directly. However, such a construction would rely heavily on the creator's implicit knowledge rather than on rules learned from ground-truth data. Therefore, the following sections

present our construction approach, which is based on learned rules developed using the created ground-truth data.

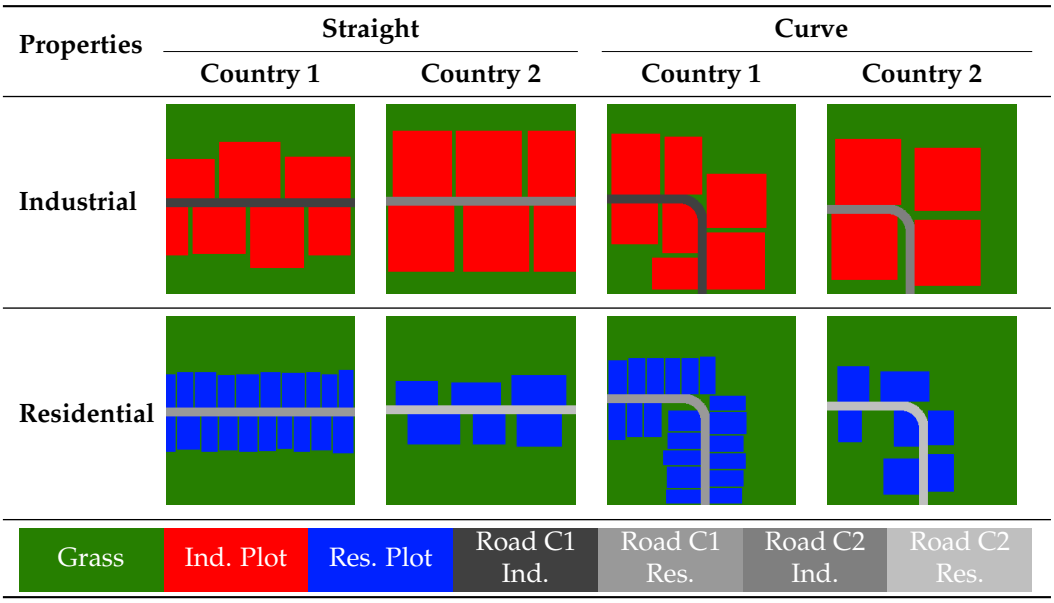


Figure 2. Samples of the training data for the land use reconstruction.

The problem is abstracted into an interpolating and extrapolating task. As mentioned earlier, some explicit information can be extracted from the input to serve as a guideline for reconstructing unknown areas. This is a common image processing task called inpainting. For this task, we use the neural network architecture proposed in "MAT: Mask-Aware Transformer for Large Hole Image Inpainting" by [45]. The proposed methodology is based on the generative adversarial network (GAN) architecture and uses a transformer for the inpainting task. The original network architecture is adapted to meet the requirements of this work. The changes are explained below and illustrated in Figure A1. In its original state, the input and output use RGB images. When it is used to reconstruct class-based land use data, for example, where each color represents a class, the output may contain invalid classes and other artifacts. This issue is resolved by utilizing additional output formats, such as one-hot encoding. The implementation of one-hot encoding ensures the elimination of ambiguous color values, thereby guaranteeing the production of exclusively valid classes. This, in turn, results in a simplification of both the training process and the subsequent postprocessing stage. To introduce this change, the original architectural design is adapted to accommodate varying input and output formats, leading to the development of generic input and output modules. The output extended by the option to generate one-hot encoded and black and white images for heightmaps. The single RGB-image input is modified to accommodate the two new data types as well. The number of input and output classes can differ in order to train on classes that will not be generated, such as the road. Finally, the existing input convolution fusing the inputs (originally RGB-image and mask) is made variable to allow for the use of multiple input maps. This optional feature is implemented to allow the generation of, for example, land use maps based on various inputs, such as a water map, a heightmap, and a mask. For this work, multiple input maps and black and white input/output have not been tested or used yet, but they are important for facilitating the use of a single base neural network technique for all the generations needed in map-based data in the future.

In order to facilitate the training of the network with one-hot encoded land use data, a minor adjustment is made to the loss calculation. The perceptual loss is replaced by cross-entropy loss. Based on the configuration, the output layer and the corresponding

activation function is substituted accordingly. In order to judge the training process, the before-used Fréchet Inception Distance (FID) is replaced by the mean Intersection over Union (MiOU) score [46,47].

We determined, given that the training data is less complex than the data for which the architecture was designed, that the discriminator network required a reduction in size by almost half of its original dimensions. This change is introduced by removing single convolution layers, effectively doubling the convolution needed from one layer to another. Furthermore, it is necessary to adopt the learning rates. The generator learning rate is changed from $\eta_{gen} = 0.002$ and to $\eta_{gen} = 0.0005$. The discriminator learning rate is set to $\eta_{dis} = \eta_{gen} \cdot 0.2 = 0.0001$. Semantic data label smoothing [48] is introduced to improve training with a smoothing factor of $\epsilon = 0.1$.

The network was trained for a total of 110 epochs. At this point, there are indications that the training process may be entering a phase of overfitting. The generator is used alone to create land use samples with just the input of a road course defined by the OpenDRIVE file and the relevant hyperparameters. Any defects that occur, such as tiny holes in individual plots, are repaired with local morphological closing inside large areas, such as the individual plots. If the outlines of the plots would get dilated and eroded, narrow plot dividing lines would get eliminated, which is unwanted. Figure 3 displays eight sampled land use maps, with post-processing applied. These are used in the further reconstruction of the environments, as the ones shown are in the generation of the dataset.

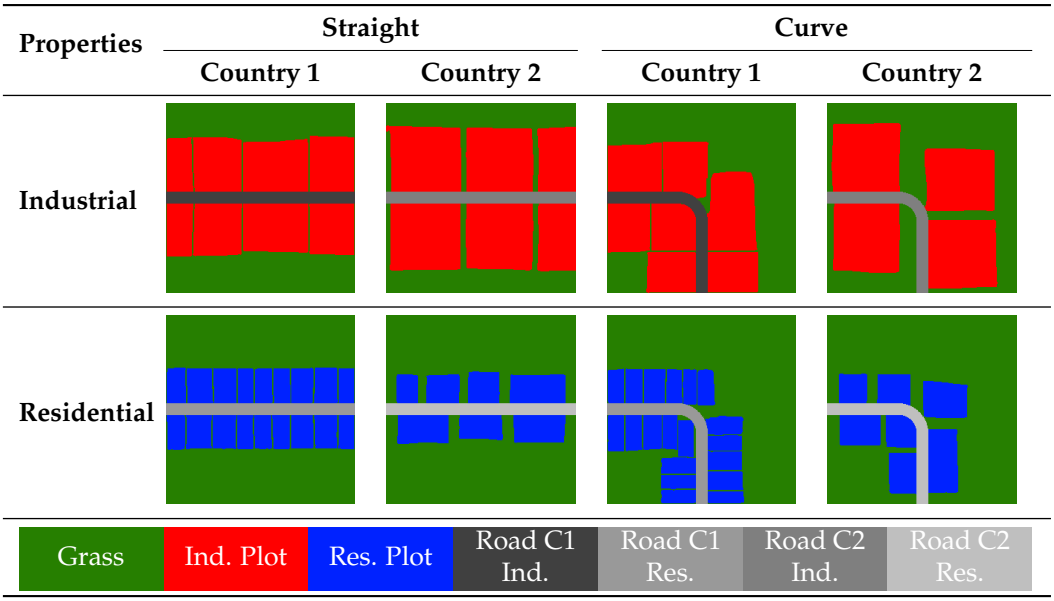


Figure 3. Land use samples generated by the trained network.

2.3. 3D Semantics Generation—Semantic City

The map-based data constitutes the foundation for the subsequent layer, namely the 3D semantics. In this layer, the position of simple three-dimensional (3D) objects, such as individual trees and props, is defined. In addition, the structure and attributes of more complex elements, including buildings, are specified. We use CityGML [49,50] to represent this information. CityGML is an open standard of the Open Geospatial Consortium (OGC) that has the capacity to represent landscapes through a variety of different modules with varying levels of detail.

This standard is chosen for several reasons. CityGML is a comprehensive model that can describe a wide range of urban and landscape elements, including buildings, transportation infrastructure, vegetation, water bodies, and terrain. Its many thematic modules permit the detailed representation of different aspects of a city or 3D environment,

thereby ensuring that all relevant components, such as roofs, facades, road areas, and vegetation types, can be modeled consistently. The schema may be extended with custom Application Domain Extensions (ADEs), rendering it suitable for specialized use cases that require additional attributes or new object types. CityGML is an XML-based open standard that is both machine-readable and widely supported, thereby enabling reliable data exchange between tools, frameworks, and institutions. The aforementioned properties of CityGML render it a robust foundation for the structuring and storage of the semantic 3D information that is required in our framework.

In order to transform the existing data into a CityGML representation, incorporating both input and previously reconstructed information, a range of tools are employed. The tool r:trån [51–53] developed at TU Munich is utilized for the purpose of converting OpenDRIVE files to a CityGML representation. The creation of buildings in CityGML is facilitated by the utilization of Random3DCity [54] functions, which have been adapted for this purpose.

Subsequent to the aforementioned conversion, rule-based, procedural modeling approaches are employed for the reconstruction of various components of the scene in a manner similar to [37,38]. In this layer, rule-based approaches are chosen to enhance the controllability of reconstruction. The restriction of possibilities has the advantage of obviating the necessity for implementing sanity checks. Furthermore, the transfer of parametrized objects to a formalized representation, such as CityGML, is thereby rendered more straightforward.

