RoadGen4Twins: A Modular Approach for Generating Multi-Purpose Geometric-Semantic Models for Digital Twins of Roads

David Crampen[1], Marcel Hein[2], Jörg Blankenbach[3]

Geodetic Institute, RWTH Aachen University, Germany
[1]crampen@gia.rwth-aachen.de
[2]marcel.hein@rwth-aachen.de
[3]blankenbach@gia.rwth-aachen.de

KEY WORDS: Digital Twins, Road Infrastructure, Automation, Building Information Modelling, Artificial Intelligence, Scan-to-BIM, Scan-to-Twin

ABSTRACT:

The development of novel and robust digital methods to support the maintenance of existing road infrastructure requires a large amount of harmonized data. Especially in the context of automated modelling having a large amount of matching data from different perspectives enables disruptive, new use cases that might largely impact the efficiency in maintenance of the built environment. Unfortunately, such data compositions are tedious to collect in real world applications, due to many influential factors, leading to deviations between multiple data sources and the sheer complexity in the process of creating a digital model. However, for deep learning applications, a large amount of carefully annotated data is necessary for robust estimations. In this contribution, we tackle this problem by presenting a novel procedural modelling and model configuration approach for generating homogeneous data combinations to step towards direct parameter estimation for machine learning approaches utilizing point clouds of roads and end-to-end model generation of digital road models.

1. Introduction

The recent rise of digital twinning for construction applications has led to an increasing demand for data relating to the target environments of digital twins. Especially in the context of management of built structures, it is important to firstly generate a digital representation of the system that closely resembles the relevant aspects of the use cases that shall be carried out. However, the creation of a digital model at a fixed point in time is not sufficient for a digital twin. The digital representation must also capture changes within the system boundaries. Additionally, the model has to be generated according to use case requirements, which have to be defined both for semantic as well as for geometric information. For this reason, we call these kinds of models geometric-semantic models. A large amount of high-quality data is crucial for the automated integration of changes and also to allow different model configurations. To employ an automated Scan-to-Twin process from survey data to such digital representations, several important steps with individual challenges are necessary. Though the steps of a Scan-to-BIM workflow are closely related to our workflow, the difference between our Scan-to-Twin workflow and a Scan-to-BIM workflow is the desired outcome, which for Scan-to-BIM typically is a digital model with as much detail as possible. Contrary, our Scan-to-Twin workflow targets the generation of highly specialized digital representations for specific digital twin use cases, leading to the necessity for a high range of adaptability of our model generation scheme. The general workflow consists of four general steps:

1. Data Capturing: The environment has to be surveyed using reality capture methods (e.g. photogrammetry and/or laser scanning).

2. Data Processing: The collected data has to be processed and semantically segmented into meaningful semantic object classes, which is done using deep learning approaches either working on image data or point clouds from laser scans. Since the goal is instance-wise modelling, the semantic segments are further split into instance segments for individual objects, such as road signs, trees or guardrails.

3. Geometry Fitting: Based on the instance segmentation outputs, object geometries are extracted from the individual object instances, which are represented as point cloud segments or image segmentation masks after step 2.

4. Model generation: Combining the semantic information extracted in step 2 and the geometric information obtained in step 3, the geometric-semantic model is derived.

While it can be argued, whether step 3 and step 4 belong together in the general view of Scan-to-BIM workflows, we split the steps in order to increase flexibility of a chosen modelling approach and to being able to formulate which information is needed and which is optional for modelling the road in a certain Level-of-as-is-Detail like we have defined in previous work (Crampen and Blankenbach, 2023a). Due to the sequential nature of the approach, several sources of uncertainty are introduced throughout the process, leading to an uncertain result. However, for digital twin applications it is crucial to evaluate the uncertainty of a used representation because it largely affects the validity of outputs of the specific use case. Validating each of the four steps is a non-trivial task, since no ground truth information for object geometry or even the model is available. For semantic segmentation, the ground truth comes from manually annotating point cloud or image data, which is very time-consuming due to the large amount of data necessary for the employment of deep learning. Even with enough data for training a model for step 2, validation of step 3 would require manual drawing of centrelines, boundary lines and other geometry to validate the automatically extracted geometry for road modelling. For semantic segmentation of point clouds, one approach to overcome laborious manual annotation of training data is synthetic point cloud generation from models to acquire annotated point clouds. Using such synthetic data to complement the real dataset has proven to be beneficial, as shown in (Inan et al., 2023). With this contribution, we aim at enabling synthetic data generation of roads not only as synthetic point clouds, but also the underlying parameter set of the

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W5-2024-103-2024 | © Author(s) 2024. CC BY 4.0 License.

desired model of the target environment, which is scanned, to combine the outputs of all four steps of the Scan-to-Twin process into one synthetic data vector. This vector is composed of a point cloud, corresponding pointwise labels, a set of geometric properties and the road model itself. We developed a parameter set with an extendable structure, a modelling regiment that uses this parameter set to create versatile road models and a parameter set generator, that can be used to either automatically generate wide ranges of road section parameter sets or manually create own road parameter sets in a quick and easy manner. We thereby enable validation of geometry extraction methods with synthetic data, while making a first step towards versatile model generation for the use in digital twin applications for roads.

