Advanced Engineering Informatics 71 (2026) 104410

Contents lists available at ScienceDirect

Advanced Engineering Informatics

journal homepage: www.elsevier.com/locate/aei

Full length article

SynthRoads: A framework for comprehensive enrichment of the Scan-to-Twin process for road space modeling with synthetic data

David Crampen *, Marcel Hein, Jörg Blankenbach

Geodetic Institute and Chair of Computing in Civil Engineering and Geoinformation Systems, RWTH Aachen University, Mies-van-der-Rohe Straße 1, Aachen, 52074, North-Rhine-Westfalia, Germany

ARTICLE INFO

Keywords:
Automation
Synthetic point cloud
Scan-to-Twin
Road infrastructure
Digital Twin
Deep learning

ABSTRACT

Digital twins of the built environment promise major efficiency gains in managing and maintaining infrastructure. A main component of Digital Twin systems is an information-rich, up-to-date, virtual representation of the physical scene. Deriving such an as-is geometric–semantic model (GSM) of the target environment that is both timely and realistic poses a substantial challenge. In particular, the required data aggregation and processing remain substantial obstacles for robust and highly automated generation. The Scan-to-Twin workflow aims to automatically produce an as-is GSM from reality-capture data, but deep learning–based automation still depends on costly manual annotations. We present SynthRoads, a framework that synthetically generates multi-modal data for every step of Scan-to-Twin, enabling deep learning for point-cloud semantic segmentation and geometric extraction in road environments. SynthRoads combines modular model generation with synthetic laser scanning to efficiently create annotated point clouds, geometry parameters, and GSMs. While broadly applicable, we demonstrate significant performance gains for road semantic segmentation and road centerline regression in Scan-to-Twin use cases. Our framework is highly adaptable, leveraging the realism of a game engine and a procedural modeling approach to efficiently generate diverse and realistic road scenes.

1. Introduction

Scan-to-Twin targets the purposeful generation of tailored 2D and 3D digital models for Digital Twin applications. The resulting as-is geometric–semantic model (GSM) is generated according to the use case requirements of a Digital Twin and acts as an efficient foundation for the development of the Digital Twin system [1]. While building information modeling (BIM) mainly targets planning and construction, a Digital Twin of the built environment focuses on impacts and interactions on an existing structure. In general, a Digital Twin consists of a physical asset, the virtual twin, sensor data inputs from the real environment into the virtual environment, data analyses, and a feedback system into the physical world to optimize or automate a specific process [2]. Automating the model generation step is essential, as multiple use cases can target the same physical environment but may require different digital representations. Creating such multi-scale representations manually is infeasible due to extensive modeling effort. Additionally, since the physical environment is subject to continuous change, which has to be reflected in the virtual twin, manual modeling lacks the required flexibility and speed [3]. A manually created model also loses the direct relation to the reality capture data, which may be important, especially for change detection tasks. Thereby, detected changes based on the comparison of survey data at different times lack clear implications for the objects in the model that have to be updated, especially if the point cloud segment of the object in question is incomplete [4]. However, establishing a robust approach for automatically deriving as-is GSMs is complex [5]. A typical step relying on artificial intelligence in the Scan-to-Twin process is semantic segmentation, where the point cloud or image data is segmented into point or pixel clusters representing specific objects, such as road furniture (e.g., guardrails, posts, gantries, and signs), road markings, or the road surface. The subsequent geometry extraction from these semantic objects could, in principle, also be conducted by supervised learning if accurate ground truth data for geometry parameters such as the road centerline, the location and shape of spalling and rutting, or even the number of lanes on each carriageway were available at large scale. Creating such ground truth from point clouds, however, requires substantial manual effort. Public datasets that provide paired point clouds and lane-level geometry labels suitable for supervised geometry extraction remain scarce; for example, [6], which specifically targets lane-level geometry extraction from mobile laser scanning (MLS) point clouds and focuses on road-marking polylines rather than full pavement geometry. For this reason, most works conduct semantic

* Corresponding author.
E-mail address: crampen@gia.rwth-aachen.de (D. Crampen).

https://doi.org/10.1016/j.aei.2026.104410
Received 23 October 2025; Received in revised form 6 January 2026; Accepted 31 January 2026
Available online 9 February 2026

segmentation as a first step to isolate the pavement and subsequently extract the geometry [7], as was done in [8,9]. For geometry extraction, most approaches rely on different sets of conventional methods, such as fitting techniques or filtering, as used in [10,11], which, however, might fail to generalize to heterogeneous environments typical of Digital Twin applications. To the best of our knowledge, end-to-end carriageway centerline regression directly from raw 3D point clouds, without intermediate rasterization or skeletonization, has not yet been systematically studied in the Scan-to-Twin context.

This indicates that the data requirements for semantic segmentation models as well as geometry regression models for 3D point clouds constitute a major issue for progressing to more comprehensive 3D reasoning [12]. An additional obstacle is the high effort of data acquisition, preprocessing, and annotation. More issues arise when acquiring point clouds in the road environment due to safety restrictions, which require special precautions and approvals from highway authorities. Currently, mobile mapping systems (MMS) are the only feasible large-scale acquisition technology for large-scale 3D surveying of roads in use. However, the use of terrestrial laser scanning (TLS) and unmanned aerial vehicle (UAV)-based laser scanning is feasible on closed roads. The completeness of the captured data depends on many factors, such as the amount of traffic leading to scanning shadows for vehicles close to the MMS during surveys or the height of the scanner system on the survey platform, which might become problematic for relatively high barriers that limit the field of view of a scanner and lead to scanning shadows. While real data is essential for training a well-generalizing segmentation model, augmenting it with representative synthetic data can enhance generalization and mitigate the challenges associated with laborious data acquisition and preprocessing [13].

In response to these constraints, we propose *SynthRoads*, a synthetic data generation framework that provides controllable, richly annotated 3D point clouds for semantic segmentation and downstream geometry extraction in road environments.

Unlike prior synthetic point cloud pipelines that either emphasize scan simulation without photorealistic rendering and material-driven appearance or focus on visually realistic driving simulators without exposing consistent infrastructure parameters and Scan-to-Twin outputs, SynthRoads closes this gap by explicitly enforcing cross-modal, end-to-end consistency across the Scan-to-Twin chain and generates paired, mutually consistent modalities of: (i) annotated point clouds with realistic scan artifacts and radiometric features, (ii) the corresponding geometry parameters (e.g., road centerlines and corridor-level descriptors), and (iii) the resulting 3D geometric–semantic models (GSMs). This pairing enables supervision not only for semantic segmentation but also for direct geometry regression from raw point clouds.

SynthRoads leverages domain knowledge about road layouts and laser scanning to automatically generate synthetic GSMs from randomized parameters and high-fidelity virtual scan simulations. To ensure realism and plausibility, parameter sampling is constrained by German highway design prescriptions (RAA) [14]. Beyond increasing training data volume for point-cloud semantic segmentation, the availability of paired geometry parameters provides direct input for subsequent Scan-to-Twin steps such as geometry extraction and model generation. Moreover, full-pipeline synthetic datasets facilitate additional use cases, including uncertainty quantification in Scan-to-Twin workflows [15]. The framework is modular by design, enabling adaptation to other domains or scenario configurations by modifying individual components, including the parameter module, the model generation module, and the Unreal Engine-based virtual scanner module [16].

Our main contributions are:

1. a modular road-scene generator with rule-constrained parameter sampling (RAA),
2. an Unreal Engine 5 scan-simulation plugin that produces feature-rich point clouds (RGB, intensity, and material-related parameters),
3. an experimental evaluation demonstrating segmentation gains on real highway data through synthetic augmentation, and
4. to the best of our knowledge, the first study evaluating end-to-end road centerline regression directly from raw 3D point clouds using synthetic geometry-parameter supervision.
5. a synthetic dataset consisting of annotated point clouds and geometry parameters for road related Scan-To-Twin.

The remainder of this paper is organized as follows. Section 2 reviews related work, Section 3 derives research gaps and research questions, and Section 4 details the SynthRoads methodology. Section 5 presents the experimental setup, and Section 6 reports results for semantic segmentation (PointNeXt [17]) and deep learning-based centerline regression using two lightweight architectures. Finally, Section 7 concludes the paper and outlines future work.

2. Related work

In the following section, we will introduce previous works that dealt with synthetic point cloud data generation and review findings on data augmentation for semantic segmentation applications to identify the existing research gaps we aim to close with SynthRoads.

2.1. Synthetic point cloud generation approaches

There are different approaches for synthetic point cloud generation, which can be categorized into four groups:

1. Scan simulations
2. Generative models (GMs)
3. Scene re-combination
4. 3D model-based surface sampling

Each category has its own respective advantages and disadvantages, which we will discuss in the following subsections.

2.1.1. Scan simulations

Scan simulations mainly target the areas of autonomous driving [18] and earth observation. One of the most popular simulators for autonomous driving is CARLA [19]. While primarily intended as a test suite for autonomous driving simulations, it was also used to generate synthetic point cloud data for scene reconstruction applications. Multiple works have used CARLA to generate synthetic point clouds based on ray casting, such as [20,21], and [22]. Other works on scan simulations in autonomous driving that developed standalone simulations are [23, 24]. A popular simulator for earth observations is Helios++ [25], which targets the simulation of geodetic surveys on varying platforms. The framework allows use for several different applications, such as testing of different scan settings, comparison of simulated data to real data, virtual laser scanning for data augmentation for supervised machine learning, or testing for novel algorithms. While Helios++ does not rely on a game engine for a realistic physical environment as CARLA does, the focus lies on realistic laser scan simulation, incorporating factors such as beam divergence, full waveform scanning, and an exchangeable deflector mechanism. A survey in Helios++ requires definition of the 3D scene, the scanner platform, the scanner itself, and the trajectory or scanner standpoints for MLS platforms or TLS platforms, respectively, which are structured into so-called legs. Helios++ is capable of retrieving the scan features intensity and the number of returns and supports scene partitioning to create per-point class labels. However, while color can be retrieved, it is sampled from assigned materials of objects and is therefore not able to capture realistic colors affected by lighting. Another example was presented in [26], where Unreal Engine was used to build a training suite for TLS. However, this work is limited to relatively few scenes and focuses on teaching the correct placement of markers and scan positions. Scan simulations in general are well suited to generate realistic synthetic point clouds due to the high level

of control over the simulation, mimicking a real survey mission to yield characteristic scan patterns and scan shadows. However, they also require the highest effort to generate diverse scenes, since models have to be created beforehand and the survey conducted has to be set up for each individual scene.

2.1.2. GMs

The use of GMs for synthetic point cloud generation is a promising approach that could potentially largely improve the performance of semantic segmentation models by directly generating 3D point clouds from noise, without the need for any previous modeling. Several works have dealt with different model architectures such as generative adversarial auto-encoders (GANs), variational auto-encoders (VAEs), or diffusion-based models for point cloud generation. In [27], a GAN was implemented to generate single object instances based on the ShapeNet dataset [28], and in [29], a diffusion model was used to generate single objects from ShapeNet, ModelNet10, and ModelNet40 [30]. Another work compared different generative models to augment point cloud-based object recognition in driving scenarios, where Gaussian Mixture Models and variational auto-encoders were tested for their capability of generating different traffic-related objects [31]. They evaluated the impact of the generated examples on the point cloud data of the nuScenes dataset [32] using a PointNet model [33], which led to improvements in the F1 score of up to 4.6% for L-GAN-based generations [34]. Upsampling augmentation could potentially be used to densify sparse point clouds using GMs, as in [35]. A general issue of GMs for the generation of whole scenes is the sheer complexity in the composition of large-scale scenes, which might be a major reason why previous works focus on the generation of single objects instead of performing scene generation. This makes GMs useful only in combination with a different approach that creates the basic scene layout, which might then be populated with GM-generated objects for our use case.

2.1.3. Scene re-combination

Another common approach for synthetic data generation is scene recombination. An existing scene acts as a foundation, and existing scanned objects are either moved within the scene or augmented by point clouds of objects sampled from 3D models or extracted from other scans. [36] used this approach to generate new scenes of level crossings by combining both real point cloud segments and synthetic point cloud segments generated from 3D mesh models. One potential issue in scene recombination, however, is the potential creation of unrealistic survey scenarios, where scan shadows and scan patterns might mismatch in comparison to real data. An additional potential issue is mixing up different scan sources with inherent domain shifts, which may lead to untrackable issues, especially concerning scanner features such as intensity.

2.1.4. 3D model-based surface sampling

A comparatively simple approach for generating synthetic point clouds is sampling points from a 3D model's surface. While it is quite straightforward, the degree of realism is relatively low, since aspects such as scan shadows, scan patterns, or occlusions are usually not included in existing versions of this approach. Most studies applying 3D model-based surface sampling are related to indoor semantic segmentation, such as [37,38]. Both works created point clouds from 3D models and achieved a performance boost when combining real and synthetic data during training. In [39], the authors created synthetic point clouds of bridges to enrich the training data of a deep-learning model for semantic segmentation. Though applicable much quicker than scan simulation, the degree of realism tends to be lower than that of scan simulations and lacks characteristic scan features such as intensity of multiple reflections.