The placement of buildings is governed by a set of rules, the majority of which are defined by environmental properties in Table A1. Nevertheless, it is feasible to derive these regulations from real-world data from sources as indicated in [55,56]. The general parameter ranges for building construction are established manually. The exact ranges and parameters used are denoted in Table A2. The parameter ranges form the basis for defining the dimensions of the building and providing additional information, such as the type of roof to be used and the location of windows and doors. The placement of buildings on the existing plots is determined by iterating through the existing land use. For each plot, a rectangle is determined, which fits into the plot boundaries and thereby determines the boundaries of the building. Respecting these boundaries, parameters for the building are randomly sampled from the established ranges.

Parameter ranges for placing street furniture are detailed in Table A3. Street lights are placed along the road based on the distance between two instances and an offset from the road edge. Randomly scaled trees are placed randomly throughout the scene, respecting the outline of the buildings and the road, based on a specified vegetation density.

2.4. Detailing of the Generated Environment

Following the establishment of the 3D semantics, the final layer—the detailing stage—serves to integrate all components. In this step, the initial ODR input, heightmap, land use information, and the semantic 3D environment are assembled into a coherent, detailed 3D scene. This layer is structured into two parts: first, preparing the data for import into any 3D simulation environment and second, finalizing it within the chosen simulation platform. These steps work together to transform the abstract semantic representation into a fully realized, simulation-ready 3D environment. This split is introduced to make the simulation platform independent to a certain degree, and to increase modularity, albeit at the cost of additional input and output parsing overhead.

The topology is constructed from the heightmap using a polygonal mesh. The heightmap has a resolution of 256 by 256 pixels, representing an area of 200 by 200 m. The land use map is one pixel smaller in size because the heightmap defines the corner positions

of each grid cell, while the land use image defines the cell itself. Road surfaces are not defined by the land use map, but rather specified in a higher geometric detail through the CityGML r:trån definition. This definition may be directly interpreted to refer to polygonal surfaces. Nevertheless, the land use road class remains a requisite factor in determining the locations of road transitions. It is evident that, due to the nature of r:trån road geometry being continuous whilst the heightmap is discrete, the presence of gaps would be inevitable in the absence of correction. The resolution of this issue necessitated the development of a stitching algorithm, the function of which is to create surfaces that connect the road borders present in the land use map to the outermost curves of the r:trån road definition. For each point on these curves, the corresponding grid cell is identified, and polygons are drawn to connect all points that are required. Upon the entry of a new cell, any pending polygon sections are closed.

The generation of buildings is achieved through the direct interpretation of surfaces defined within the CityGML file. The surfaces are triangulated into polygonal meshes. These meshes are then combined with predefined high-resolution 3D objects.

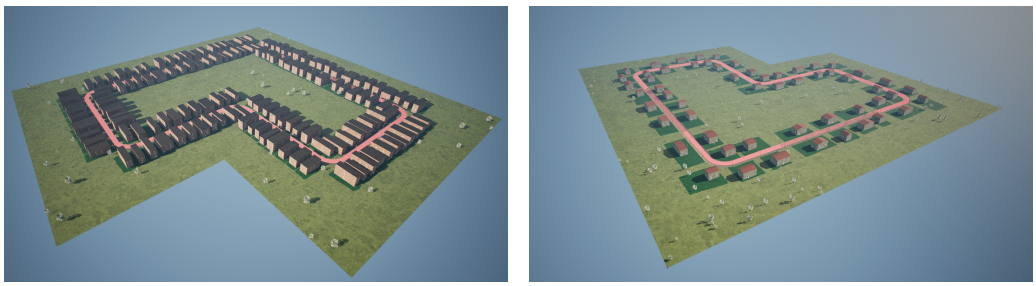
The selection of 3D objects and the later application of materials are governed by a mapping system that links metadata—such as country, area type, and object category—to a distribution. This distribution delineates the probability of spawning specific props or selecting particular materials. The system allows for realistic variation, which would be impossible with a single fixed choice. For example, while two countries might generally use fundamentally different materials for pavements, each country would still use a multitude of different variants with different probabilities. For now, these distributions have to be defined using expert knowledge, but they can also be learnt from real-world data. Therefore, the library is initially created manually.

Two principal methodologies can be employed for the incorporation of three-dimensional objects into the scene. The objects may be embedded directly in the glTF file, or nodes containing the metadata can be created for use in spawning the three-dimensional objects inside the simulation software later. The choice depends on the simulation environment and the intended use of the objects. In instances where the simulation supports direct import and the object does not necessitate additional features or functionality, embedding is a viable option. However, embedding increases file size and consumes more resources during import. In this instance, the second option is employed to distinctly separate the information generation process from the finalization of the environment. All three-dimensional objects are prepared in Unreal Engine as blueprints within a designated library. These blueprints are then inserted during the loading process of the environment. During the process of spawning, the metadata of the dummy nodes is utilized to adapt the 3D objects. In general, metadata attached to the nodes transfers information across tools, such as material selection or segmentation classes. In this final assembly of the environments, high-resolution 3D models are employed for the modeling of the windows, the doors [57], the trees [58] and streetlights [59,60]. Furthermore models and materials of Quixel Megascans [61] are used. In order to enhance the realism and aesthetic appeal of the materials, a technique known as texture bombing is employed on select materials.

The final target output format is a three-dimensional environment that is ready to be simulated in the open-source CARLA simulator [22]. In this paper, CARLA 0.10.0 is used, which is based on Unreal Engine 5 [21], in order to provide support for recent advancements in rendering techniques, including the Nanite system. The incorporation of such a system has the potential to enhance the framework's resilience and adaptability to future changes.

Following the transformation of the data into the agnostic 3D format GLTF, its import into CARLA is facilitated by a bespoke import pipeline based on glTFRuntime [62].

The glTFRuntime pipeline is significantly modified to enable the saving of parts of the environment, as well as to interpret additional metadata embedded within the created 3D environments. All elements of the environment are automatically interpreted and prepared for use within CARLA. This preparation involves the establishment of collision properties, the placement of predefined three-dimensional objects as previously referenced, and the allocation of suitable materials. In order to capture synthetic sensor data from the environments, it is necessary to combine multiple chunks in order to create a cohesive street network. This will ensure that route generation is smooth and that roads do not abruptly end. An example of such a street network is illustrated in Figure 4. Each chunk is generated individually, allowing for flexibility and customization in the reconstruction process.



(**a**) (**b**)

Figure 4. Two street networks generated with the framework are shown. (**a**) Presents an environment generated for Country 1 residential. (**b**) Presents an environment generated for Country 2 residential.

3. Results

Based on the proposed generation framework, four environments are created for each configuration defined earlier. These environments serve as the basis for generating camera data, accompanied by semantically segmented ground-truth data with CARLA. Sample frames showcasing these generated environments are displayed in Figure 5. The whole dataset, named "LGM dataset", is published along with this work in [63]. In total, the dataset contains 2388 images. The distribution of generated images across the generated environments is shown in Table A4.

To validate the approach, two methodologies may be employed. Firstly, the environments can be assessed for realism. This task proves challenging, as purely synthetic environments may lack real-life counterparts, necessitating comparisons of distributions—such as the placement of certain elements. Evaluating the entire environment in this manner presents difficulties due to the multitude of possible distributions that could be analyzed. It is more manageable to perform this assessment on individual reconstruction layers, as demonstrated with the generation of the land use map.

The second validation method focuses on usability rather than strict realism. While validating for realism assumes that the data closest to reality provides the best training and validation outcomes, one could argue that synthetic data may still be useful, despite not closely resembling realistic sensor data from actual test drives. The evaluation of the usefulness of the system is achieved by training a segmentation neural network using various mixtures of realistic and synthetic data. The accuracy of the network is monitored throughout the training process. This kind of validation is done in [13,14,30,39]. The objective is to ascertain the extent to which more expensive real-world data may be replaced with cheaper synthetic alternatives while maintaining accuracy in model performance. The task of image segmentation is chosen for validation because it is a fundamental function in many Automated Driving Systems. Since it represents only a part of the overall ADAS functionality, the performance influences of the input data can be observed more clearly.