The rest of this contribution is structured as follows: Section 2 concerns the general concepts of road modelling for existing structures with automatic methods. Section 3 briefly discusses existing approaches of synthetic data generation mainly focussing on synthetic point cloud generation, since in the construction sector, research focuses on the task of pure computer vision applications, while only very recently approaching end-to-end tasks for 3D modelling in general. In Section 4 we elaborate on our modelling approach, the metadata interface, we developed and our road generator GUI for generating such data interfaces, before we discuss and point out the next step towards synthetic data generation and improving modelling capabilities as well as limitations to our approach in Section 5. Lastly, we conclude our work in section 6.

2. Road Modelling

Current works on automatic modelling of road infrastructure mainly focus on case studies, where chosen road sections are modelled in a simplified manner to prove the feasibility of a customized workflow on specific data. Though roads seem to have a relatively simple geometry at first glance, until now mostly linear road section without lane transitions or junctions are in scope of research on fully automatic modelling of roads, because these scenarios are in fact very complex to formulate consistently and model accurately, based on survey data. If transversal and longitudinal inclination are considered, even modelling the road surface itself can become challenging, especially in cases, where modelling involves different Levels of detail like in our approach, since the surface has to be modelled true to deformation at the highest Level of Geometric Representation (Crampen and Blankenbach, 2023a). In general, modelling a road section from a high-level perspective requires the main axis of the road often denoted as the centreline as the most important asset, which is then used to reference all other objects such as separate lanes, road furniture or even vegetation in lateral proximity of the road segment. The centreline can be defined parametrically as single or connected splines, Bezier curves, clothoids, circular arcs or as discrete polylines, consisting of 3D points. The parametric definitions of the centreline are generally more efficient and lead to smooth curves, yet they must be interpreted correctly by a software for visualization and are not as easily adjustable to point cloud data with absolute positions as a mere polyline. If necessary, parametric representations can be discretized to polylines, while a spline can be fit to a polyline by utilizing smoothing filters for example (McCrae and Karang, 2009). From the Scan-to-Twin perspective, the parametric representation with the smallest cumulated error to the actual alignment, is best suited for the centreline. Since point clouds are discrete and sparse the first step for centreline extraction typically consists of the extraction of discrete points, that are then used to fit a parametric curve (Xiao et al., 2022), this process will potentially introduce inaccuracies (Nguyen et al. 2014), by suboptimal fits or inaccurate points in the first place. For this reason, the purest information is the set of points extracted from the point cloud, which is why we chose to use a polyline as geometry of our centreline in the first step.

In (Justo et al., 2021) the authors perform a case study for a Scan-to-BIM workflow that outputs an IFC file, with focus on the road and road furniture objects such as guardrails and road signs. Most works use pre-modelled assets for placing road furniture objects at locations alongside the road at positions identified by employing semantic point cloud segmentation, thereby simplifying the modelling process to reduce variability. Since the Scan-to-BIM workflow is a sequential process of individual complex tasks, the robustness of these approaches largely depends on the data source and local conditions during survey time. If, for example, a different sensor is used for data acquisition, this is very likely affecting the result of the process in a way that certain methods in the process have to be reparametrized or are no longer applicable at all. While most works are conducted similarly developing an approach to extract properties that sufficiently describe the road for modelling, our approach starts at the other end of this workflow by first defining the desired outcome of the Scan-to-Twin workflow to then draw conclusions about the necessary parameters to extract from reality capturing data. By turning the process around, we avoid limiting our own workflow to a specific scanning setup in the first step and focus on defining the required outcome for applicability for specific use cases of a Digital Twin. The general idea is creating a feedback loop instead of a one-way process to enable multistep validation. A schematic example of this idea regarding Digital Twins is shown in Figure 1.

While manual and interactive road modelling using existing Software products like ArcGIS CityEngine by ESRI (ESRI, 2024), Roadrunner by Mathworks (Mathworks, 2024) or Trian3DBuilder (TrianGraphics, 2024) can be used to create realistic road models with road crossings and lane transitions, they lack the necessary scalability, if the goal of a road model is to be used for synthetic point cloud generation only, since the creation of each model still requires significant modelling effort. With our approach we aim at minimizing manual labour, by allowing to create a single configuration once and generating a practically unlimited number of different road models, with characteristics true to the input configuration.