2.2. Data augmentation for deep learning approaches

Most synthetic point cloud generation approaches are compared to standard data augmentation methods for point clouds, which are a research topic of their own with many works dealing with wide-ranging approaches to augment point cloud data without direct generation of synthetic data but by transforming the existing real data in various ways. In [40], the authors differentiate between basic augmentation methods and specialized augmentation for point clouds. Basic data augmentation comprises affine transformations such as translation, rotation, scaling, or shear. Other basic methods are dropping points, jittering, or ground truth sampling, which is equivalent to what we refer to as scene recombination with real object instances added to an existing scene. Adding point cloud instances of underrepresented objects may help with issues regarding class imbalances but requires robust measures of instance placements to prevent unrealistic placing, as discussed in Section 2.1.3. Specialized point cloud augmentation comprises different methods that mainly target specific use cases or target environments, such as mixing two instances of the same or of different classes to form a new object, which, however, is also only applicable on the instance level. Another interesting method is domain augmentation, where training data is changed to represent a different domain. A simple example is simulating different weather conditions in the data, as done by [41]. Though such augmentations are much more common in autonomous driving applications, where obstacles have to be robustly detected independently from varying weather conditions, this approach could potentially also benefit the robustness of Scan-to-Twin-related semantic segmentation approaches, since it introduces another dimension of variance into the training data that directly affects point distributions, reflectance, and texture. Instance deformation is used to deform instances of objects to push a model towards generalizing class detection to diverse shapes, as done in [42]. A similar approach would be completion augmentation, where GANs could be employed to reconstruct missing parts of scanned instances to create a complete representation [43], while another approach would be generating data with a GM using data from a different modality, as in [44], where depth images were used with a GM to create dense pothole point cloud data for pavement damage segmentation.

3. Research gaps

Following the review of previous works, we identified several gaps in our target research field of synthetic data generation for scene-level data augmentation of road environments. The first issue of general data augmentation arises from the scene's inherent placement rules for new object instances, which are restricted to on-road objects such as vehicles or barriers and roadside objects such as traffic signs, guardrails, or vegetation. Additionally, the object orientation for on-road objects has to follow the driving direction in our highway scenarios. Formulating these kinds of relationships requires knowledge about the scene, which in most works is linked to an underlying 3D model providing this information. However, there is no fast and comprehensive way to automatically generate varying road scenes, specifically targeting synthetic data generation (1). Generating realistic and information-rich synthetic point clouds to this point is mostly limited to one of two options. Either synthetic laser scanning simulators are dealing with the scanning process itself and disregard important information for semantic segmentation, such as realistic lighting and color retrieval, or a simulation provides realistic shading, as in CARLA, but targets autonomous driving scenarios with limited applicability for Scan-to-Twin workflows. For example, there is no approach that can aggregate both a synthetic labeled point cloud and the road-related geometry parameters such as the centerline, road width, road boundary, and object placements that would enable augmenting supervised learning methods for geometry extraction as well as data augmentation for semantic segmentation (2). There is no comprehensive framework combining both automatic highly variable road model generation and synthetic laser scanning simulation to optimize the Scan-to-Twin pipeline as a whole (3). For this reason, we aim to answer the following research questions in this contribution:

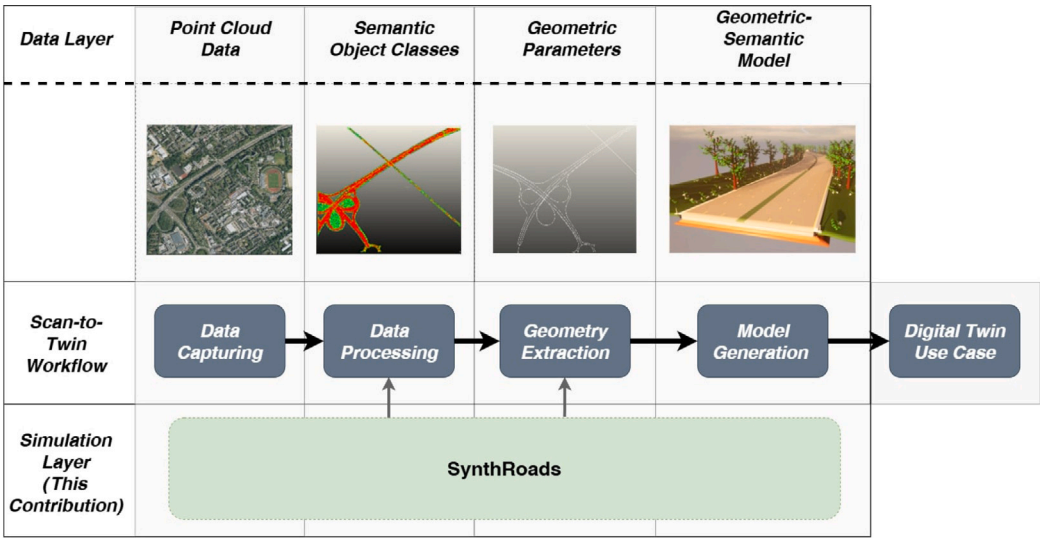

Fig. 1. Overview of connection between the SynthRoads framework and Scan-to-Twin.

1. How can synthetic point cloud data be generated for increasing robustness and generalization of semantic segmentation models of the road? (RQ1)
2. How can model generation be automated to provide synthetic scanning methods with large amounts of different scenes? (RQ2)
3. How can the Scan-to-Twin process be comprehensively supported with a pipeline for generating complete sets of corresponding, synthetic point clouds, annotations, geometry parameters and models? (RQ3)

4. Methodology

In the following section, we will introduce each component of our framework, SynthRoads, which comprises a parameter generator module, a road modeling module that we introduced in [45] in a preliminary stage, and the core of this work, an Unreal Engine 5-based scan simulator, capable of directly using models created by the model generator and retrieving a virtual point cloud with color values, reflectance values, and even material properties of the captured surfaces. This set of modules enables the collection of fully synthetic Scan-to-Twin datasets consisting of 3D models, geometry parameters, and labeled point clouds. Figs. 1 and 2 depict our workflow in conjunction with Scan-to-Twin. The starting point of SynthRoads was chosen in a reverse engineering fashion. Starting with creating a model that conforms to the desired outcome of Scan-to-Twin increases the controllability of uncertainties that are introduced in the workflow. Having a 'true' model as a basis for synthetic point cloud generation is crucial, since a core concept of Scan-to-Twin, as we defined in [46], is the aggregation and measurement of the model uncertainty. This facilitates the calibration of the Digital Twin use case utilizing the digital GSM, which in turn leads to more accurate results and better estimates for the decision support's inherent uncertainty.

4.1. Model generation

Our model generation workflow consists of two parts: first, we generate a geometry parameter set with a Python module and then use this set for actual modeling of the road in the SideFX Houdini software for procedural modeling.

4.1.1. Parameter generator

The Python-based geometry parameter generator is capable of automatically generating a road setup with a centerline acting as a backbone for the road. The road curvature is constrained by the German highway construction codes RAA [14] and limits the minimum and maximum curvature of the centerline. The number of lanes and their width on each carriageway can be specified, and the topography close to the road boundary can be either generated or defined using an existing digital elevation model (DEM). We further differentiate between point objects and line objects for road furniture, such as traffic signs, shield gantries (overhead sign gantries), posts, and guardrails or barriers, respectively, which are placed either randomly or according to specified safety requirements for highways with guardrails required in curves or separating the carriageways. To manage these options, we developed a graphical user interface (GUI) in which parts of the configuration of a road can be manually adjusted or fully automatically regenerated with random values within prescribed limits. This GUI is depicted in Fig. 3. Although there are options to directly model roads with CityEngine [47] or RoadRunner [48] for example, these solutions firstly rely on mainly manual interaction with the software to model a road and are not capable of exposing road geometry parameters directly, in contrast to our two-step approach. An additional important argument for this approach is the fact that in the Scan-to-Twin workflow, model generation relies on the extractable parameters, which might lead to infeasible solutions if the necessary parameter set for modeling a road is unknown. Therefore, we are not just able to define diverse parameter sets that are sufficient for model generation, but we also deliver an approach that actually generates consistent models from a set of parameters. In turn, the successful extraction of such geometry parameters from a semantically segmented point cloud demonstrates the feasibility of model generation based on a minimal set of required parameters. We export the parameter set to JSON, choosing a relatively compact structure similar to the CityJSON standard [49].

4.1.2. Modeling module

The procedural modeling module was implemented in SideFX Houdini. The main reason for that choice is seamless integratability into Unreal Engine, allowing the import of the road modeling module as an interactive asset, which then avoids creating an interface between the model and the scanning process. This allows directly generating new roads by simply loading a new JSON-based parameter set in Unreal Engine, while Houdini exposes all relevant parameters, allowing adjusting the parameters of the model within the engine. Fig. 4 depicts an example of a generated road model with the modeling module in Unreal Engine. The modeling of road furniture assets such as road signs, shield gantries, guiding posts, and guardrails is referenced in the schema by

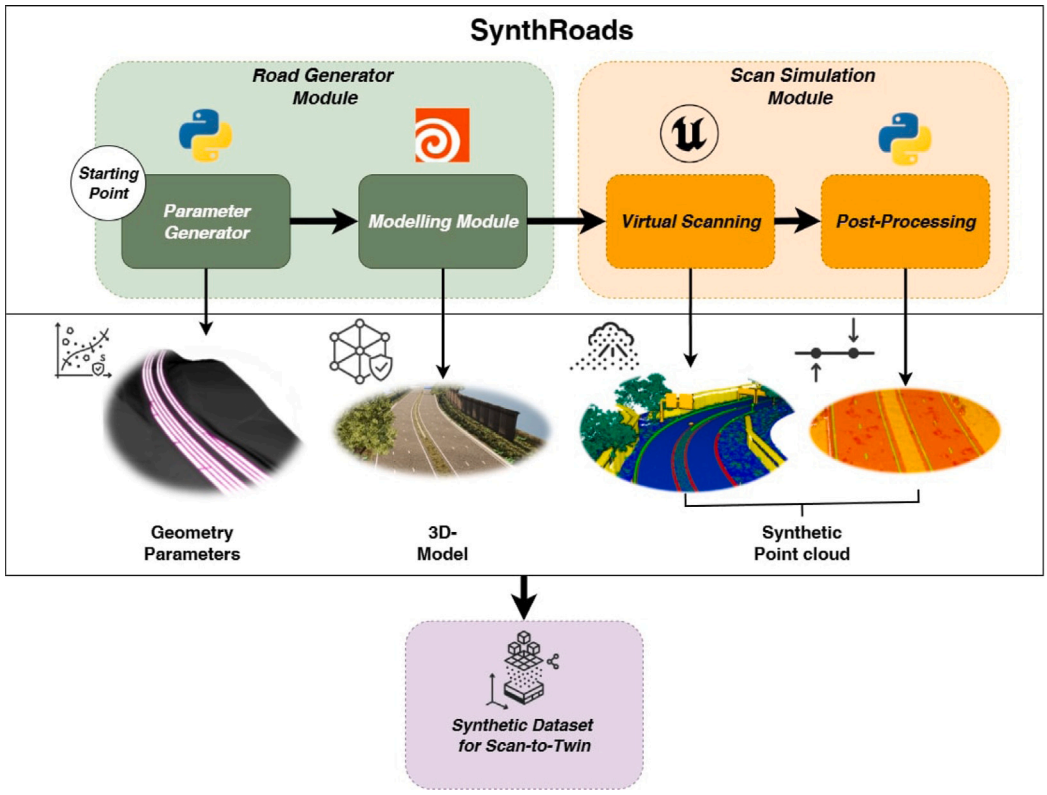

Fig. 2. Overview of the SynthRoads process pipeline.