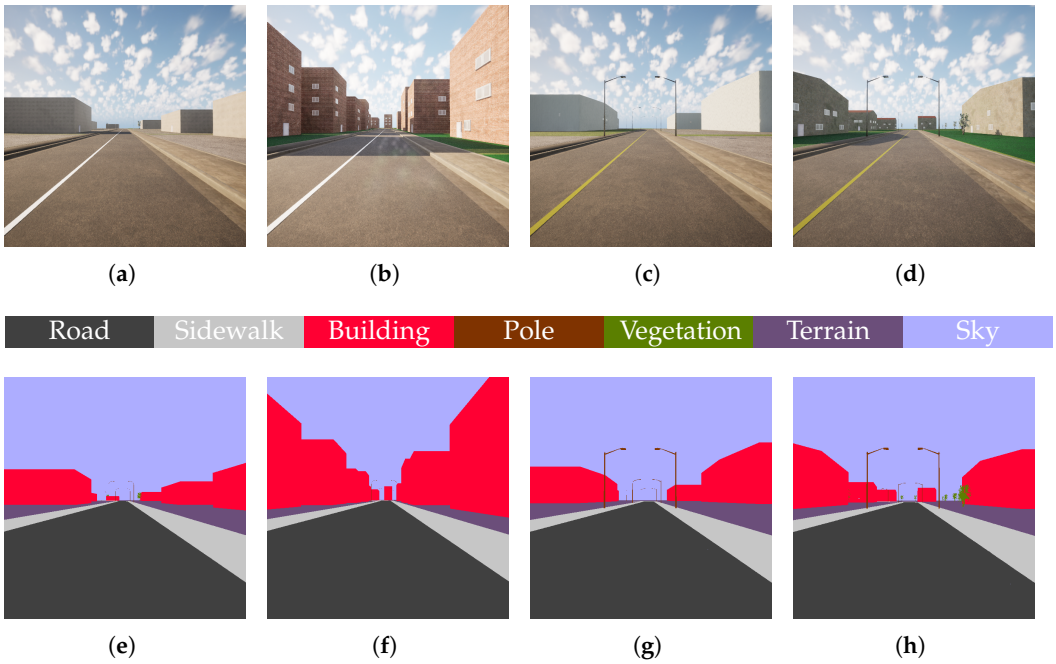


Figure 5. Four samples of color camera data (**top row**) and respective semantic segmentation (**bottom row**) sampled from the four generated environments: Country 1 industrial (**a**,**e**), Country 1 residential (**b**,**f**), Country 2 industrial (**c**,**g**), Country 2 residential (**d**,**h**).

Similarly to [13,30], the segmentation accuracies are analyzed per class and combined. As the global accuracy may be deceiving, we also include the Intersection over Union (IoU), the mean IoU (mIoU) and Frequency-weighted IoU (FWIoU). Additionally we analyzed the training process, varying both the number of images used for training and the ratio of real and synthetic data in the training dataset, while validating only on real data. The network chosen to be trained is Yolo-v8-n [43]. We chose the KITTI-360 dataset [42] as our real dataset. The segmentation classes trained on are road, sidewalk, building, pole, vegetation, terrain, and sky.

The neural network is trained 300 epochs each time. For each configuration, five different seeds are used. The number of images utilized for training purposes is determined to be 100, 500, and 2000 in total. We varied the ratio of real to synthetic data in 25% steps, later adding some 5% steps to spotlight the region where degradation starts to happen. 100% Real/Synth ratio means that 100% real data is used for the training process. Every trained network is validated at every epoch on 2000 images sampled randomly from all available drives of the KITTI-360 dataset, which were not used during training.

The global accuracies of the individual trainings are shown in Figure 6. As expected, with fewer images per epoch, the training process is slower, as all learning curves are more gradual. In addition, an increase in standard deviation is observed. In general, the greater the amount of synthetic data introduced, the greater the increase in standard deviation and the greater the decrease in accuracy.

All metrics per-class and combined at epoch 300 are displayed in the Tables A5 and A6. The metrics for each class show that the rare class pole is difficult to segment, exhibiting consistently low performance across all training configurations and metrics. Other classes achieve considerably better results. In the 2000-image and 500-image datasets, certain classes—road and building—are even segmented slightly more accurately when training is performed with partially synthetic data.

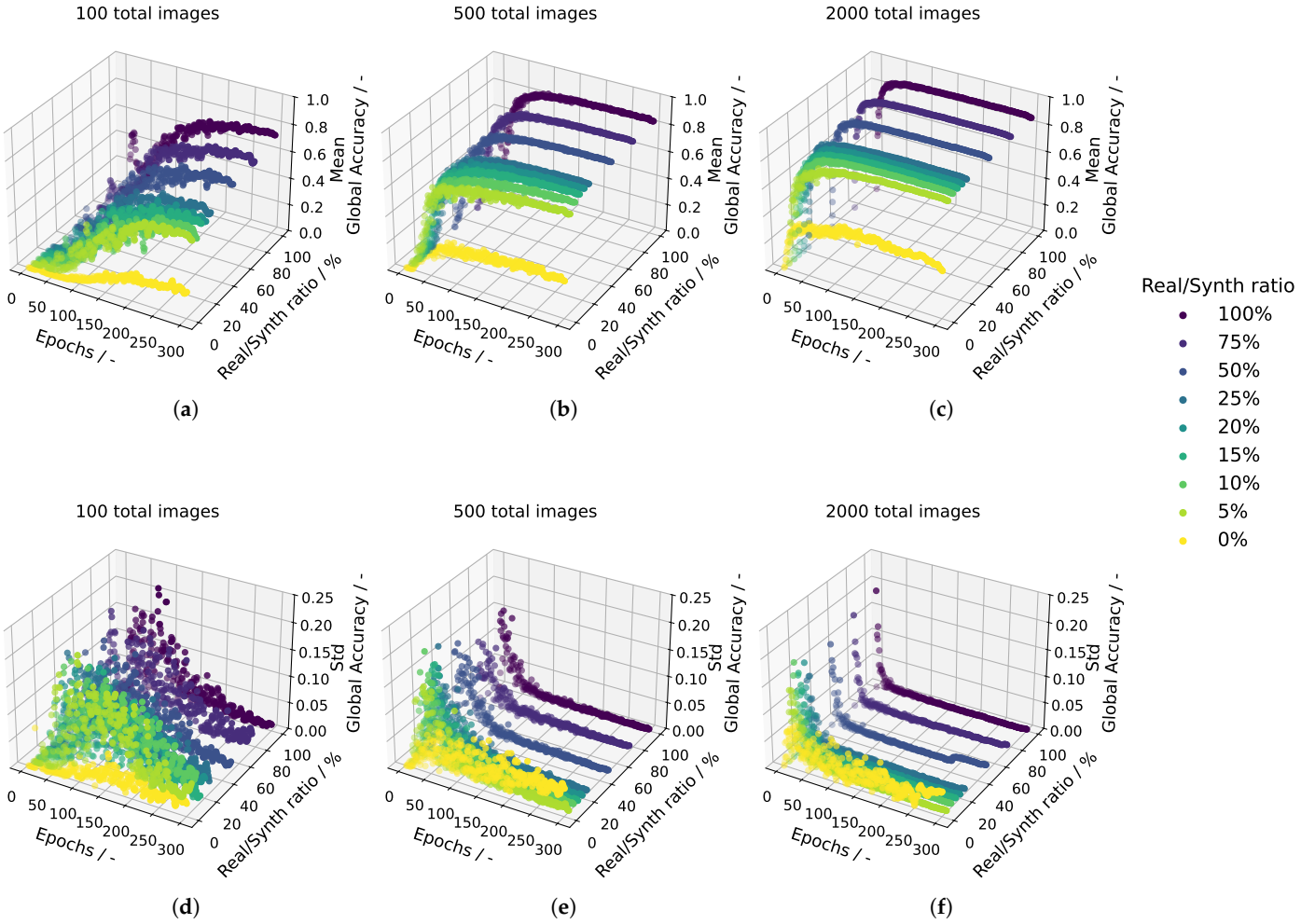


Figure 6. Global accuracies along the training process. In the upper row (**a**–**c**), the mean value across the five different seeds is displayed. In the lower row (**d**–**f**), the standard deviation is displayed. The columns correspond to 100 (**a**,**d**), 500 (**b**,**e**) and 2000 (**c**,**f**) training samples per epoch.

Figure 7 shows the Pareto front of the final global accuracies and mIoUs at epoch 300 for the different Real/Synth ratios. The mIoU scores are generally lower because this metric emphasizes rare classes more strongly. The degradation in accuracy and IoU scores becomes significant at different ratios depending on the dataset size, showing an initially gradual decline across all reported metrics until reaching a tipping point, after which the scores drop sharply. For a training set of 2000 images, the tipping point is located at up to 95% data replacement with an approximate 5.3% drop in global accuracy, 11% drop in the mIoU and a 6.7% drop in the FWIoU in contrast to the trainings performed solely on real samples. A fixed 5% performance drop happens for the mIoU and per-class accuracy at up to 85% and for the FWIoU at up to 90% replaced data. In the 500-image dataset, all metrics show that the gradual decline becomes more pronounced. The tipping point occurs earlier, requiring fewer real samples to be replaced. In contrast, the 100-image dataset exhibits a much more pronounced decline: Starting from an initial global accuracy of 73.9% and mIoU of 53.6%, performance drops by roughly 20% for global accuracy and 22.5% for mIoU when 95% of the real data is replaced by synthetic data. Overall, the smallest dataset shows higher standard deviations, indicating increased training instability.

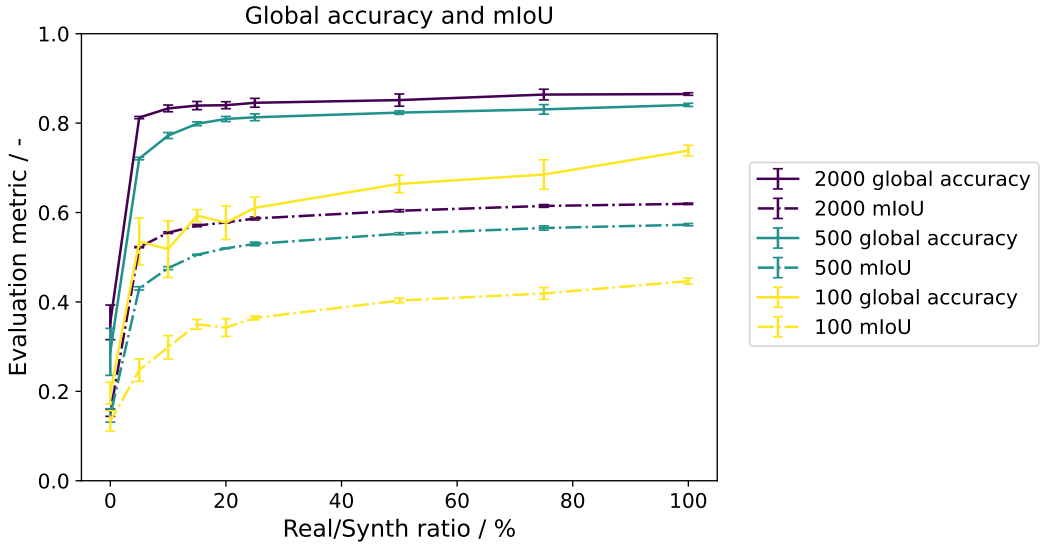


Figure 7. Mean global accuracies and mIoU scores with their standard deviations, at epoch 300, at varying Real/Synth ratios. Each line represents one of the three training dataset sizes used.

4. Limitations and Applicability Scope

The presented framework demonstrates the feasibility of generating synthetic 3D environments from sparse input data using a modular and layered design. The current implementation, however, represents an initial proof of concept evaluated under simplified boundary conditions.

The evaluation is conducted using fictional ground-truth data and simplified environmental assumptions. While this allows controlled testing of the modular pipeline, direct conclusions regarding real-world deployment remain limited. The framework is designed to integrate heterogeneous ground-truth sources, and initial tests indicate that such integration is feasible, although further investigation is required and planned.

Domain variability is currently restricted to two fictional countries and two region types, which are used to illustrate the general approach. The modular parameterization is intended to support extension to additional domains but has not yet been explored systematically.

Additionally, simplified scene assumptions, including flat terrain and static environments, are applied to reduce complexity during initial evaluation. The framework is designed to allow interchangeable generation techniques within individual modules, enabling incremental expansion toward more complex environments.

5. Conclusions

This work presents a modular and flexible to generate synthetic 3D environments. The fast and cheaply generated 3D environments are used to produce high-quality synthetic sensor data, in conjunction with consistent ground-truth. The findings from the validation experiments demonstrate that the synthetic data generated has the capacity to substitute a significant proportion of real data prior to the occurrence of perceptible accuracy degradation. This finding suggests that the generated environments are capable of capturing the relevant structural and semantic properties needed for training and validating perception models. Synthetic data and our underlying framework present a promising alternative to the usage of purely real-world data collection, as real-world data is connected to high costs, exhaustive manual work, and—in some cases—even unethical and dangerous techniques. Nevertheless, the results of the study indicate further possibilities for improvement.

We show the usage of inpainting networks for maps used to reconstruct 3D environments. Our presented technique works, but may be utilized further. Additional map-based data, such as the heightmap, may be generated. This data may serve as the foundation

for the generation of water maps or further data, which can be abstracted to a 2D map. As the data to be generated can be diverse, the architecture may need additional changes. Additionally, automated hyperparameter tuning should be explored.

CityGML proves to be a powerful foundation for representing the whole static 3D environment semantically. For now, a variety of different objects are introduced, which may be expanded in the future to fill the scene with life.

This holds true for the detailing stage. Here, additional components should be introduced, such as foliage and depth to certain materials, with, for example, virtualized geometry through systems like Unreal Engine's Nanite system. This would introduce more (wanted) noise to the generated 3D environments, making them more variable. This addition is also interesting for generating LiDAR data from the given 3D environments.

Our validation shows that in our experiments, approximately up to 85% of the real data can be replaced by our synthetic data without significant performance degradation, depending on the training configuration. Our extensive training results with different Real/Synth ratios and training set sizes give a good indication that our data can be used in varying situations. A key contribution is the LGM dataset we present here, alongside its use in [63]. The dataset offers a total of 2388 color images along with a semantic segmentation in seven classes.

For future research, we identified three main goals we want to explore further. Firstly, domain shifts or general variations of the generated environments are of great interest. In this regard, the framework may be further extended by introducing additional features and components in the individual modules. In the detailing layer, dynamic effects such as weather and day/night cycles may be introduced. In the case of the map-based data reconstruction and 3D semantics, further areas may be included in the generation as well as further use cases. Two interesting use cases are agricultural and maritime applications, where map-based data, such as currents or, in the case of agriculture, soil conditions can be reconstructed along with relevant scenery.

Secondly, we would like to explore further techniques for the generation of the different layers. In the case of placement of scene components or map-based generation approaches, such as stated in [64], this can be implemented. This would enhance the possibilities and the flexibility of the proposed framework, as more diverse input data may be introduced. Also, environments could be represented with varying granularity, which may be more suitable for a specific use case. We plan on incorporating dynamic objects with interactions, as shown in [65] in the framework as well.

Thirdly, although the data produced is already usable in its current state, we would like to introduce real data as the baseline on which we train or define every single generation step. For this, the groundwork is already laid, as the layers are designed to encompass real data sources, as indicated. All of these points would greatly enhance the possibilities of the framework for generating scenarios, which can cover a great portion of the ODD of an ADAS functionality. As previously mentioned, this could lead to further advancements in the fields of ADAS fuzzing and ODD exploration. Consequently, this could facilitate targeted searches specifically targeting edge cases. Such a possibility would increase the safety for autonomous vehicles.

Author Contributions: Conceptualization, T.T., J.D., M.E. and J.A.; methodology, T.T., J.D., L.L., B.K. and T.B.; software, T.T., J.D. and L.L.; validation, T.T. and J.D.; formal analysis, T.T. and J.D.; investigation, T.T.; resources, T.T., J.D. and B.K.; data curation, T.T. and J.D.; writing—original draft preparation, T.T.; writing—review and editing, T.T., J.D., L.L., B.K., T.B., M.E. and J.A.; visualization, T.T. and J.D.; supervision, M.E. and J.A.; project administration, T.T., M.E. and J.A.; funding acquisition, M.E. and J.A. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Data Availability Statement: The original data presented in the study are openly available in LGM_SemanticSegmentationDataset_1 at https://doi.org/10.5281/ZENODO.17949756.

Acknowledgments: Simulations were performed with computing resources granted by RWTH Aachen University under project rwth1892.

Conflicts of Interest: The authors declare no conflicts of interest.

Abbreviations

The following abbreviations are used in this manuscript:

AD	Automated Driving
ADS	Automated Driving Systems
ADAS	Advanced Driver Assistance Systems
AV	Automated Vehicle
FWIoU	Frequency Weighted IoU
GAN	Generative Adversarial Networks
IND	Industrial
IoU	Intersection Over Union
ODD	Operational Design Domain
ODR	OpenDRIVE
OSC	OpenSCENARIO
MBSE	Model-Based Systems Engineering
mIoU	Mean Intersection Over Union
RES	Residential
SBSE	Scenario-Based Systems Engineering
STD	Standard Deviation

Appendix A

Table A1. Overview of environment properties.

Properties	**Country 1**		**Country 2**	
	Industrial	**Residential**	**Industrial**	**Residential**
General ground type	Green grass	Green grass	Green grass	Green grass
Road markings	White	White	Yellow	Yellow
Plot size	Medium	Long, narrow	Large	Medium
House size	Large warehouses	Tall inner-city style	Large warehouses	Small rural
House material	Sheet metal	Red brick	Sheet metal	Flagstone
Ground type	Rubble	Fit green grass	Rubble	Fit green grass
Street light type	Virginia street light	Virginia street light	New York street light	New York street light
Tree density	Medium	High	Low	Very High

Table A2. Overview of the parameter ranges used for the procedural building generation.

Category	Property	Unit	Country 1		Country 2	
			Industrial	Residential	Industrial	Residential
Building Dimensions	Min. length	m	20	25	35	10
	Max. length	m	30	36	65	15
	Min. width	m	15	8	35	10
	Max. width	m	20	21	65	25
	Min. distance to plot border	m	2	2	4	3
Building Stories	Min. count	-	2	4	3	3
	Max. count	-	3	5	4	3
	Height	m	3.1	3.1	3.1	3.1
Building Door	Height	m	2	2	2	2
	Width	m	1	1	1	1
	Min. distance to the left	m	1	1	1	1
	Min. distance to the right	m	1	1	1	1
Building Roof	Type	-	Flat	Flat	Flat, Gabled	Gabled
Building Windows—parameters for each side and story	Min. count	-	Story 0–1: 0	Story 0: 0, Story 1–4: 1	Story 0: 0, Story 1: 0	Story 0–1: 1
	Max. count	-	Story 0–1: 0	Story 0: 1, Story 1–4: 2	Story 0: 0, Story 1: 2	Story 0–1: 3
	Min. width	m	Story 0–1: 0	Story 0–4: 1	Story 0: 0, Story 1: 0.5	Story 0–1: 1
	Max. width	m	Story 0–1: 0	Story 0–4: 1	Story 0: 0, Story 1: 1.5	Story 0–1: 1
	Min. distance to the left	m	Story 0–1: 0	Story 0–4: 1	Story 0: 0, Story 1: 1	Story 0–1: 1
	Min. distance to the right	m	Story 0–1: 0	Story 0–4: 1	Story 0: 0, Story 1: 1	Story 0–1: 1
	Min. distance between windows	m	Story 0–1: 0	Story 0–4: 0.3	Story 0: 0, Story 1: 0.3	Story 0–1: 0.3
	Copy window configuration from the story below?	bool	Story 0–1: 0	Story 0: 0, Story 1–4: 1	Story 0–1: 0	Story 0–1: 0
Street light	Distance between street lights	m	40	40	40	40
	y-Offset from the edge of the road inwards	m	0.5	0.5	0.5	0.5

Table A3. Overview of the parameter ranges used for the procedural street furniture generation.