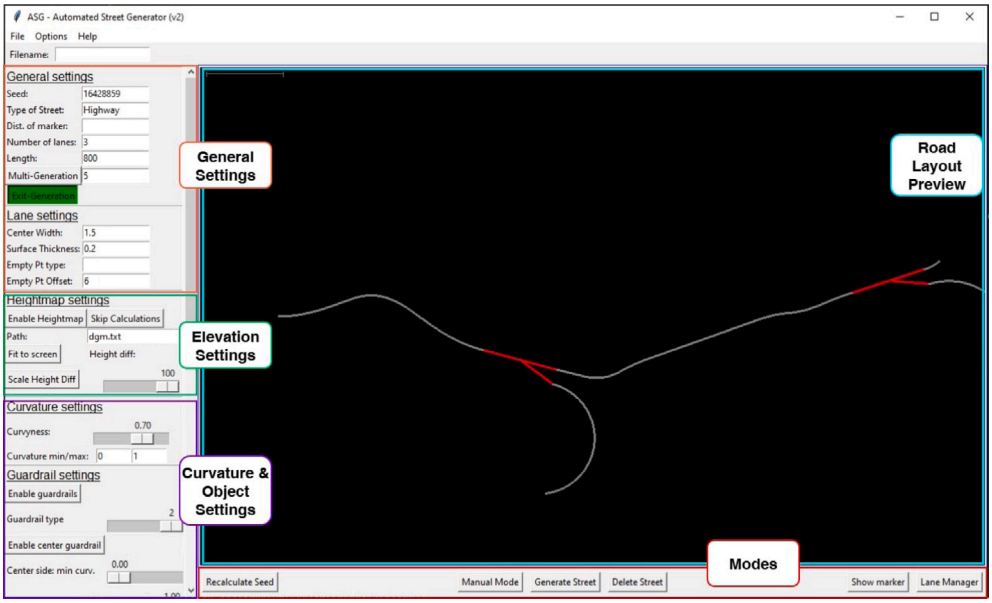

Fig. 3. Example of the revised road generation module with two generated turnouts.

positions relative to the centerline, with additional roadside information. This logic corresponds to the structure of the IfcRoad standard, where assets can be placed at stations of an underlying IfcAlignment entity via IfcLinearPlacement. The process is designed modularly to allow replacement of single object generation and placement through further development. In that way, we introduced turnouts (short lane divergences such as emergency bays or small exits) into the preliminary system, adding a new dimension to the possible road configurations. As well as an additional flexible placement module, which allows easily adding more assets into the scene for increased diversity. However, it has to be noted that the scan simulation does not depend on models generated by the modeling module. Therefore, other assets or full scenes can be scanned similarly if they can be imported as static mesh objects into Unreal Engine. The classification corresponds to component tags, which can be manually assigned in Unreal Engine. The modeling module automatically assigns component tags that are captured by the

Fig. 4. Example of a road model generated in Unreal Engine.

scan simulation via a mapping array that can be easily extended to add new object classes.

4.2. Scan simulation

The scan simulation relies on a ray casting process, where a scanner instance is placed at a position of a spline curve or polyline, which is automatically created when baking the model in Unreal Engine. This scan path corresponds to the centerline either directly, by simply defining a flight altitude, or can be set to a smoothed option or a zigzag pattern, where the path created is represented by a polynomial curve, which is fit to the centerline. The scanner provides options for beam divergence, multiple returns, noise, and the scan resolution, which corresponds to the angle increment of the scanner and field of view, which is adjustable. For our experiments, we defined the scanner parameters according to those of a Riegl miniVUX 3 UAV scanner [50] in the standard configuration (beam divergence: 1.6 × 0.5 mrad, angle increment: 0.018°), as the real surveys were conducted with the same scanner. The intensity profile is calculated using either the Lambertian reflectance model or the Oren-Nayar reflectance model. The Lambertian reflectance model, Eq. (1) is derived from [51] under the condition of local object homogeneity and adapted with a heuristic for estimating the reflectivity from material texture parameters. We denote I_0 as the emitted beam intensity and R as the surface reflectivity, which is calculated with Eq. (2) from the texture values and albedo, which is computed as the luminance, T_{albedo} with Eq. (3) from the base color of a texture. d is the distance between the scanner and surface, θ is the angle of incidence between the laser beam and surface normal, and α is the atmospheric attenuation coefficient, which is a coefficient for signal strength reduction in the atmosphere through absorption and scattering. We are able to adjust α according to different weather conditions, such as rainfall, fog, or dust, to adapt the scan simulation to different scenarios. The reflectivity of objects is extracted from the specularity ($T_{specularity}$) and metallic ($T_{metallic}$) parameters of the applied material of an object.

$$I_{\text{return}}^{\text{LAM}} = I_0 * \frac{R * \cos\theta}{d^2} * e^{-\alpha * d} \tag{1}$$

$$R = (1 - T_{metallic}) * (T_{specularity} + 0.5 * T_{albedo}) + T_{metallic} * T_{albedo} \tag{2}$$

$$T_{albedo} = 0.3 * Red + 0.59 * Green + 0.11 * Blue \tag{3}$$

The Oren–Nayar reflectance model extends this formulation by explicitly accounting for surface roughness through a micro-facet bidirectional reflectance distribution function (BRDF), which we adopt in the simplified configuration proposed by Carrea et al. [52]. In our implementation, the Oren–Nayar reflectance is obtained by modulating the Lambertian term in Eq. (1) with an angular factor that accounts for surface roughness,

$$I_{\text{return}}^{\text{ON}} = I_0 \frac{R\cos\theta}{d^2}\left(A + B\,\sin\theta\,\tan\theta\right) e^{-\alpha d}, \tag{4}$$

where A and B are functions of the standard deviation of the facet slope distribution σ_{slope} (in radians),

$$A = 1 - \frac{1}{2}\frac{\sigma_{\text{slope}}^2}{\sigma_{\text{slope}}^2 + 0.33}, \tag{5}$$

$$B = 0.45\frac{\sigma_{\text{slope}}^2}{\sigma_{\text{slope}}^2 + 0.09}. \tag{6}$$

In the monostatic TLS configuration considered by Carrea et al. [52], the incoming and outgoing rays are assumed to be collinear, so the bidirectional Oren–Nayar BRDF simplifies to $L_r = R\,E_0\cos\theta\left(A + B\sin\theta\tan\theta\right)$, where E_0 is the irradiance at normal incidence. We incorporate this factor into the LiDAR range equation by replacing the Lambertian $\cos\theta$ term in Eq. (1) with $\cos\theta\left(A + B\sin\theta\tan\theta\right)$, which yields the Oren–Nayar-based return intensity in Eq. (4). The roughness parameter σ_{slope} is mapped from the material roughness of the Unreal Engine texture. In the case of $\sigma_{\text{slope}} \to 0$, we obtain $A \to 1$ and $B \to 0$, so that $I_{\text{return}}^{\text{ON}}$ reduces to the Lambertian reflectance in Eq. (1). Supporting this second reflectance model improves the adaptability of our framework to application domains with more heterogeneous incidence angles and viewing geometries than in our main use case of road surface scanning. A figure comparing the intensity distributions generated by each model on the same example road is included in the supplementary material (Page 8, Fig. 10).

The scan patterns can be adjusted to four different options:

1. Grid
2. Oscillating
3. Rotating
4. Wedge Prism

An additional option to apply dynamic shadows from clouds can be enabled as well to yield more diverse surface colors. The option for multiple returns is achieved by a multi-line trace that registers the permeability of the first hit object and can therefore penetrate objects whose textures indicate permeability, such as vegetation in our case. The decision to not consider reflections for metallic or polished objects targeted efficiency in point cloud generation, with complex reflection and refraction adding a relatively large additional computation cost while adding rather small amounts of relevant realism for our road-related use case. Furthermore, the scene can be enhanced with dynamic objects, such as vehicles driving on the road, which can then be labeled as well to add to realism. These vehicle trajectories can be optionally generated during model cooking and represent the centerlines of

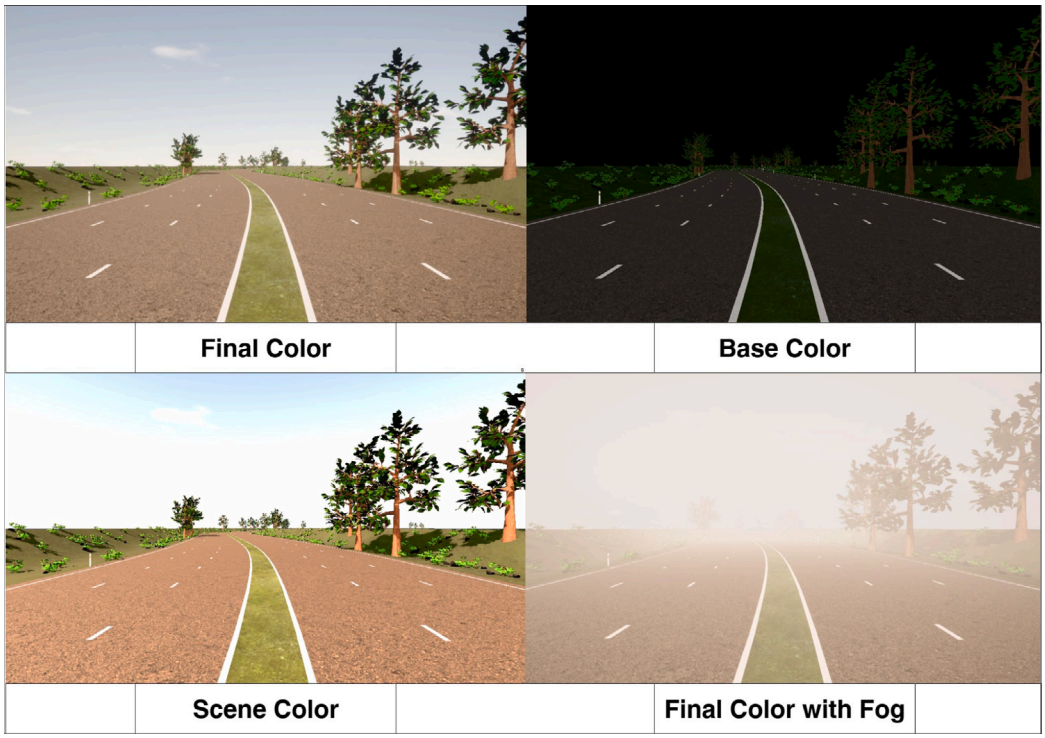

Fig. 5. Examples of different rendering modes in Unreal Engine.

each lane. The vehicles can be assigned either one unified class via tagging the asset or individual classes by vehicle types. An example of the scanning results with added dynamic vehicles is provided in the supplementary material document (Page 8, Fig. 9). The movement speed of the vehicles is reduced by an adjustable factor to match the scanning speed and yield realistic results, as during the scan simulation, the scanner also moves slower than in a real scan.

4.2.1. Texture property retrieval

RGB colors are important features for point cloud semantic segmentation [53]. The virtual scanner captures RGB values, similarly to the real laser scanning practice, where a calibrated camera is mounted to the scanner and captures images that are then used to map color values to the points in the scene. We replicate this process by capturing calibrated render target images from the scanner and mapping the image pixel's UV coordinates to the point cloud dynamically. This approach allows us to not only retrieve RGB values from the rendered Unreal scene but also directly adjust the colors by changing the rendering mode in Unreal. An example of different rendering modes is depicted in Fig. 5.

Additionally, we optionally capture so-called render targets from different layers of the scene. These render targets can be viewed as a virtual camera, capable of capturing arbitrary scene layers, which we leverage to capture the texture parameters directly from the scanner's perspective. Since color is only one of several properties of a textured surface, we are able to also capture texture parameters, such as roughness, metallic values, and specularity, using the same approach, which is necessary for intensity calculation and surface adjustment. Fig. 6 shows examples of the different texture parameters we can automatically capture and map as point-wise features to the point cloud. This retrieval of surface textures has significant advantages. While the original object's hit box can be relatively simple, e.g., a flat surface for the road, the point-wise roughness and surface normal can be extracted and used to adjust points accordingly to achieve a more realistic pattern resembling the microstructure of different materials. In that way, we use a simple roughness post-processing to shift the local point neighborhoods to correspond to the roughness of the terrain and materials captured. An example of the comparison of the synthetic point cloud surface patterns to a real scan with a Riegl miniVUX 3 UAV scan is depicted in Fig. 7 with coloring according to height. In this regard, the possible degree of realism corresponds to the quality of the applied textures, which can be easily exchanged within Unreal Engine and potentially even be randomly assigned during generation to create different surface patterns.

4.2.2. Beam divergence

Our virtual scanner features configurable beam divergence, following the approach from Helios++ [25] and adding further functionality for customization. A cylindrical ray sampling with adaptable resolution can be used to compute the point measurements with respect to the adjustable beam divergence, where the footprint radius is computed based on the distance to an initial hit. Additionally, we feature conical beam divergence to even more closely mirror the physical realism of beam divergence compared to the cylindrical approach. Figs. 8 and 9 depict different examples for our beam divergence with both the cylindrical and the conical methods.