Category	Property	Unit	Country 1		Country 2	
			Industrial	Residential	Industrial	Residential
Street lights	Distance between street lights	m	40	40	40	40
	y-Offset from the edge of the road inwards	m	0.5	0.5	0.5	0.5
Trees	Min. number of trees per chunk	-	5	5	1	30
	Max. number of trees per chunk	-	6	10	3	40
	Min. scale	-	0.8	0.7	0.7	0.5
	Max. scale	-	1.2	1.3	1.1	1

Table A4. Distribution of the generated images in the accompanying dataset across the four 3D environments created.

Country 1 Industrial	Country 1 Residential	Country 2 Industrial	Country 2 Residential
575	626	557	630

Table A5. The accuracies for each class in epoch 300, averaged over the 5 training runs per configuration, with their respective standard deviations. All numerical values are expressed as percentages.

2000 Training Samples									
Real/Synth Ratio	**Road**	**Sidewalk**	**Building**	**Pole**	**Vegetation**	**Terrain**	**Sky**	**Per-Class Accuracy**	**Global Accuracy**
100	92.4 ± 0.4	82.1 ± 0.2	82.8 ± 1.2	17.1 ± 1.0	87.5 ± 0.6	78.1 ± 0.8	91.5 ± 0.2	75.9 ± 0.2	86.5 ± 0.3
75	93.8 ± 0.4	81.1 ± 0.5	84.1 ± 0.8	15.7 ± 1.0	86.2 ± 3.0	77.6 ± 0.7	91.6 ± 0.4	75.7 ± 0.5	86.4 ± 1.2
50	93.3 ± 0.2	79.2 ± 0.7	84.2 ± 0.5	12.1 ± 1.3	83.9 ± 3.1	76.5 ± 0.6	91.0 ± 0.2	74.3 ± 0.5	85.2 ± 1.4
25	92.6 ± 0.2	74.5 ± 0.9	81.9 ± 0.6	8.6 ± 1.0	85.2 ± 2.2	73.1 ± 1.2	90.8 ± 0.4	72.4 ± 0.5	84.5 ± 1.0
20	92.0 ± 0.7	71.9 ± 1.2	80.6 ± 0.9	7.3 ± 1.2	85.3 ± 1.7	73.1 ± 1.1	90.1 ± 0.5	71.5 ± 0.2	84.0 ± 0.8
15	91.5 ± 0.7	69.5 ± 0.3	79.8 ± 1.2	6.4 ± 0.7	86.2 ± 1.9	71.8 ± 1.2	90.2 ± 0.3	70.8 ± 0.3	83.9 ± 0.9
10	91.0 ± 0.8	65.4 ± 0.6	78.0 ± 1.2	6.1 ± 1.5	87.3 ± 2.0	64.7 ± 2.3	89.6 ± 0.4	68.9 ± 0.3	83.3 ± 0.8
5	88.4 ± 0.6	61.4 ± 1.6	73.8 ± 0.8	2.8 ± 0.6	87.6 ± 0.7	54.0 ± 3.7	88.4 ± 0.8	65.2 ± 0.5	81.2 ± 0.3
0	9.3 ± 7.6	2.1 ± 0.9	26.2 ± 6.5	1.2 ± 0.5	51.3 ± 10.4	51.6 ± 12.6	45.5 ± 16.4	26.7 ± 0.9	34.9 ± 4.6

500 Training Samples									
Real/Synth Ratio	**Road**	**Sidewalk**	**Building**	**Pole**	**Vegetation**	**Terrain**	**Sky**	**Per-Class Accuracy**	**Global Accuracy**
100	90.7 ± 0.6	70.8 ± 1.4	78.5 ± 1.1	8.5 ± 1.3	87.8 ± 0.4	67.8 ± 1.9	90.1 ± 0.4	70.6 ± 0.4	84.1 ± 0.4
75	91.2 ± 0.3	70.8 ± 1.3	79.5 ± 1.0	6.5 ± 0.9	85.0 ± 2.3	64.4 ± 1.1	90.5 ± 0.3	69.7 ± 0.7	83.1 ± 1.1
50	91.2 ± 0.7	67.9 ± 1.3	79.7 ± 0.4	4.2 ± 0.5	84.5 ± 0.8	57.2 ± 2.7	90.1 ± 0.6	67.8 ± 0.5	82.4 ± 0.4
25	88.9 ± 1.1	60.6 ± 1.3	75.2 ± 1.1	4.6 ± 0.7	86.6 ± 2.6	56.8 ± 3.8	89.0 ± 0.5	66.0 ± 0.6	81.3 ± 0.8
20	88.8 ± 0.9	58.8 ± 2.7	75.2 ± 1.3	3.9 ± 0.9	86.4 ± 1.6	53.3 ± 2.2	88.3 ± 0.3	65.0 ± 0.4	80.9 ± 0.6
15	88.1 ± 1.2	55.1 ± 1.4	73.1 ± 1.1	2.8 ± 1.1	85.8 ± 1.5	54.5 ± 1.4	87.8 ± 0.3	63.9 ± 0.2	79.9 ± 0.4
10	85.7 ± 1.7	49.8 ± 4.3	66.9 ± 1.4	1.8 ± 0.8	86.2 ± 1.0	38.7 ± 7.2	87.6 ± 0.3	59.5 ± 0.9	77.2 ± 0.7
5	81.2 ± 3.2	45.4 ± 4.0	53.3 ± 0.2	1.5 ± 0.7	84.6 ± 1.3	29.4 ± 2.9	86.4 ± 0.8	54.6 ± 0.8	72.1 ± 0.3
0	19.9 ± 15.5	7.2 ± 2.7	13.7 ± 1.3	0.6 ± 0.4	30.4 ± 9.8	52.1 ± 4.0	71.2 ± 2.2	27.9 ± 2.3	27.6 ± 5.2

100 Training Samples									
Real/Synth Ratio	**Road**	**Sidewalk**	**Building**	**Pole**	**Vegetation**	**Terrain**	**Sky**	**Per-Class Accuracy**	**Global Accuracy**
100	78.9 ± 2.9	49.4 ± 3.5	68.6 ± 4.1	0.3 ± 0.3	80.2 ± 1.9	41.5 ± 6.0	86.2 ± 1.1	57.9 ± 1.4	73.9 ± 1.2
75	78.2 ± 3.0	41.7 ± 5.7	59.1 ± 3.3	0.6 ± 0.6	73.9 ± 7.4	41.6 ± 3.8	84.9 ± 1.9	54.3 ± 1.9	68.5 ± 3.3
50	77.9 ± 1.9	39.7 ± 2.9	55.5 ± 3.7	0.3 ± 0.4	72.0 ± 4.8	39.2 ± 5.1	81.8 ± 0.6	52.4 ± 1.5	66.4 ± 2.0
25	69.7 ± 5.2	35.5 ± 6.2	43.9 ± 2.9	0.0 ± 0.0	70.0 ± 7.6	41.8 ± 7.9	76.1 ± 3.4	48.1 ± 0.5	61.1 ± 2.4
20	71.8 ± 4.7	33.0 ± 7.9	39.1 ± 6.1	0.0 ± 0.0	65.7 ± 7.1	32.7 ± 13.5	73.1 ± 7.5	45.0 ± 3.4	57.7 ± 3.8
15	71.0 ± 2.8	35.0 ± 3.4	40.1 ± 4.7	0.0 ± 0.0	67.6 ± 4.0	38.2 ± 5.2	76.9 ± 5.4	47.0 ± 0.9	59.3 ± 1.3
10	65.7 ± 7.5	29.3 ± 7.9	35.8 ± 1.4	0.0 ± 0.0	57.8 ± 14.2	37.6 ± 5.9	62.1 ± 8.6	41.2 ± 2.9	51.8 ± 6.3
5	58.3 ± 8.0	32.2 ± 4.0	13.2 ± 3.3	0.0 ± 0.0	81.7 ± 10.1	36.5 ± 12.1	31.9 ± 11.4	36.2 ± 4.0	53.5 ± 5.3
0	20.7 ± 17.3	15.6 ± 3.4	11.6 ± 3.7	0.0 ± 0.0	14.6 ± 7.9	58.2 ± 12.9	48.5 ± 12.0	24.2 ± 3.2	19.6 ± 2.4

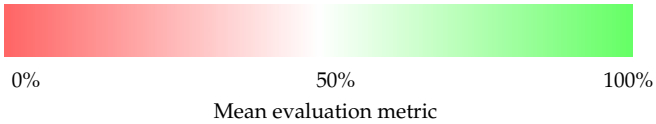


Table A6. The intersection over union for each class at epoch 300, averaged over the 5 training runs per configuration, with their respective standard deviations. All numerical values are expressed as percentages.

2000 Training Samples									
Real/Synth Ratio	**Road**	**Sidewalk**	**Building**	**Pole**	**Vegetation**	**Terrain**	**Sky**	**mIoU**	**FWIoU**
100	90.9 ± 0.3	75.0 ± 0.3	79.2 ± 1.2	15.8 ± 0.8	84.4 ± 0.6	70.5 ± 0.7	88.9 ± 0.1	72.1 ± 0.2	83.2 ± 0.3
75	91.9 ± 0.3	74.6 ± 0.6	80.3 ± 0.7	14.5 ± 0.9	83.1 ± 2.9	69.6 ± 0.6	89.0 ± 0.4	71.8 ± 0.5	83.0 ± 1.1
50	91.3 ± 0.2	72.5 ± 0.4	80.3 ± 0.5	11.5 ± 1.2	80.8 ± 2.9	68.9 ± 0.5	88.5 ± 0.3	70.5 ± 0.4	81.7 ± 1.3
25	90.2 ± 0.3	68.4 ± 0.6	78.4 ± 0.4	8.2 ± 1.0	81.5 ± 2.1	65.5 ± 1.0	87.7 ± 0.2	68.5 ± 0.5	80.8 ± 0.9
20	89.4 ± 0.6	65.9 ± 1.0	77.0 ± 0.8	7.0 ± 1.1	81.4 ± 1.5	65.2 ± 0.8	87.1 ± 0.5	67.6 ± 0.2	80.2 ± 0.7
15	88.5 ± 0.4	63.7 ± 0.5	76.3 ± 1.0	6.2 ± 0.6	82.1 ± 1.8	64.3 ± 1.0	86.8 ± 0.1	66.8 ± 0.3	80.0 ± 0.8
10	87.7 ± 0.5	59.6 ± 0.6	74.4 ± 1.0	5.9 ± 1.4	82.6 ± 1.5	58.7 ± 1.7	86.1 ± 0.5	65.0 ± 0.2	79.1 ± 0.6
5	84.7 ± 0.6	55.2 ± 1.6	70.2 ± 0.7	2.7 ± 0.6	81.9 ± 0.5	48.4 ± 2.6	84.8 ± 0.4	61.1 ± 0.4	76.5 ± 0.2
0	9.0 ± 5.5	2.1 ± 1.0	20.5 ± 2.9	0.9 ± 0.4	45.3 ± 8.6	32.2 ± 2.3	36.9 ± 8.0	21.0 ± 1.6	29.6 ± 3.7

500 Training Samples									
Real/Synth Ratio	**Road**	**Sidewalk**	**Building**	**Pole**	**Vegetation**	**Terrain**	**Sky**	**mIoU**	**FWIoU**
100	88.1 ± 0.5	64.3 ± 0.9	75.0 ± 0.9	8.1 ± 1.2	83.7 ± 0.4	61.7 ± 1.2	86.7 ± 0.2	66.8 ± 0.3	80.2 ± 0.4
75	88.2 ± 0.3	63.6 ± 0.8	75.8 ± 0.8	6.2 ± 0.8	81.1 ± 2.2	59.0 ± 1.0	87.3 ± 0.3	65.9 ± 0.6	79.2 ± 1.0
50	87.9 ± 0.6	61.3 ± 1.0	75.9 ± 0.3	4.1 ± 0.5	80.1 ± 0.7	52.9 ± 2.4	87.0 ± 0.5	64.2 ± 0.4	78.3 ± 0.4
25	84.9 ± 1.1	54.8 ± 1.1	71.5 ± 0.7	4.4 ± 0.6	81.6 ± 2.1	51.2 ± 3.0	85.3 ± 0.6	62.0 ± 0.5	76.8 ± 0.7
20	84.6 ± 0.5	53.2 ± 1.8	71.2 ± 0.9	3.7 ± 0.9	81.0 ± 1.3	47.9 ± 1.7	84.8 ± 0.4	60.9 ± 0.3	76.2 ± 0.5
15	83.6 ± 0.7	50.1 ± 1.3	68.9 ± 1.0	2.7 ± 1.0	80.3 ± 1.0	48.0 ± 1.4	84.1 ± 0.4	59.7 ± 0.1	75.0 ± 0.2
10	81.0 ± 1.4	45.3 ± 2.9	63.5 ± 1.3	1.7 ± 0.8	78.6 ± 0.5	35.1 ± 6.2	83.9 ± 0.3	55.6 ± 0.7	71.8 ± 0.5
5	76.5 ± 2.7	40.5 ± 2.8	51.0 ± 0.1	1.5 ± 0.7	74.5 ± 0.6	26.6 ± 2.9	80.0 ± 0.9	50.1 ± 0.7	65.6 ± 0.3
0	17.9 ± 9.8	5.7 ± 2.3	12.0 ± 1.7	0.6 ± 0.3	29.8 ± 8.2	33.2 ± 0.8	38.9 ± 6.7	19.7 ± 1.9	23.1 ± 4.2

100 Training Samples									
Real/Synth Ratio	**Road**	**Sidewalk**	**Building**	**Pole**	**Vegetation**	**Terrain**	**Sky**	**mIoU**	**FWIoU**
100	75.5 ± 2.4	41.4 ± 2.1	64.5 ± 3.4	0.3 ± 0.3	74.8 ± 1.0	36.9 ± 4.2	81.8 ± 0.5	53.6 ± 1.1	69.0 ± 1.0
75	74.5 ± 2.6	35.9 ± 4.0	56.3 ± 2.7	0.6 ± 0.6	68.8 ± 6.2	36.7 ± 2.8	80.5 ± 1.0	50.5 ± 1.7	64.1 ± 2.9
50	74.0 ± 1.6	35.3 ± 2.0	53.0 ± 3.0	0.3 ± 0.4	66.6 ± 4.0	34.6 ± 4.3	77.6 ± 1.4	48.8 ± 1.3	62.1 ± 1.8
25	66.7 ± 4.7	30.9 ± 4.1	42.4 ± 2.7	0.0 ± 0.0	63.1 ± 5.9	35.0 ± 6.5	70.2 ± 3.1	44.0 ± 0.3	56.2 ± 1.8
20	68.5 ± 4.0	28.4 ± 6.1	37.9 ± 5.7	0.0 ± 0.0	59.2 ± 4.9	27.8 ± 9.9	67.7 ± 6.4	41.3 ± 2.9	53.2 ± 3.4
15	67.8 ± 2.4	29.6 ± 2.4	38.8 ± 4.1	0.0 ± 0.0	60.7 ± 2.2	32.1 ± 3.0	68.3 ± 3.2	42.5 ± 0.7	54.3 ± 0.8
10	62.3 ± 6.5	25.0 ± 5.4	34.8 ± 1.4	0.0 ± 0.0	52.2 ± 11.1	31.4 ± 4.0	57.7 ± 7.0	37.6 ± 2.6	47.7 ± 5.2
5	55.8 ± 6.5	25.8 ± 2.9	13.0 ± 3.3	0.0 ± 0.0	63.5 ± 4.8	30.8 ± 9.0	28.8 ± 10.0	31.1 ± 3.3	44.5 ± 3.3
0	19.1 ± 14.7	11.5 ± 2.5	10.0 ± 3.0	0.0 ± 0.0	13.9 ± 7.3	24.4 ± 8.7	39.9 ± 9.3	17.0 ± 3.1	16.2 ± 2.4



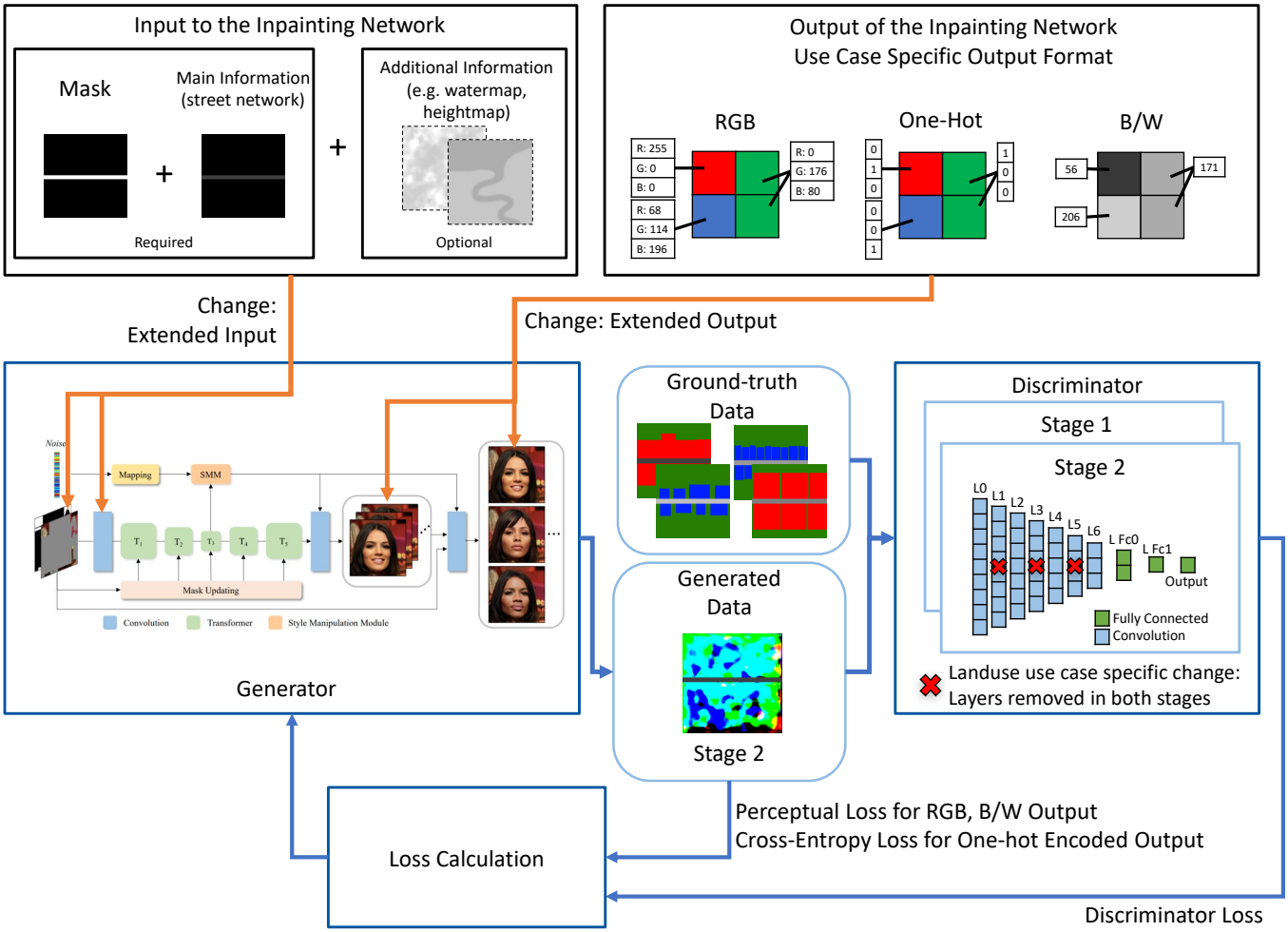


Figure A1. Changes introduced to the original MAT inpaint network architecture [45] for the generation of map-based land use data. The orange arrows indicate where the changes in the generator input and output are introduced.