4.3. Centerline regression

As has been stated in Section 1 there is a lack of approaches to directly perform centerline regression from point clouds using a single deep learning approach. To demonstrate the potential to enable deep learning for centerline regression with our new data, we developed two simple neural network architectures for this task. We utilize geospatial data for creating sequences of point cloud sections along the road, which is an adaptation of our roadblock approach from [54]. A critical point lies in avoiding regularities in positioning from this slicing process, which might lead to the model learning undesirable characteristics that stem from our preprocessing. Therefore, we apply a random XY shift within a 12 m window to again crop out a subset of each individual slice, making sure the centerline does not correspond to the geometric center of a respective slice, and for the single-section

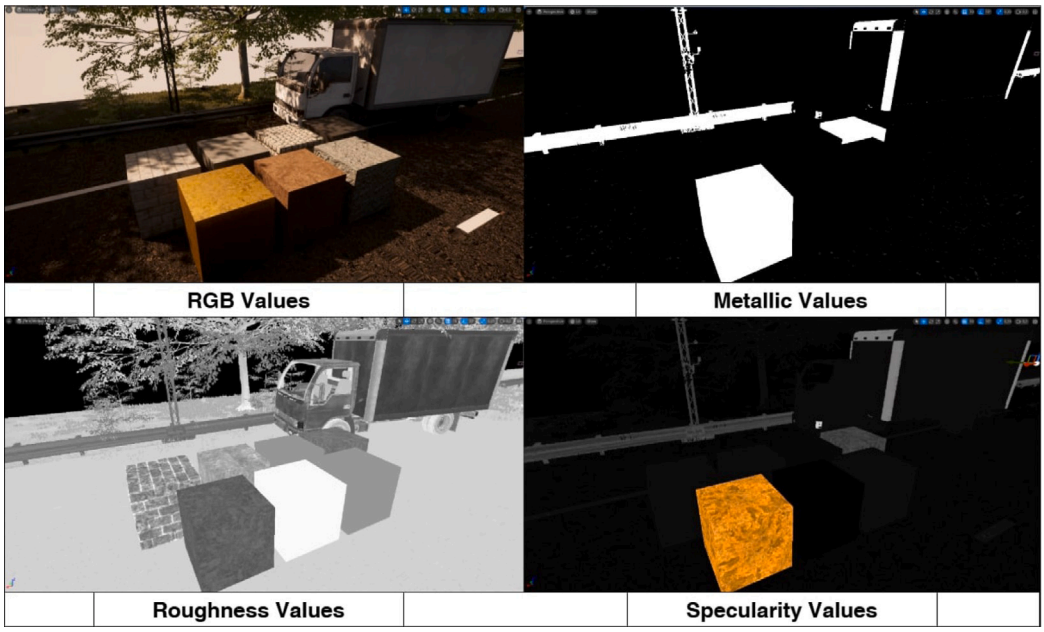

Fig. 6. Example of extracted color and material properties at the examples of different material cubes.

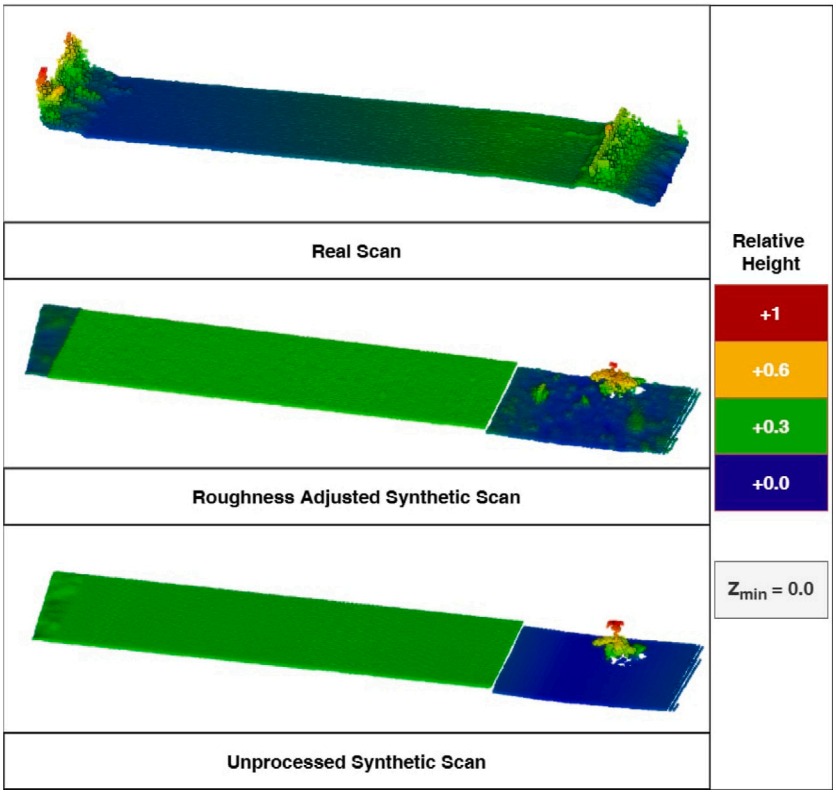

Fig. 7. Comparison between unprocessed and processed synthetic point clouds with real point cloud from UAV laser scan.

model, apply random rotations per slice and Gaussian jittering for both models. The first model uses long-short-term-memory (LSTM) layers and receives a sequence of roadblocks, which are ordered, and outputs predictions for each individual roadblock. The second model processes individual blocks using a simplified self-attention mechanism. Both models initialize 25 kernel points along the orthogonal of the principal direction of the slices to perform a simplified kernel point attention inspired by the Kernel Point Convolution model (KPConv) [55]. While we use fixed kernel points for the sequence model, we used a simple deformable kernel point mechanism for the single-section model.

4.4. Performance metrics

This section summarizes the metrics used to evaluate our experiments. For a given class, we denote true positives (TP), false positives (FP), and false negatives (FN). Based on these counts, we report per-class *precision* (7) and *recall* (8) for the semantic segmentation task, as

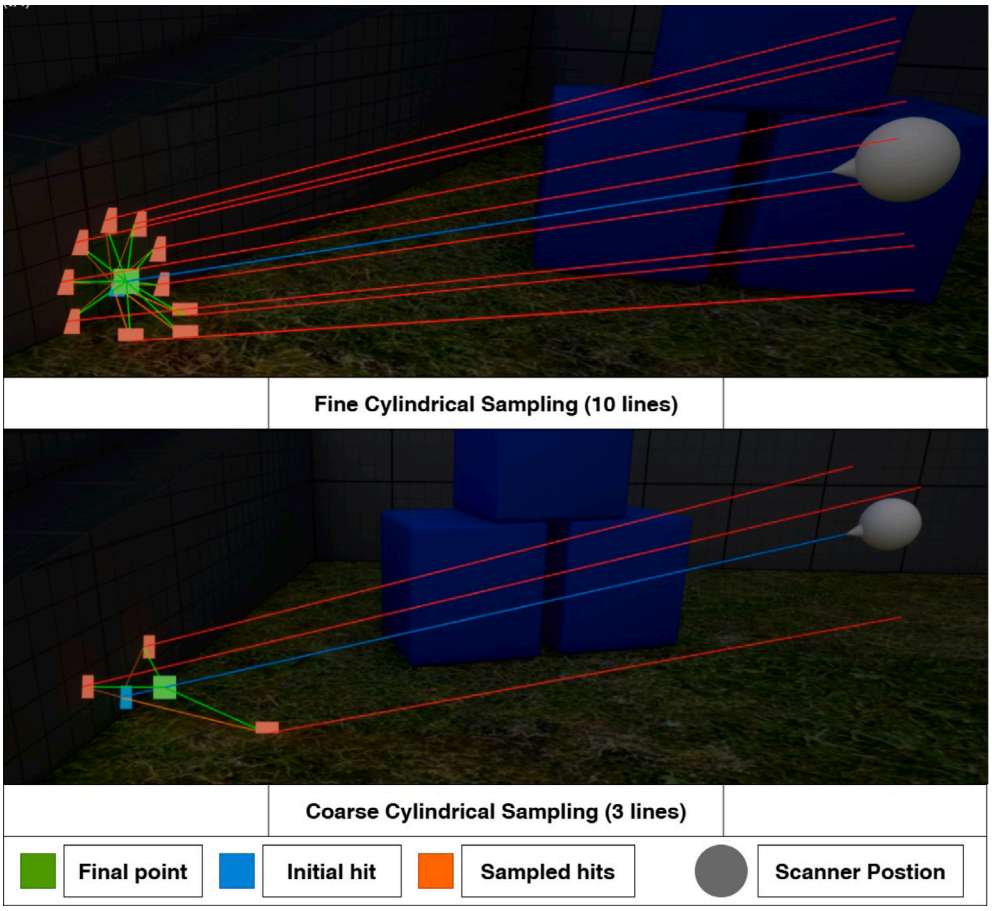

Fig. 8. Examples of different beam divergence configurations (exaggerated for visualization).

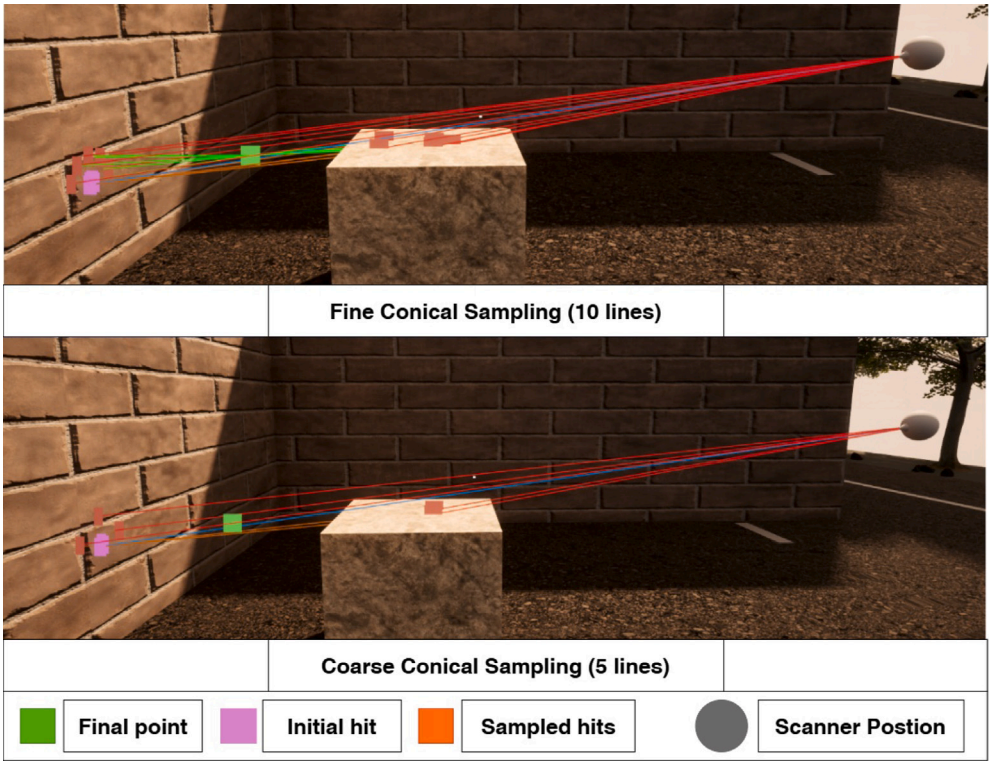

Fig. 9. Examples of different beam divergence configurations (exaggerated for visualization).

Table 1
Runtime of individual processing steps in the SynthRoads pipeline.

Process step	Domain	Time [s]
Parameter generation	Python	20
Model loading & cooking	Unreal Engine	186
Model baking	Unreal Engine	154
Scan preparation	Unreal Engine	38
Scanning	Unreal Engine	180
Scan merging	Python	48
Roughness adjustment	Python	100
Full process	Unreal Engine & Python	**726**

well as the *intersection over union (IoU)* (9).

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{7}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{8}$$

$$\text{IoU} = \frac{\text{TP}}{\text{TP} + \text{FP} + \text{FN}} \tag{9}$$

For experiment domain two, we use the mean absolute error rate (MAE) and Chamfer Distance (CD).

Let $\{(y_i, \hat{y}_i)\}_{i=1}^{N}$ be ground-truth/prediction pairs with $y_i, \hat{y}_i \in \mathbb{R}^2$ denoting 2D centerline points. We measure pointwise error by the Euclidean norm and report the mean absolute error.

$$\text{MAE} = \frac{1}{N} \sum_{i=1}^{N} \left\| y_i - \hat{y}_i \right\|_2, \tag{10}$$

where N is the number of evaluated points. Here, y_i denotes the ground-truth point and $\hat{y}_i$ the predicted point.

For curve-level evaluation, we fit a polynomial to the predicted and the ground-truth centerline, respectively, and sample both curves at uniform arc-length spacing of 1 m to obtain point sets $X = \{x_k\}_{k=1}^{|X|} \subset \mathbb{R}^2$ (prediction) and $Y = \{y_\ell\}_{\ell=1}^{|Y|} \subset \mathbb{R}^2$ (ground truth). The symmetric Chamfer distance is defined as:

$$\text{CD}(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\|_2 + \frac{1}{|Y|} \sum_{y \in Y} \min_{x \in X} \|y - x\|_2, \tag{11}$$

where $\| \cdot \|_2$ denotes the Euclidean norm.

We train all models five times to analyze model reliability. As a result, each metric is reported with its mean value and its standard deviation in brackets to allow insights into the stability of the respective tests.

5. Experimental evaluation

For the evaluation of our approach, we measured the computation time from initial parameter generation to the post-processed point cloud on a desktop computer with the following specifications:

- RAM: 64 GB DDR5
- Processor: Intel i9-11900K
- GPU: Nvidia Quadro RTX 5000
- Storage: Samsung NVMe PM9A1

The full process takes approximately 12 min from generation of the parameters to roughness-adjusted point cloud, as shown in Table 1. While the time of model cooking (loading the model and preparing procedural assets) and baking (finalizing the mesh representation) could potentially be accelerated through more efficient modeling strategies, the total time to receive a synthetic 600-meter-long road point cloud consisting of 15.46 million points with RGB color, intensity, metallic, roughness, specularity, object classes, and surface normals, as well as a corresponding geometry set, is acceptable for our use case. The resulting point cloud from this timed example is depicted in Fig. 10. While the point density of our scan can be easily adjusted, the data used for training is subsampled to 4 cm grid spacing anyway, which makes the tested configuration feasible.

To quantitatively assess the similarity between the synthetic and real data, we directly compare the real survey of the A45 with a corresponding synthetic point cloud generated by our framework. Both point clouds are grid-subsampled to a resolution of 4 cm to ensure comparable sampling density. For each dataset, we compute a set of geometric features (number of points, point density, and height statistics) and a local roughness descriptor, as well as radiometric features in the form of intensity and luminance statistics (Table 2). Intensities and RGB values are normalized to the native scales of the respective sensor or simulation, and roughness is derived as eigenvalue-based surface variation within a fixed neighborhood.

The synthetic point cloud exhibits a point density that is relatively close to the real scan, indicating that the simulated sampling pattern reproduces the overall coverage of the real survey. The lower mean height in the synthetic data is expected, as the synthetic scene focuses on the road space and its immediate surroundings, whereas the A45 survey includes additional surrounding topography. The mean roughness of the synthetic point cloud is higher than in the real scan, suggesting that the current roughness post-processing slightly overemphasizes small-scale surface variation. Radiometrically, the mean intensity in the synthetic data is significantly lower than in the real scan, whereas the mean luminance is close to the real values, indicating that the texture brightness in Unreal Engine is more consistent with the real appearance than the resulting LiDAR intensity.

This comparison is intentionally based on an uncalibrated synthetic point cloud from SynthRoads and illustrates that the geometric and radiometric feature distributions are a direct consequence of adjustable parameters in the framework. Roughness can be tuned by scaling the roughness post-processing strength, while intensity and luminance follow directly from the material parameters assigned in Unreal Engine. Because the global mean features of the synthetic point cloud are aggregates over all material classes, it is therefore sensible to calibrate the materials of each object type individually to more closely match the real data. A first step towards such calibration is shown in Fig. 11, where we introduce a software tool to qualitatively align the expected intensity distribution of a given material with its texture map or, alternatively, a fixed color and metallic/specularity combination at controllable incidence angles and distances. This type of statistical comparison provides a concrete basis for optimizing material parameters and roughness scaling in future iterations of the framework.

5.1. Value for Scan-to-Twin

The potential areas of application for SynthRoads are wide-ranging due to the versatility and number of adjustable parameters and degrees of freedom we allow in defining scenes, adjusting the scanner, and exposing output parameters, which can be used in multiple ways in post-processing.

In this paper, since the main goal of our workflow is to support the Scan-to-Twin approach, we focus our evaluation on two experimental domains. The first experiment domain is evaluating the performance gain of point cloud semantic segmentation when training on a mixed dataset of real and synthetic data compared to only real data. The real dataset we use for this experiment stems from our own surveys on closed German highway roads using a Riegl miniVUX 3 UAV scanner, which is depicted in Fig. 12.

Surveys were conducted on sections of the A1, A45, and A544, which are all part of the federal highway network of North Rhine-Westphalia. We additionally captured a section of A61 with a Riegl VZ600i terrestrial laser scanner for a more advanced analysis of generalization potential to different scan sources. The spatial extents of the conducted surveys range from 460 m to 2.1 km stretches of the respective highway environment. For comparison, we employ PointNeXt-B, a recent state-of-the-art point-based architecture that builds upon the

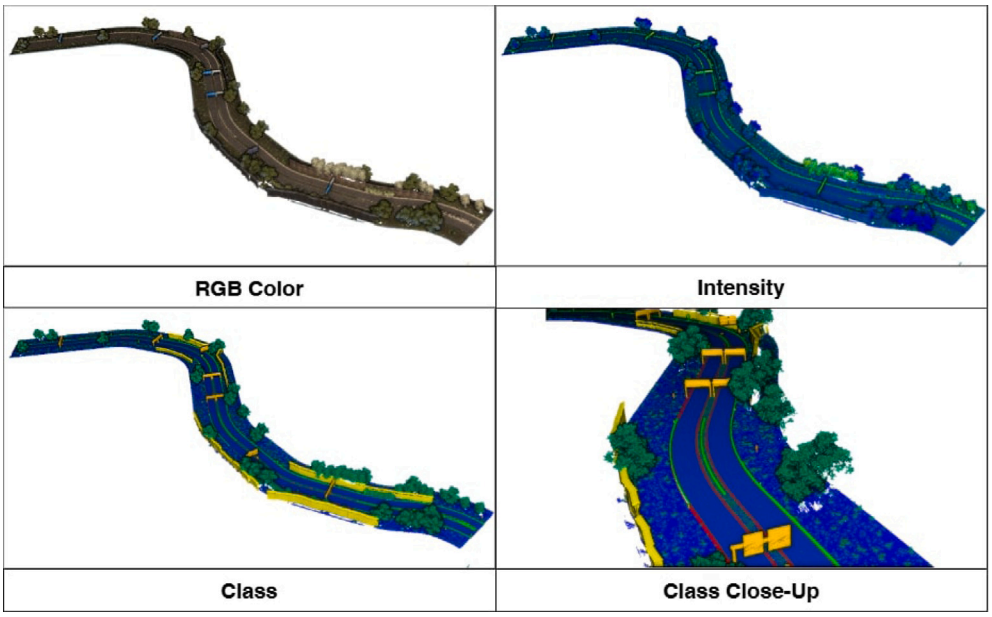

Fig. 10. Simulated scan of 600 m of automatically generated road model.

Table 2
Global comparison of geometric and radiometric statistics between real and synthetic point clouds using native intensity scales.

Dataset	Points [$\times 10^6$]	Density [pts/m^2]	$z_{mean} \pm z_{std}$ [m]	Roughness$_{mean}$	$I_{mean} \pm I_{std}$	$L_{mean} \pm L_{std}$
Real	56.41	134.0	24.57 ± 6.09	0.00197	0.607 ± 0.098	0.32 ± 0.21
Synthetic	13.71	146.3	3.02 ± 4.28	0.00320	0.154 ± 0.094	0.35 ± 0.14

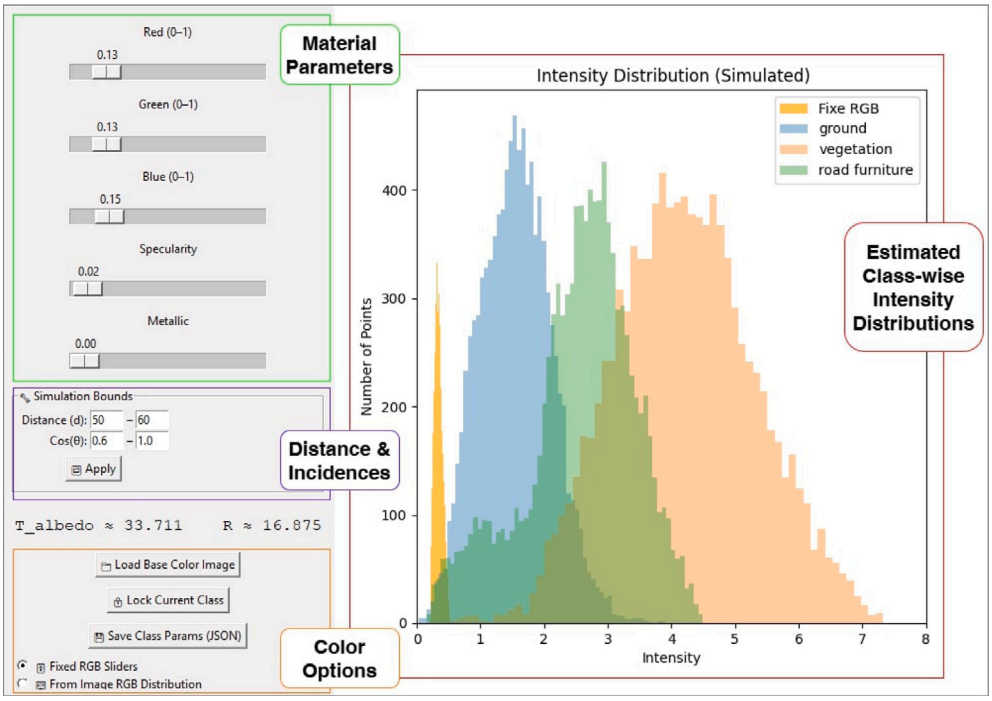

Fig. 11. Intensity calibration tool for Lambertian reflectance model.

residual design of PointNet++ while achieving higher efficiency and scalability. Its strong performance on large-scale benchmarks such as ScanNet and SemanticKITTI, combined with its robustness to varying point densities, makes it well suited for the evaluation of semantic segmentation in highway environments. We train three models on different dataset compositions. The first model is trained on a real dataset only, which we use as the baseline. The second model will be trained only on synthetic data as an indication of whether generalization towards real data is even possible with our synthetic point clouds. The third model is expected to perform best on a mixed dataset of real and synthetic data, as some studies have explored previously, such as [56]. The amount of real data is limited for model one and three to the same subsets of the

Fig. 12. Riegl miniVUX 3 UAV Scanner and drone used for surveys.

two surveys on A1 and A544, respectively, while A45 and A61 are used for testing. We use a dataset consisting of 20 synthetic point clouds of 600- (16) and 800-meter-long (4) road stretches, which we slice into subscenes with 60 m × 60 m dimensions for training. The features we use for training are the XYZ coordinates, color values, and intensity for all data sources. While A1 has no RGB values available due to the scanning setup at the time. We use small random noise to pad these features. We use grid subsampling with 4 cm resolution on all scenes. Furthermore, we train this first experiment on a five-class annotation setup with the classes non-road ground, road, road markings, vegetation, and road furniture. While the definition of the major parts of the scene is conducted automatically with our road generation module, we enhanced seven of the 20 synthetic road sections with additional assets to increase scene diversity using our asset placement module. To keep the generation time as low as possible, we used Trellis [57], an image-to-3D model generation model, to automatically generate additional assets such as shield gantries, surveillance poles, different types of guardrails, noise barrier objects, and road construction assets and placed them within our road scenes. As stated before, these objects can be simply imported into Unreal Engine as static mesh objects and tagged according to the desired class each object shall be assigned to. While this increases the diversity of the synthetic data used for our experiment, it also demonstrates the versatility and another key feature of our framework, being flexibility, as 3D models generated with Trellis are even textured, which can be easily fused into an unreal material. The road scene and also road assets can be generated with diverse approaches and can be used as long as they have complex collision boxes (meaning standard Unreal Engine foliage objects are not applicable). The fact that this is even true for fully automatically generated 3D meshes shall emphasize the versatility of our method. The training configuration was fixed as follows:

- voxel size: 0.1 m
- batch size: 16
- epochs: 50
- learning rate: 0.01
- gradient clip norm: 10
- weighted loss: True

The PointNeXt model default configuration for S3DIS benchmarking was used except for the initial radius, which we increased to 0.3 m.

This leads to the following two factors of interest in the first experiment:

1. Performance boost on same scan source data (A45)
2. Generalization towards different scan source data (A61)

The second experiment aims to evaluate the second synthetic data modality of geometry parameters. Specifically, we evaluate the potential of centerline regression from point clouds using deep learning. While both used models, which we introduced in Section 4.3, receive road slices that are extracted by cropping out blocks along the geospatial data of the road network, they differ in their approach with regard to input shapes. For comparison, we use a uniform slice length of 1 meter and fixed the input sequence length to 10 slices for the LSTM model. The comparison of different dataset compositions corresponds to the experiment for semantic segmentation. We manually extract the principal centerline of the main body of the highway sections from our real surveys as ground truth data and use the A1 and A544 point clouds for training the models with real data, while using 20 synthetic point clouds with corresponding automatically generated centerlines. We again test on A45 and A61. Since the slicing process requires approximate alignments, we use the publicly available geospatial data of the German ATKIS (Federal topographic cartographic information system) for coarsely aligning and slicing the road body while directly using the centerline of the synthetic data to replicate this process synthetically. The stretched length of the A45 segment is 850 m, and the stretched length of the A61 is 460 m. The input features per point are X, Y, Z, R, G, B, and intensity, similarly to the first experiment. Both models predict a single point per point cloud slice. We report the results using the mean absolute error (MAE) and the mean chamfer distance between third-degree polynomials fit to the ground truth and the predicted centerlines. To recall, the models developed for this comparison are meant to show the impact of our synthetic data on the task on different network approaches, not to identify the best-suited approach for centerline regression. Therefore, we do not claim the architectures developed for this experiment to be particularly well suited for the task, as developing a state-of-the-art centerline regression model is out of scope for this work.