References

1. Lou, G.; Deng, Y.; Zheng, X.; Zhang, M.; Zhang, T. Testing of Autonomous Driving Systems: Where Are We and Where Should We Go? In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Singapore, 14–18 November 2022*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 31–43. [CrossRef]
2. Song, Q.; Bensoussan, A.; Mousavi, M.R. Synthetic versus Real: An Analysis of Critical Scenarios for Autonomous Vehicle Testing. *Autom. Softw. Eng.* **2025**, *32*, 37. [CrossRef]
3. Wäschle, M.; Thaler, F.; Berres, A.; Pölzlbauer, F.; Albers, A. A Review on AI Safety in Highly Automated Driving. *Front. Artif. Intell.* **2022**, *5*, 952773. [CrossRef]
4. Mehlhorn, M.A.; Richter, A.; Shardt, Y.A. Ruling the Operational Boundaries: A Survey on Operational Design Domains of Autonomous Driving Systems. *IFAC-PapersOnLine* **2023**, *56*, 2202–2213. [CrossRef]
5. ASAM. ASAM OpenSCENARIO®. Available online: https://www.asam.net/standards/detail/openscenario/v200/ (accessed on 16 December 2025).
6. ASAM. ASAM OpenDRIVE®. Available online: https://www.asam.net/standards/detail/opendrive/ (accessed on 16 December 2025).
7. ASAM. ASAM OpenMATERIAL® 3D. Available online: https://www.asam.net/standards/detail/openmaterial/ (accessed on 16 December 2025).
8. Temmen, T.; Meyer, M.A.; Wachtmeister, L.; Zabihi, M.; Kugler, C.; Christiaens, S.; Rumpe, B.; Andert, J. Application of Model-Based Systems Engineering Methods in Virtual Homologation Procedures for Automated Driving Functions. In *Automatisiertes Fahren 2024*; Heintzel, A., Ed.; Springer Fachmedien Wiesbaden: Wiesbaden, Germany, 2024; pp. 1–10. [CrossRef]

9. Meyer, M.A.; Silberg, S.; Granrath, C.; Kugler, C.; Wachtmeister, L.; Rumpe, B.; Christiaens, S.; Andert, J. Scenario- and Model-Based Systems Engineering Procedure for the SOTIF-Compliant Design of Automated Driving Functions. In *2022 IEEE Intelligent Vehicles Symposium (IV)*; IEEE: Piscataway, NJ, USA, 2022; pp. 1599–1604. [CrossRef]
10. Wang, Y.; Xing, S.; Can, C.; Li, R.; Hua, H.; Tian, K.; Mo, Z.; Gao, X.; Wu, K.; Zhou, S.; et al. Generative AI for Autonomous Driving: Frontiers and Opportunities. *arXiv* **2025**, arXiv:2505.08854. [CrossRef]
11. Greff, K.; Belletti, F.; Beyer, L.; Doersch, C.; Du, Y.; Duckworth, D.; Fleet, D.J.; Gnanapragasam, D.; Golemo, F.; Herrmann, C.; et al. Kubric: A Scalable Dataset Generator. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 3739–3751. [CrossRef]
12. Feng, T.; Wang, W.; Quan, R.; Yang, Y. Shape2Scene: 3D Scene Representation Learning Through Pre-training on Shape Data. *arXiv* **2024**, arXiv:2407.10200. [CrossRef]
13. Richter, S.R.; Vineet, V.; Roth, S.; Koltun, V. Playing for Data: Ground Truth from Computer Games. In *Computer Vision–ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer International Publishing: Cham, Switzerland, 2016; Volume 9906, pp. 102–118. [CrossRef]
14. Nowruzi, F.E.; Kapoor, P.; Kolhatkar, D.; Hassanat, F.A.; Laganiere, R.; Rebut, J. How Much Real Data Do We Actually Need: Analyzing Object Detection Performance Using Synthetic and Real Data. *arXiv* **2019**, arXiv:1907.07061. [CrossRef]
15. Wen, B.; Xie, H.; Chen, Z.; Hong, F.; Liu, Z. 3D Scene Generation: A Survey. *arXiv* **2025**, arXiv:2505.05474. [CrossRef]
16. Bai, X.; Luo, Y.; Jiang, L.; Gupta, A.; Kaveti, P.; Singh, H.; Ostadabbas, S. Bridging the Domain Gap between Synthetic and Real-World Data for Autonomous Driving. *arXiv* **2023**, arXiv:2306.02631. [CrossRef]
17. Sun, P.; Kretzschmar, H.; Dotiwalla, X.; Chouard, A.; Patnaik, V.; Tsui, P.; Guo, J.; Zhou, Y.; Chai, Y.; Caine, B.; et al. Scalability in Perception for Autonomous Driving: Waymo Open Dataset. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 2443–2451. [CrossRef]
18. Koopman, P.; Widen, W. Safety Ethics for Design and Test of Automated Driving Features. *IEEE Des. Test* **2024**, *41*, 17–24. [CrossRef]
19. Hu, A.; Russell, L.; Yeo, H.; Murez, Z.; Fedoseev, G.; Kendall, A.; Shotton, J.; Corrado, G. GAIA-1: A Generative World Model for Autonomous Driving. *arXiv* **2023**, arXiv:2309.17080. [CrossRef]
20. Projekt, P. PEGASUS METHOD An Overview. Available online: https://www.pegasusprojekt.de/files/tmpl/Pegasus-Abschlussveranstaltung/PEGASUS-Gesamtmethode.pdf (accessed on 16 December 2025).
21. Epic Games, I. Unreal Engine. Available online: https://www.unrealengine.com/en-US (accessed on 16 December 2025).
22. Dosovitskiy, A.; Ros, G.; Codevilla, F.; Lopez, A.; Koltun, V. CARLA: An Open Urban Driving Simulator. In *Proceedings of the 1st Annual Conference on Robot Learning*; Levine, S., Vanhoucke, V., Goldberg, K., Eds.; PMLR: Cambridge, MA, USA, 2017; Volume 78, pp. 1–16.
23. Feng, T.; Wang, W.; Yang, Y. A Survey of World Models for Autonomous Driving. *arXiv* **2025**, arXiv:2501.11260. [CrossRef]
24. Yang, Z.; Chai, Y.; Anguelov, D.; Zhou, Y.; Sun, P.; Erhan, D.; Rafferty, S.; Kretzschmar, H. SurfelGAN: Synthesizing Realistic Sensor Data for Autonomous Driving. In *Proceedings of the 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 13–19 June 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 11115–11124. [CrossRef]
25. Gaidon, A.; Wang, Q.; Cabon, Y.; Vig, E. VirtualWorlds as Proxy for Multi-object Tracking Analysis. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 4340–4349. [CrossRef]
26. Cabon, Y.; Murray, N.; Humenberger, M. Virtual KITTI 2. *arXiv* **2020**, arXiv:2001.10773. [CrossRef]
27. Xie, Z.; Zhang, J.; Li, W.; Zhang, F.; Zhang, L. S-NERF: NEURAL RADIANCE FIELDS FOR STREET. *arXiv* **2023**, arXiv:2303.00749. [CrossRef]
28. Feng, T.; Wang, W.; Yang, Y. Gaussian-Based World Model: Gaussian Priors for Voxel-Based Occupancy Prediction and Future Motion Prediction. Available online: https://openaccess.thecvf.com/content/ICCV2025/papers/Feng_Gaussian-based_World_Model_Gaussian_Priors_for_Voxel-Based_Occupancy_Prediction_and_ICCV_2025_paper.pdf (accessed on 16 December 2025).
29. Richter, S.R.; Hayder, Z.; Koltun, V. Playing for Benchmarks. In *Proceedings of the 2017 IEEE International Conference on Computer Vision (ICCV), Venice, Italy, 22–29 October 2017*; IEEE: Piscataway, NJ, USA, 2017; pp. 2232–2241. [CrossRef]
30. Ros, G.; Sellart, L.; Materzynska, J.; Vazquez, D.; Lopez, A.M. The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 3234–3243. [CrossRef]
31. Li, Y.; Jiang, L.; Xu, L.; Xiangli, Y.; Wang, Z.; Lin, D.; Dai, B. MatrixCity: A Large-scale City Dataset for City-scale Neural Rendering and Beyond. In *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 1–6 October 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 3182–3192. [CrossRef]
32. Kim, S.; Liu, M.; Rhee, J.J.; Jeon, Y.; Kwon, Y.; Kim, C.H. DriveFuzz: Discovering Autonomous Driving Bugs through Driving Quality-Guided Fuzzing. In *Proceedings of the 2022 ACM SIGSAC Conference on Computer and Communications Security, Los Angeles CA USA, 7–11 November 2022*; Association for Computing Machinery: New York, NY, USA, 2022; pp. 1753–1767. [CrossRef]

33. Li, G.; Li, Y.; Jha, S.; Tsai, T.; Sullivan, M.; Hari, S.K.S.; Kalbarczyk, Z.; Iyer, R. AV-FUZZER: Finding Safety Violations in Autonomous Driving Systems. In *Proceedings of the 2020 IEEE 31st International Symposium on Software Reliability Engineering (ISSRE), Coimbra, Portugal, 12–15 October 2020*; IEEE: Piscataway, NJ, USA, 2020; pp. 25–36. [CrossRef]
34. Yang, H.; Zhou, Y.; Chen, T. SimADFuzz: Simulation-feedback Fuzz Testing for Autonomous Driving Systems. *ACM Trans. Softw. Eng. Methodol.* **2025**. [CrossRef]
35. Lin, S.; Chen, F.; Xi, L.; Wang, G.; Xi, R.; Sun, Y.; Zhu, H. TM-fuzzer: Fuzzing Autonomous Driving Systems through Traffic Management. *Autom. Softw. Eng.* **2024**, *31*, 61. [CrossRef]
36. Krautwig, B.; Wans, D.; Li, L.; Temmen, T.; Koch, L.; Eisenbarth, M.; Andert, J. Navigating the Trade-Offs: A Quantitative Analysis of Reinforcement Learning Reward Functions for Autonomous Maritime Collision Avoidance. *J. Mar. Sci. Eng.* **2025**, *13*, 2233. [CrossRef]
37. Zhou, M.; Wang, Y.; Hou, J.; Zhang, S.; Li, Y.; Luo, C.; Peng, J.; Zhang, Z. SceneX: Procedural Controllable Large-Scale Scene Generation. In *Proceedings of the Thirty-Ninth AAAI Conference on Artificial Intelligence and Thirty-Seventh Conference on Innovative Applications of Artificial Intelligence and Fifteenth Symposium on Educational Advances in Artificial Intelligence*; AAAI Press: Washington, DC, USA, 2025. [CrossRef]
38. Raistrick, A.; Lipson, L.; Ma, Z.; Mei, L.; Wang, M.; Zuo, Y.; Kayan, K.; Wen, H.; Han, B.; Wang, Y.; et al. Infinite Photorealistic Worlds Using Procedural Generation. In *Proceedings of the 2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Vancouver, BC, Canada, 17–24 June 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 12630–12641. [CrossRef]
39. Tsirikoglou, A.; Kronander, J.; Wrenninge, M.; Unger, J. Procedural Modeling and Physically Based Rendering for Synthetic Data Generation in Automotive Applications. *arXiv* **2017**, arXiv:1710.06270. [CrossRef]
40. Wrenninge, M.; Unger, J. Synscapes: A Photorealistic Synthetic Dataset for Street Scene Parsing. *arXiv* **2018**, arXiv:1810.08705. [CrossRef]
41. Sun, T.; Segu, M.; Postels, J.; Wang, Y.; Van Gool, L.; Schiele, B.; Tombari, F.; Yu, F. SHIFT: A Synthetic Driving Dataset for Continuous Multi-Task Domain Adaptation. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022*; IEEE: Piscataway, NJ, USA, pp. 21339–21350. [CrossRef]
42. Liao, Y.; Xie, J.; Geiger, A. KITTI-360: A Novel Dataset and Benchmarks for Urban Scene Understanding in 2D and 3D. *IEEE Trans. Pattern Anal. Mach. Intell.* **2023**, *45*, 3292–3310. [CrossRef]
43. Jocher, G.; Chaurasia, A.; Qiu, J. Ultralytics Yolov8, 2023. Available online: https://docs.ultralytics.com/models/yolov8/#supported-tasks-and-modes (accessed on 16 December 2025).
44. OpenStreetMap contributors. Planet Dump Retrieved from https://planet.osm.org. Available online: https://www.openstreetmap.org (accessed on 16 December 2025).
45. Li, W.; Lin, Z.; Zhou, K.; Qi, L.; Wang, Y.; Jia, J. MAT: Mask-Aware Transformer for Large Hole Image Inpainting. In *Proceedings of the 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), New Orleans, LA, USA, 18–24 June 2022*; IEEE: Piscataway, NJ, USA, 2022; pp. 10748–10758. [CrossRef]
46. Bynagari, N.B. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. *Asian J. Appl. Sci. Eng.* **2019**, *8*, 25–34. [CrossRef]
47. Everingham, M.; Van Gool, L.; Williams, C.K.I.; Winn, J.; Zisserman, A. The Pascal Visual Object Classes (VOC) Challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
48. Szegedy, C.; Vanhoucke, V.; Ioffe, S.; Shlens, J.; Wojna, Z. Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Las Vegas, NV, USA, 27–30 June 2016*; IEEE: Piscataway, NJ, USA, 2016; pp. 2818–2826. [CrossRef]
49. Open Geospatial Consortium. CityGML. Available online: https://www.ogc.org/standards/citygml/ (accessed on 16 December 2025).
50. Open Geospatial Consortium; Kolbe, T.H.; Kutzner, T.; Smyth, C.S.; Nagel, C.; Roensdorf, C.; Heazel, C. OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard. Available online: https://docs.ogc.org/is/20-010/20-010.html (accessed on 16 December 2025).
51. Schwab, B.; Beil, C.; Kolbe, T.H. R:Trån. Zenodo. 2023. Available online: https://zenodo.org/records/7702313 (accessed on 16 December 2025).
52. Schwab, B.; Beil, C.; Kolbe, T.H. Spatio-Semantic Road Space Modeling for Vehicle–Pedestrian Simulation to Test Automated Driving Systems. *Sustainability* **2020**, *12*, 3799. [CrossRef]
53. Schwab, B.; Kolbe, T.H. VALIDATION OF PARAMETRIC OPENDRIVE ROAD SPACE MODELS. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2022**, *X-4/W2-2022*, 257–264. [CrossRef]
54. Biljecki, F.; Ledoux, H.; Stoter, J. GENERATION OF MULTI-LOD 3D CITY MODELS IN CITYGML WITH THE PROCEDURAL MODELLING ENGINE RANDOM3DCITY. *ISPRS Ann. Photogramm. Remote Sens. Spat. Inf. Sci.* **2016**, *IV-4/W1*, 51–59. [CrossRef]
55. Wysocki, O.; Schwab, B.; Beil, C.; Holst, C.; Kolbe, T.H. Reviewing Open Data Semantic 3D City Models to Develop Novel 3D Reconstruction Methods. *Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci.* **2024**, *XLVIII-4-2024*, 493–500. [CrossRef]

56. Wysocki, O.; Schwab, B.; Willenborg, B. OloOcki/Awesome-Citygml: Release. Zenodo. 2022. Available online: https://zenodo.org/records/5899096 (accessed on 16 December 2025).
57. 3DHaupt. Room Door Animation. Available online: https://sketchfab.com/3d-models/room-door-animation-e85909e80635445b92fb80772af80c84 (accessed on 16 December 2025).
58. EFX. HIGH QUALITY TREE 6/6. Available online: https://sketchfab.com/3d-models/high-quality-tree-66-221b8943cc514162a05be24b7ba35430 (accessed on 16 December 2025).
59. Fan, A. NYC Streetlight 1. Available online: https://sketchfab.com/3d-models/nyc-streetlight-1-456df4ed4b3241938695eef72b941123 (accessed on 16 December 2025).
60. Fan, A. Virginia Streetlight 1. Available online: https://sketchfab.com/3d-models/virginia-streetlight-1-d2ae505054a142a596b1386d0b5f1f98 (accessed on 16 December 2025).
61. Megascans, Q. Quixel Megascans. Available online: https://www.fab.com/sellers/Quixel%20Megascans (accessed on 16 December 2025).
62. De Ioris, R. glTFRuntime: Unreal Engine Plugin for Loading glTF Files at Runtime, 2025. Available online: https://github.com/rdeioris/glTFRuntime (accessed on 16 December 2025).
63. Temmen, T.; Debougnoux, J.; Li, L.; Krautwig, B.; Brinkmann, T.; Eisenbarth, M.; Andert, J. TTemmen/LGM_SemanticSegmentationDataset_1: V1.1. Zenodo. 2025. Available online: https://zenodo.org/records/17949756 (accessed on 16 December 2025).
64. He, L.; Aliaga, D. GlobalMapper: Arbitrary-Shaped Urban Layout Generation. In *Proceedings of the 2023 IEEE/CVF International Conference on Computer Vision (ICCV), Paris, France, 1–6 October 2023*; IEEE: Piscataway, NJ, USA, 2023; pp. 454–464. [CrossRef]
65. Li, L.; Brinkmann, T.; Temmen, T.; Eisenbarth, M.; Andert, J. Multi-Agent Scenario Generation in Roundabouts with a Transformer-enhanced Conditional Variational Autoencoder. *arXiv* **2025**, arXiv:2510.24671. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.