Table 3
Model performance on A45 and A61 for the performance boosting use case (precision/recall in %).

Model	Dataset	Per-class precision/recall [%]				
		N.R.Gr.	Road	Road M.	Veg.	Road Fur.
UAV laser scanning data A45						
Mod. 1	Real	97 (1.3)/5 (2.0)	77 (5.3)/55 (31.7)	73 (20.2)/41 (31.3)	70 (13.5)/95 (2.3)	7 (5.9)/76 (16.6)
Mod. 2	Synth.	75 (5.8)/25 (8.7)	82 (6.2)/97 (2.5)	32 (12.4)/63 (25.3)	95 (2.0)/81 (9.6)	6 (2.8)/65 (26.0)
Mod. 3	Mixed	90 (3.5)/19 (10.4)	76 (1.8)/100 (0.7)	80 (18.5)/10 (8.4)	87 (1.3)/95 (0.8)	14 (2.7)/86 (1.7)
TLS data A61						
Mod. 1	Real	90 (5.2)/2 (2.0)	71 (1.3)/100 (0.0)	99 (0.4)/61 (2.5)	88 (3.2)/92 (4.2)	44 (11.7)/78 (5.3)
Mod. 2	Synth.	80 (9.0)/23 (6.8)	74 (0.9)/100 (0.1)	83 (9.2)/57 (15.4)	92 (1.7)/93 (2.5)	50 (7.9)/80 (8.5)
Mod. 3	Mixed	88 (3.5)/9 (3.7)	73 (0.4)/100 (0.2)	98 (0.6)/63 (7.4)	88 (0.9)/95 (1.9)	52 (3.9)/83 (3.4)

Table 4
Model performance on A45 and A61 for the performance boosting use case (intersection over union, IoU, in %).

Model	Dataset	Per-class IoU [%] (mean (std))				
		N.R.Gr.	Road	Road M.	Veg.	Road Fur.
UAV laser scanning data A45						
Mod. 1	Real	5.40 (2.06)	46.93 (25.66)	26.85 (16.59)	67.18 (11.79)	6.66 (5.18)
Mod. 2	Synth.	22.78 (6.93)	79.58 (3.96)	24.88 (9.54)	77.53 (7.92)	6.22 (2.78)
Mod. 3	Mixed	18.94 (10.10)	75.45 (1.54)	10.33 (8.29)	82.95 (0.84)	12.27 (2.58)
TLS data A61						
Mod. 1	Real	1.92 (2.02)	70.97 (1.26)	60.28 (2.47)	81.27 (1.01)	38.45 (9.44)
Mod. 2	Synth.	21.39 (5.76)	73.63 (0.87)	50.09 (12.23)	86.21 (1.07)	43.46 (4.96)
Mod. 3	Mixed	8.50 (3.60)	73.26 (0.41)	62.16 (7.18)	84.24 (1.96)	46.77 (3.25)

6. Results

In this section, we present the results of our experiments in the domains of point cloud semantic segmentation and geometry extraction detailed in Section 5. Tables 3 and 4 depict the results of experiment one on semantic segmentation. Table 5 depicts the result of the second experiment in the semantic segmentation domain, and Table 6 depicts the results of the experiment on centerline regression. As stated in Section 4.4, we repeated training and inference for all models five times and report the mean of the performance metrics and the standard deviations for each metric in brackets in the following evaluation tables to analyze model stability.

6.1. Synthetic data for semantic segmentation

The results indicate that the synthetic data is capable of providing enough context and characteristic patterns to the PointNeXt model that it is capable of surpassing the small real dataset's performance by only ever seeing synthetic data. When predicting TLS data, Model 2, trained on the synthetic dataset, shows good generalization and superior stability, especially compared to Model 1, which has the worst performance of the three models. The reason for this is likely the number of diverse scenes provided to Model 2 by the synthetic data. Model 3 shows a general reduction in model instability through generally low standard deviations compared to the other models except for the non-road ground class. It also shows the best performance for vegetation and road furniture objects, as their shapes supposedly match best between real and synthetic data. Comparing the performance of road marking segmentation over all three models shows that mixing the data leads to a decreased capability to extract this class. The reason for this is likely as follows: While the qualitative intensity distributions between classes are relatively well reflected by the synthetic data, it has a certain bias, potentially leading to a domain shift, confusing the model when mixing the different data. This is less of an issue when the target data source is different from both data sources used for training, as can be seen in the performance of Model 3 on the terrestrial data, where the road markings are relatively unaffected by the data source. The model comparisons on the terrestrial data show that Model 3 outperforms Model 1 over all classes with regard to IoU, while only slightly adding instability, as can be seen in the road markings class.

The results from the first experiment led to additional considerations of the potential use of the synthetic data in a scenario with a larger diversity of classes. Since the classes of the synthetic scans can be easily extended by adding new tags for specific assets, we extended the dataset to a total of 17 classes, defined as follows: Ground, road, road markings, vegetation, guardrail, barrier, road sign, poles, traffic light, construction barriers, noise barrier, shield gantry, utility, vehicle, warning post, roadside, and bridge. Especially in road environments, a typical issue is the strong class imbalance in real datasets: relevant assets may only appear a few times, even in surveys covering several kilometers. We therefore extended the evaluation to test the potential of targeted class mix-in through our synthetic data. For this test, we chose the shield gantry class, since it is present in multiple real surveys but difficult to extract when training only on the real dataset. We generated six basic road sections and added shield gantries in different configurations, e.g., one-sided gantries, double-sided gantries, and shaded gantries, and trained four models on the extended-class setup. Model 4 is trained on **all** available real data from A544 and A45 only. For Models 5, 6, and 7, we add the point clouds of the six new synthetic scenes to the training set. Models 5 and 6 use a single training phase on the mixed dataset, whereas Model 7 follows a two-stage pretraining–finetuning scheme. It is first trained on the synthetic scenes only and then fine-tuned on the real data for 30 epochs. This allows us to explicitly evaluate the model-level domain adaptation potential of synthetic pretraining. We swap the evaluation set from the A45 to the A1 highway section, since it is a survey with multiple shield gantries, which makes our evaluation more robust. And test the effect of intensity quantile mapping (qm) against the same setup with no quantile mapping (nqm) on the model performance.

The results are summarized in Table 5. Since the validation dataset only comprises nine of the 17 classes, we only report IoUs and standard deviations for these classes. To analyze the reduction of the domain shift, we introduce adaptation both at the data and at the model level. On the data side, Model 5 applies intensity quantile mapping between the A544 section and the synthetic scenes to make the synthetic intensities more similar to the real ones. This preprocessing targets the pronounced domain gap in intensity statistics between synthetic and real point clouds, which is also commonly observed between different real laser scanning systems. Although quantile mapping may reduce

Table 5
Model performance on A1 (class mix-in use case), intersection over union (IoU).

Model	Data	Per-class IoU [%] (mean (std))				
		N.R.Gr.	Road	Road M.	Veg.	Guardr.
Mod. 4	Real	55.16 (1.67)	83.21 (1.90)	83.87 (2.95)	84.34 (3.16)	72.42 (5.92)
Mod. 5	Mixed (qm)	51.81 (3.52)	84.96 (0.76)	**87.84** (0.47)	80.36 (1.24)	59.19 (6.81)
Mod. 6	Mixed (nqm)	54.52 (5.24)	78.39 (2.01)	0.00 (0.00)	83.27 (1.39)	80.97 (3.40)
Mod. 7	Mixed (qm + Pre/Fine)	53.27 (3.52)	86.08 (0.55)	86.98 (1.56)	82.63 (2.88)	73.42 (5.31)
Val. sup.	Num. pts	3.93 M	2.77 M	125 k	5.99 M	109 k
Model	Data	Additional classes (IoU [%] (mean (std)))				
		Warn. P.	Noise B.	Sh. Gant.	Veh.	
Mod. 4	Real	20.78 (5.19)	12.32 (5.94)	0.00 (0.00)	78.75 (17.87)	
Mod. 5	Mixed (qm)	26.94 (3.75)	8.82 (6.46)	36.32 (16.29)	94.29 (2.48)	
Mod. 6	Mixed (nqm)	21.78 (5.45)	20.95 (14.14)	**45.42** (7.92)	63.23 (10.64)	
Mod. 7	Mixed (qm + Pre/Fine)	18.86 (4.99)	7.71 (4.52)	0.00 (0.00)	85.26 (13.91)	
Val. sup.	Num. pts	2.06 k	151 k	22.7 k	12.8 k	

Table 6
Centerline regression performance on A45 and A61 (mean (std) in meters).

Model	Dataset	Metrics [m] (mean (std))	
		MAE	Chamfer dist. (fit)
UAV-laser scanning data A45			
Single section model (1)	Real	0.3293 (0.0003)	0.1529 (0.0077)
Sequence model (4)	Real	0.2572 (0.0406)	0.1497 (0.0553)
Single section model (2)	Synthetic	0.3201 (0.0132)	0.1400 (0.0121)
Sequence model (5)	Synthetic	0.2559 (0.0322)	**0.1065** (0.0309)
Single section model (3)	Mixed	0.3202 (0.0052)	0.1516 (0.0051)
Sequence model (6)	Mixed	**0.2112** (0.0219)	0.1463 (0.0264)
TLS data A61			
Single section model (1)	Real	0.4903 (0.0216)	1.0288 (0.0363)
Sequence model (4)	Real	0.3423 (0.0496)	0.7796 (0.2038)
Single section model (2)	Synthetic	0.4085 (0.0268)	1.0090 (0.0664)
Sequence model (5)	Synthetic	0.3287 (0.0567)	0.7588 (0.3975)
Single section model (3)	Mixed	0.4109 (0.0187)	1.0198 (0.0873)
Sequence model (6)	Mixed	**0.3199** (0.0883)	**0.6563** (0.2416)

some information contained in the synthetic intensity values, it substantially improves performance for road markings, vehicles, warning posts, and the road surface, while slightly degrading results for ground, vegetation, guardrails, noise barriers, and the shield gantry class compared to the real-only baseline.

We report per-class IoUs for the real-data baseline (Model 4), the two single-stage mixed models with (Model 5, "qm") and without (Model 6, "nqm") quantile mapping, and the two-stage pretraining–finetuning model (Model 7), which first trains on synthetic data and is then finetuned on real data. The mixed models clearly demonstrate the potential for targeted performance boosting with our synthetic data: both mixed variants lead to high performance on the shield gantry class, despite it being a minority class with less than 0.4% of the points of the majority class vegetation in the validation set. While the model trained on real data is not capable of detecting shield gantries at all, we receive relatively high IoUs in Model 5 and 6. In addition, the other minority classes warning post and vehicle receive a substantial performance boost when quantile mapping is applied.

Comparing the single-stage mixed setups (Models 5 and 6) with the pretraining–finetuning scheme (Model 7) provides an explicit model-level domain adaptation baseline. Model 7 acts as a more conservative adaptation towards the real-data distribution: it slightly improves dominant classes such as road and guardrail over the real-only baseline and remains close to the best mixed model for road markings, but it fails to preserve the synthetic-data gains for very rare classes such as shield gantries and warning posts, where IoUs drop back to near-zero levels. This behavior indicates that naive fine-tuning on highly imbalanced real data can lead to partial forgetting of structures that are mainly introduced by the synthetic data, whereas single-stage mixed training with synthetic data is more effective when the goal is to boost specific minority asset classes. Examples of the targeted asset models within Unreal Engine and the corresponding synthetic point cloud data are provided in the supplementary material (Pages 1–7, Figs. 2–8).

6.2. Synthetic data for centerline regression

The results of the centerline regression in Table 6 demonstrate a significant performance enhancement of the sequence model on the mixed dataset over the real data with a reduction in MAE on the A45 by almost 18% and 6.5% on the terrestrial data of A61. Interestingly, the models trained on the synthetic data still match the models trained on real data, even though the underlying roads had one standard layout, which did not include any lane transitions or turnouts, as is the case for A1 and A45. While the single-section model generally lacked the ability to surpass the performance boundary of 32 cm MAE on A45, the sequence model consistently outperformed the single-section model with any of the dataset compositions. Interestingly, the sequence model gained stability with more data on A45, which is indicated by the declining standard deviations of both metrics from the real to the synthetic and from the synthetic to the mixed dataset. The closest matches of fitted polynomials are generated on prediction of models trained on the synthetic data, which may be due to both models being influenced by the regularity of the synthetic data and reducing strong outliers in their predictions. The performance on A61 is overall weaker than on A45, reflecting a domain gap between UAV laser scanning and TLS. Generally, TLS exhibits steeper incidence angles and sparser coverage near scene boundaries, which increases extrapolation and fitting errors. However, the training of the sequence model on the mixed dataset closes part of this gap and delivers the best overall results on A61.

This experiment clearly demonstrates that we are able to improve the performance of centerline regression models with our synthetic data.

To contextualize the proposed end-to-end centerline regression, we evaluate it against a conventional extraction baseline that reflects common practice in road-centerline pipelines. Following the idea of skeletonization-based methods such as [58], we use the trained PointNeXt model (Mod. 3) from the previous experiment to classify road points on A45, rasterize the resulting road mask, and apply binary skeletonization to extract a candidate centerline.

To contextualize the proposed end-to-end centerline regression, we evaluate it against a conventional extraction baseline that reflects common practice in road-centerline pipelines, i.e., semantic road segmentation followed by rasterization and skeletonization [58]. Concretely, we (i) classify road points using the PointNeXt model from Experiment 1 (Mod. 3), (ii) rasterize the predicted road mask, (iii) apply binary skeletonization, (iv) prune short branches, and (v) derive a corridor centerline by projecting skeleton vertices onto the principal corridor axis and averaging their lateral position per longitudinal bin. We report the same metrics as for learning-based regression (MAE and curve-level Chamfer distance).

The baseline has two key hyperparameters, the raster resolution r and the minimum branch length L_{min}. To provide a best-case reference, we select r and L_{min} via grid search to minimize MAE to the ground-truth centerline on A45, resulting in $r = 1$ m and $L_{min} = 15$ m. We then keep these parameters fixed when transferring the baseline to A61.

With this configuration, the skeletonization baseline achieves an MAE of 1.44 m and a Chamfer distance of 1.43 m on A45, whereas our regression model (Sequence Model (6)) achieves 0.211 m MAE and 0.1463 m Chamfer distance (Table 6). On A61, the baseline achieves 1.47 m MAE but degrades strongly in Chamfer distance (6.27 m) due to missing skeleton segments caused by sparse corridor edges. This highlights a known sensitivity of rasterization-based pipelines to incomplete coverage. To isolate the effect of edge sparsity, we additionally evaluate the baseline after manually excluding sparse boundary regions. In this controlled setting, the baseline improves to 0.714 m MAE and 322 m Chamfer Distance yet underperforms the centerline regression model with MAE of 0.380 m and Chamfer Distance of 0.227 m.

Overall, this benchmark indicates that direct centerline regression is competitive with skeletonization-based pipelines as commonly used in recent work while being more flexible and less sensitive to sparsity at the corridor edges. Even when performance degrades under severe sparsity, the regression approach maintains relatively stable accuracy. Visual examples of the lower edge of A45 and the cropped A61 corridor, showing ground truth, skeletonization-based, and learned centerlines, are presented in Fig. 13.

7. Conclusion

Our SynthRoads framework allows the automatic generation of realistic synthetic point clouds of highway scenes. The conducted experiments demonstrate that leveraging the synthetic data improves centerline regression and point cloud semantic segmentation tasks, which are essential for Scan-to-Twin (RQ1). The modular structure of model definition, model generation, and scanning is fast, highly versatile, and easily customizable. The features of our scan simulation allow the generation of realistic synthetic point clouds with a high degree of diversity. As shown in Section 5, the generation of a 600 m long highway segment with a 3D model, geometry parameters, and a synthetic point cloud takes approximately 12 min, which answers research question RQ2. Lastly, we showed that the synthetic data can be used for multiple tasks of the Scan-to-Twin process with our experiments, which answers research question RQ3.

Besides the advantages of our framework, there are also challenges and limitations, especially with regard to domain shifts in the derived point features such as intensity and material-related parameters. Another challenge, which already became apparent in the experiments, concerns the domain gap between synthetic and real data in several scan-related aspects. Even after material calibration, the intensity values differ in their distribution for comparable scenes. This issue, however, also exists between different real scanners and can be partially overcome with various methods of domain adaptation. We demonstrated that intensity quantile mapping is an effective and straightforward way of partially reducing the intensity domain shift, leading to better generalization of semantic segmentation of road markings, while slightly reducing performance on a few other classes.

A further limitation concerns the direct applicability of our scan simulation to other infrastructure domains such as tunnels or bridges, since the current scanning platform and the generation of the standard scanning trajectory are focused on UAV-based scanning. A future research perspective will be to implement additional platforms for scan simulation to also allow direct use for other domains where UAV-based platforms are infeasible. This is also true for the model generation part: as we currently focus on roads with relatively simple scenarios, more complex models may further increase the value of our synthetic data generation framework. While Unreal Engine allows the integration of dynamic objects, and we demonstrated a possible approach to incorporate moving vehicles on the road, this currently requires additional effort and is not yet implemented in the automatic process, which will, however, be pursued in future work.

Lastly, a challenge is closing the loop between scans and models by providing a schema or interface that can define both model configurations and scan-related extractions. Such an interface would enable the automatic generation of diverse GSMs for different Digital Twin use cases from one set of extractions resulting from the Scan-to-Twin process. In this work, we showed the inverse process, where we first defined a model that is then synthetically converted into a scan.

The simplified Lambertian reflectance model tends to yield unrealistic results for small incidence angles, leading to comparatively low intensities and neglecting microfacet backscattering and other effects that are better captured by the Oren-Nayar model. We therefore allow configuration of both reflectance models in our framework to simplify the adaptation of our scan simulation to other use cases. As our main use case, however, is UAV laser scanning of road surfaces with mostly near-orthogonal incidences, we used the Lambertian model for our analyses, for which its reflectance behavior is sufficient.

Since we already showed the value for the Scan-to-Twin enhancement use case, the potential of the presented framework goes far beyond the demonstrated applications, as the modularity allows the generation of synthetic point clouds of diverse scenes and could therefore be used for different domains by exchanging specific components such as the modeling module and extending the scan simulation with new platform configurations. The workflow can also be used to provide data for other tasks, such as high-resolution change detection or material estimation. To allow fellow scholars to leverage our work for new research, we will provide the virtual laser scanner module as an open-source Unreal Engine plugin upon publication of this paper, together with the synthetic dataset used in this work. It comprises a total of 26 highway sections, with 20 annotated using the five-class schema from Experiment 1 and six using the 17-class schema from Experiment 2. Furthermore, the point cloud data are accompanied by the centerlines of each point cloud upon request.

In this work, we deliberately focused on extreme cases (real-only, synthetic-only, and a fixed mixed setting) to evaluate the fundamental potential of our synthetic data, while future research on more differentiated fusion ratios of real and synthetic data and on the impact of annotation consistency could provide further insight into the optimal data composition for performance enhancements. Another important future research topic building on this framework will be end-to-end point-cloud-to-model prediction, where we will extend the approach used for centerline regression to more parameters, with the goal of directly predicting the full road corridor of a real scan. A further perspective is using the synthetic data to calibrate single-task models throughout the Scan-to-Twin process to allow better estimates of model uncertainty for both semantic and geometric tasks.

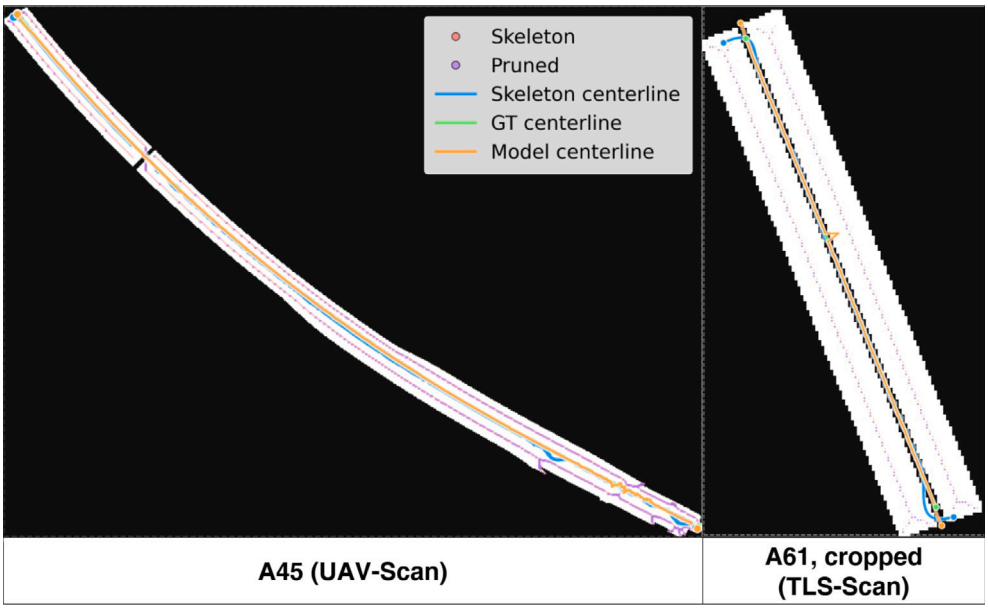

Fig. 13. Comparison of learned centerline prediction and a skeletonization-based rasterization approach.

CRediT authorship contribution statement

David Crampen: Writing – review & editing, Writing – original draft, Visualization, Validation, Methodology, Investigation, Formal analysis, Data curation, Conceptualization. **Marcel Hein:** Software, Investigation. **Jörg Blankenbach:** Writing – review & editing, Supervision, Project administration, Funding acquisition.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded by the German Research Foundation (DFG), as part of the Collaborative Research Center 339, Germany (SFB/TRR 339) (project ID: 453596084). The financial support from the DFG, Germany is gratefully acknowledged. We also acknowledge SideFX Software Inc. for providing us with their software products.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.aei.2026.104410.

Data availability

Data will be made available on request.

References

[1] D. Crampen, M. Hein, J. Blankenbach, A level of as-is detail concept for digital twins of roads—A case study, in: T.H. Kolbe, A. Donaubauer, C. Beil (Eds.), Recent Advances in 3D Geoinformation Science, Springer Nature Switzerland, Cham, 2024, pp. 499–515, http://dx.doi.org/10.3390/rs13142663.

[2] Y. Jiang, M. Li, W. Wu, X. Wu, X. Zhang, X. Huang, R.Y. Zhong, G.G. Huang, Multi-domain ubiquitous digital twin model for information management of complex infrastructure systems, Adv. Eng. Inform. 56 (2023) 101951, http://dx.doi.org/10.1016/j.aei.2023.101951.

[3] M. Mehranfar, M.A. Vega-Torres, A. Braun, A. Borrmann, Automated data-driven method for creating digital building models from dense point clouds and images through semantic segmentation and parametric model fitting, Adv. Eng. Inform. 62 (2024) 102643, http://dx.doi.org/10.1016/j.aei.2024.102643.

[4] S. Chen, G. Fan, J. Li, Improving completeness and accuracy of 3D point clouds by using deep learning for applications of digital twins to civil structures, Adv. Eng. Inform. 58 (2023) 102196, http://dx.doi.org/10.1016/j.aei.2023.102196.

[5] A. Justo, M. Soilán, A. Sánchez-Rodríguez, B. Riveiro, Scan-to-BIM for the infrastructure domain: Generation of IFC-compliant models of road infrastructure assets and semantics using 3D point cloud data, Autom. Constr. 127 (2021) 103703, http://dx.doi.org/10.1016/j.autcon.2021.103703.

[6] X. Mi, Z. Dong, Z. Cao, B. Yang, Z. Cao, C. Zheng, J. Stoter, L. Nan, A benchmark approach and dataset for large-scale lane mapping from MLS point clouds, Int. J. Appl. Earth Obs. Geoinf. 133 (2024) 104139, http://dx.doi.org/10.1016/j.jag.2024.104139.

[7] Z. Chen, L. Deng, Y. Luo, D. Li, J.M. Junior, W.N. Gonçalves, A.A.M. Nurunnabi, J. Li, C. Wang, D. Li, Road extraction in remote sensing data: A survey, Int. J. Appl. Earth Obs. Geoinf. 112 (2022) 102833, http://dx.doi.org/10.1016/j.jag.2022.102833.

[8] M. Soilán, H. Tardy, D. González-Aguilera, Deep learning-based road segmentation of 3D point clouds for assisting road alignment parameterization, Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci. 43 (2022) 283–290, http://dx.doi.org/10.5194/isprs-archives-XLIII-B2-2022-283-2022.

[9] Y. Wang, W. Wang, J. Liu, T. Chen, S. Wang, B. Yu, X. Qin, Framework for geometric information extraction and digital modeling from LiDAR data of road scenarios, Remote. Sens. 15 (3) (2023) 576, http://dx.doi.org/10.3390/rs15030576.

[10] A.H. Safaie, H. Rastiveis, A. Shams, W.A. Sarasua, J. Li, Automated street tree inventory using mobile LiDAR point clouds based on hough transform and active contours, ISPRS J. Photogramm. Remote Sens. 174 (2021) 19–34, http://dx.doi.org/10.1016/j.isprsjprs.2021.01.026.

[11] F. Li, G. Vosselman, Pole-like road furniture detection and decomposition in mobile laser scanning data based on spatial relations, Remote. Sens. 10 (4) (2018) 531, http://dx.doi.org/10.3390/rs10040531.

[12] J. Zhang, X. Zhao, Z. Chen, Z. Lu, A review of deep learning-based semantic segmentation for point cloud, IEEE Access 7 (2019) 179118–179133, http://dx.doi.org/10.1109/ACCESS.2019.2958671.

[13] R. Cazorla, L. Poinel, P. Papadakis, C. Buche, Reducing domain shift in synthetic data augmentation for semantic segmentation of 3d point clouds, in: 2022 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2022, pp. 1198–1205, http://dx.doi.org/10.1109/SMC53654.2022.9945480.

[14] Forschungsgesellschaft für Straßen- und Verkehrswesen (FGSV), Richtlinien für die Anlage von Autobahnen (RAA), FGSV Verlag, Cologne, Germany, 2008, Guidelines for the Design of Motorways.

[15] H. Vassilev, M. Laska, J. Blankenbach, Uncertainty-aware point cloud segmentation for infrastructure projects using Bayesian deep learning, Autom. Constr. 164 (2024) 105419, http://dx.doi.org/10.1016/j.autcon.2024.105419.

[16] Epic Games, Unreal engine, 2025, https://www.unrealengine.com/, Version 5.3. (Accessed 21 May 2025).

[17] G. Qian, Y. Li, H. Peng, J. Mai, H. Hammoud, M. Elhoseiny, B. Ghanem, Pointnext: Revisiting pointnet++ with improved training and scaling strategies, Adv. Neural Inf. Process. Syst. 35 (2022) 23192–23204, http://dx.doi.org/10.48550/arXiv.2206.04670.

[18] Z. Song, Z. He, X. Li, Q. Ma, R. Ming, Z. Mao, H. Pei, L. Peng, J. Hu, D. Yao, et al., Synthetic datasets for autonomous driving: A survey, IEEE Trans. Intell. Veh. 9 (1) (2023) 1847–1864, http://dx.doi.org/10.1109/TIV.2023.3331024.

[19] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, CARLA: An open urban driving simulator, in: Conference on Robot Learning, PMLR, 2017, pp. 1–16.

[20] D. Dworak, F. Ciepiela, J. Derbisz, I. Izzat, M. Komorkiewicz, M. Wójcik, Performance of LiDAR object detection deep learning architectures based on artificially generated point cloud data from CARLA simulator, in: 2019 24th International Conference on Methods and Models in Automation and Robotics, MMAR, IEEE, 2019, pp. 600–605, http://dx.doi.org/10.1109/MMAR.2019.8864642.

[21] Y. Yuan, M. Sester, Comap: A synthetic dataset for collective multi-agent perception of autonomous driving, Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci. 43 (2021) 255–263, http://dx.doi.org/10.5194/isprs-archives-XLIII-B2-2021-255-2021.

[22] J.-E. Deschaud, D. Duque, J.P. Richa, S. Velasco-Forero, B. Marcotegui, F. Goulette, Paris-CARLA-3D: A real and synthetic outdoor point cloud dataset for challenging tasks in 3D mapping, Remote. Sens. 13 (22) (2021) 4713, http://dx.doi.org/10.3390/rs13224713.

[23] C. Li, Y. Ren, B. Liu, Pcgen: Point cloud generator for lidar simulation, in: 2023 IEEE International Conference on Robotics and Automation, ICRA, IEEE, 2023, pp. 11676–11682.

[24] A. Xiao, J. Huang, D. Guan, F. Zhan, S. Lu, Transfer learning from synthetic to real lidar point cloud for semantic segmentation, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 36, 2022, pp. 2795–2803, http://dx.doi.org/10.1609/aaai.v36i3.20183.

[25] L. Winiwarter, A.M.E. Pena, H. Weiser, K. Anders, J.M. Sánchez, M. Searle, B. Höfle, Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic full-waveform 3D laser scanning, Remote Sens. Environ. 269 (2022) 112772, http://dx.doi.org/10.1016/j.rse.2021.112772.

[26] T. Luhmann, M. Chizhova, D. Gorkovchuk, D. Popovas, J. Gorkovchuk, M. Hess, Development of terrestrial laser scanning simulator, Int. Arch. Photogramm. Remote. Sens. Spat. Inf. Sci. 46 (2022) 329–334, http://dx.doi.org/10.5194/isprs-archives-XLVI-2-W1-2022-329-2022.

[27] R. Li, X. Li, K.-H. Hui, C.-W. Fu, SP-GAN: Sphere-guided 3D shape generation and manipulation, ACM Trans. Graph. 40 (4) (2021) 1–12, http://dx.doi.org/10.1145/3450626.3459766.

[28] A.X. Chang, T. Funkhouser, L. Guibas, P. Hanrahan, Q. Huang, Z. Li, S. Savarese, M. Savva, S. Song, H. Su, et al., Shapenet: An information-rich 3d model repository, 2015, arXiv preprint arXiv:1512.03012.

[29] S. Luo, W. Hu, Diffusion probabilistic models for 3d point cloud generation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2021, pp. 2837–2845.

[30] Z. Wu, S. Song, A. Khosla, F. Yu, L. Zhang, X. Tang, J. Xiao, 3D shapenets: A deep representation for volumetric shapes, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1912–1920.

[31] Z. Xiang, Z. Huang, K. Khoshelham, Synthetic lidar point cloud generation using deep generative models for improved driving scene object recognition, Image Vis. Comput. 150 (2024) 105207, http://dx.doi.org/10.1016/j.imavis.2024.105207.

[32] H. Caesar, V. Bankiti, A.H. Lang, S. Vora, V.E. Liong, Q. Xu, A. Krishnan, Y. Pan, G. Baldan, O. Beijbom, Nuscenes: A multimodal dataset for autonomous driving, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11621–11631.

[33] C.R. Qi, H. Su, K. Mo, L.J. Guibas, Pointnet: Deep learning on point sets for 3d classification and segmentation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 652–660, http://dx.doi.org/10.1109/CVPR.2017.16.

[34] P. Achlioptas, O. Diamanti, I. Mitliagkas, L. Guibas, Learning representations and generative models for 3d point clouds, in: International Conference on Machine Learning, PMLR, 2018, pp. 40–49.

[35] R. Li, X. Li, C.-W. Fu, D. Cohen-Or, P.-A. Heng, Pu-gan: a point cloud upsampling adversarial network, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 7203–7212.

[36] G. Uggla, M. Horemuz, Towards synthesized training data for semantic segmentation of mobile laser scanning point clouds: Generating level crossings from real and synthetic point cloud samples, Autom. Constr. 130 (2021) 103839, http://dx.doi.org/10.1016/j.autcon.2021.103839.

[37] E. Frías, J. Pinto, R. Sousa, H. Lorenzo, L. Díaz-Vilariño, Exploiting BIM objects for synthetic data generation toward indoor point cloud classification using deep learning, J. Comput. Civ. Eng. 36 (6) (2022) 04022032, http://dx.doi.org/10.1061/(ASCE)CP.1943-5487.0001039.

[38] J.W. Ma, B. Han, F. Leite, An automated framework for generating synthetic point clouds from as-built BIM with semantic annotation for scan-to-BIM, in: 2021 Winter Simulation Conference, WSC, IEEE, 2021, pp. 1–10, http://dx.doi.org/10.1109/WSC52266.2021.9715301.

[39] L. Yang, Y.-C. Lin, H. Cai, A. Habib, From scans to parametric BIM: an enhanced framework using synthetic data augmentation and parametric modeling for highway bridges, J. Comput. Civ. Eng. 38 (3) (2024) 04024008, http://dx.doi.org/10.1061/JCCEE5.CPENG-5640.

[40] Q. Zhu, L. Fan, N. Weng, Advancements in point cloud data augmentation for deep learning: A survey, Pattern Recognit. (2024) 110532, http://dx.doi.org/10.1016/j.patcog.2024.110532.

[41] J.R. Vargas Rivero, T. Gerbich, B. Buschardt, J. Chen, Data augmentation of automotive lidar point clouds under adverse weather situations, Sensors 21 (13) (2021) 4503, http://dx.doi.org/10.3390/s21134503.

[42] S. Qiu, J. Chen, C. Lai, H. Lu, X. Xue, J. Pu, Leveraging smooth deformation augmentation for lidar point cloud semantic segmentation, IEEE Trans. Intell. Veh. 9 (2) (2024) 3316–3329, http://dx.doi.org/10.1109/TIV.2023.3348805.

[43] B. Fei, W. Yang, W.-M. Chen, Z. Li, Y. Li, T. Ma, X. Hu, L. Ma, Comprehensive review of deep learning-based 3d point cloud completion processing and analysis, IEEE Trans. Intell. Transp. Syst. 23 (12) (2022) 22862–22883, http://dx.doi.org/10.1109/TITS.2022.3195555.

[44] J. Dong, N. Wang, H. Fang, H. Lu, D. Ma, H. Hu, Automatic augmentation and segmentation system for three-dimensional point cloud of pavement potholes by fusion convolution and transformer, Adv. Eng. Inform. 60 (2024) 102378, http://dx.doi.org/10.1016/j.aei.2024.102378.

[45] D. Crampen, M. Hein, J. Blankenbach, RoadGen4Twins: A modular approach for generating multi-purpose geometric-semantic models for digital twins of roads, ISPRS Ann. Photogramm. Remote. Sens. Spat. Inf. Sci. X-4/W5-2024 (2024) 103–110, http://dx.doi.org/10.5194/isprs-annals-X-4-W5-2024-103-2024, URL https://isprs-annals.copernicus.org/articles/X-4-W5-2024/103/2024/.

[46] D. Crampen, J. Blankenbach, LOADt: towards a concept of level of as-is detail for digital twins of roads, in: Proceedings of the 30th International Workshop on Intelligent Computing in Engineering (EG-ICE), London, UK, 2023, pp. 4–7.

[47] Esri Inc., Esri CityEngine, Esri, Redlands, CA, 2024, Version 2024.0, https://www.esri.com/en-us/cityengine.

[48] MathWorks, Inc., RoadRunner, MathWorks, Natick, MA, 2024, RoadRunner Scene Builder, Version 2024a. Available at: https://www.mathworks.com/products/roadrunner.html.

[49] H. Ledoux, K. Arroyo Ohori, K. Kumar, B. Dukai, A. Labetski, S. Vitalis, CityJSON: A compact and easy-to-use encoding of the CityGML data model, Open Geospatial Data Softw. Stand. 4 (1) (2019) 1–12, http://dx.doi.org/10.1186/s40965-019-0064-0.

[50] RIEGL MiniVUX-3UAV: LiDAR Sensor for Unmanned Laser Scanning, RIEGL Laser Measurement Systems GmbH, 2025, Datasheet (PDF). URL https://www.riegl.com/de-deutschland/products/detail/riegl-minivux-3uav.

[51] T. Xu, L. Xu, B. Yang, X. Li, J. Yao, Terrestrial laser scanning intensity correction by piecewise fitting and overlap-driven adjustment, Remote. Sens. 9 (11) (2017) 1090, http://dx.doi.org/10.3390/rs9111090.

[52] D. Carrea, A. Abellan, F. Humair, B. Matasci, M.-H. Derron, M. Jaboyedoff, Correction of terrestrial LiDAR intensity channel using Oren–Nayar reflectance model: An application to lithological differentiation, ISPRS J. Photogramm. Remote Sens. 113 (2016) 17–29, http://dx.doi.org/10.1016/j.isprsjprs.2015.12.004.

[53] S. Zhou, J.-R. Lin, P. Pan, Y. Pan, I. Brilakis, Impact of color and mixing proportion of synthetic point clouds on semantic segmentation, Autom. Constr. 171 (2025) 105963, http://dx.doi.org/10.1016/j.autcon.2025.105963.

[54] D. Crampen, J.M. Blankenbach, Data enrichment for semantic segmentation of point clouds for the generation of geometric-semantic road models, Front. Built Environ. 11 (2025) 1607375, http://dx.doi.org/10.3389/fbuil.2025.1607375.

[55] H. Thomas, C.R. Qi, J.-E. Deschaud, B. Marcotegui, F. Goulette, L.J. Guibas, Kpconv: Flexible and deformable convolution for point clouds, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 6411–6420.

[56] F. Noichl, F.C. Collins, A. Braun, A. Borrmann, Enhancing point cloud semantic segmentation in the data-scarce domain of industrial plants through synthetic data, Comput.-Aided Civ. Infrastruct. Eng. 39 (10) (2024) 1530–1549, http://dx.doi.org/10.1111/mice.13153.

[57] J. Xiang, Z. Lv, S. Xu, Y. Deng, R. Wang, B. Zhang, D. Chen, X. Tong, J. Yang, Structured 3D latents for scalable and versatile 3D generation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR, 2025, pp. 21469–21480.

[58] X. Wang, M. Ibrahim, A. Mansoor, H. Tareque, A. Mian, Automated road extraction and centreline fitting in LiDAR point clouds, in: 2024 International Conference on Digital Image Computing: Techniques and Applications, DICTA, IEEE, 2024, pp. 600–607, http://dx.doi.org/10.1109/DICTA63115.2024.00092.