3. Data Generation

Computer vision tasks require large amounts of data. Depending on data complexity and the specific goal, the annotation process of such large amounts of data is extremely time-consuming. Therefore, the use of synthetic point cloud data promises high leverage for enriching the data basis efficiently. Synthetic point cloud data for semantic segmentation has been proven to boost performance of diverse machine learning methods, when combined with real data. Works like (Griffiths and Boehm, 2019) and (Deschaud et al., 2021) provide environments to generate synthetic data with a set of object classes in a fixed environment, while (Uggla and Horemuz, 2021) even partially randomize synthetic scenes to obtain higher diversity in the generated data. In general, it can be stated that the more diverse scenes and compositions are provided to a machine learning model, the higher the robustness of a trained model. However, randomization of scene generation requires a procedural modelling process to, enable valid representations. Also, each

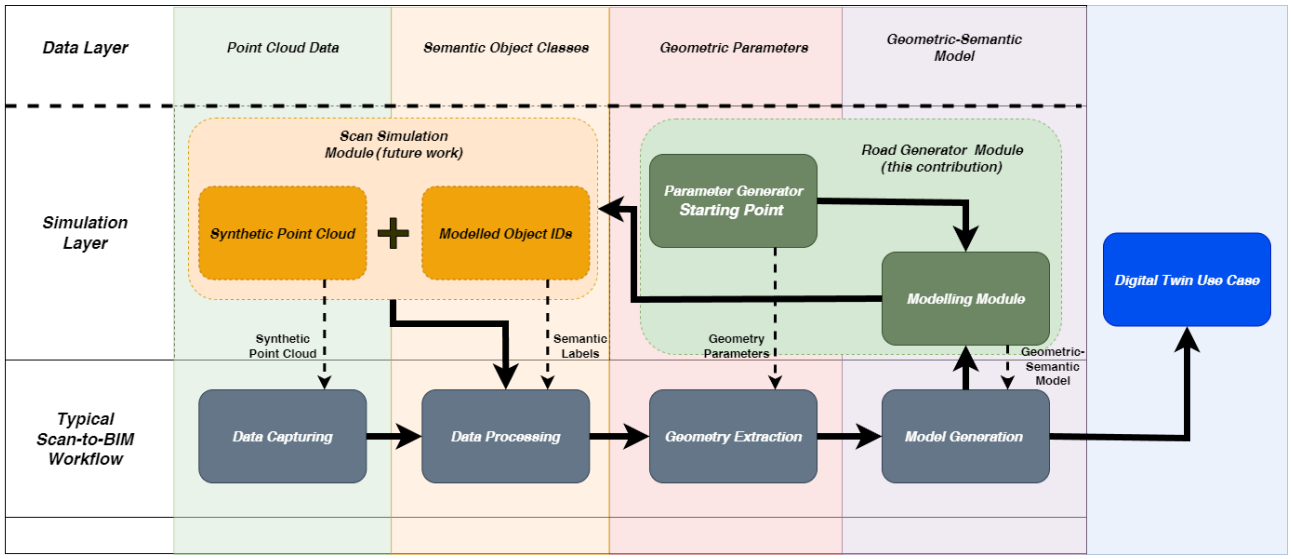

Figure 1: Overview of our Scan-to-Twin Feedback Loop.

variable parameter added to a procedural modelling approach increases diversity and corresponding dimensionality of what we call representation space. This on the other side makes it necessary to carefully formulate the modelling procedure to as far as possible avoid impossible configurations and errors in the model such as gaps between assets, wrong placements, or modelling artifacts. Applying procedural generation of roads and cities is recently largely employed for training autonomous driving models, due to the large number of different scenarios that can be created within a simulated environment like stated in (Li et al., 2020). Especially in the context of efficiently creating diverse scenes for computer vision tasks, the procedural modelling approach has several advantages over a manual modelling approach, with diversity and efficiency being the most important ones. For generating synthetic point clouds, the created models are virtually scanned, with a simulated laser scanner mostly using ray casting to replicate the physical properties of a real survey scenario, like developed in (Winiwarter et al., 2022) where several scanning systems can be simulated to generate synthetic point clouds with intensity values and colour, given the model was designed according to the requirements of their software called Helios++. Leveraging the model generation process to enable large and diverse annotated synthetic point cloud datasets, while also providing the underlying geometric properties of the initial models has not been done yet and has great potential to improve the overall Scan-to-BIM process, by creating a feedback loop for improving point cloud semantic segmentation models and geometry fitting respectively.

4. Novel Approach Automatic Road Model Generation

Our road model generator is composed of three main components, with the first one being the Python based road parameter generator, which is used to define a road segment with all its characteristics and assets, outputting this information into JSON interface files. The second component is the interface structure itself, encoding the geometry of a road section in a sufficient way for modelling different scenes. The third component is the modelling module, which uses the generated interface files for creating the road models. Next, we will elaborate on each of these three components, before explaining, the workflow leading to a wide variety of road models.

4.1 Road Generator

As depicted in Figure 1, we start our process at the geometry definition step. The road generator allows two options for creating a road. The first option allows manual drawing of road segments and adding road furniture objects at specific locations alongside the road axis, which is supposed to be a niche solution in case, where a specific road shape is desired. The second and main option allows defining a configuration for a characteristic road within a range of the configurable parameters that are then randomly chosen within the boundaries of the configuration to create random geometry sets. The generator was implemented with a graphic user interface (GUI) in python, which is depicted in Figure 2.

The sliders on the left side of the GUI are used in automatic mode to define a specific configuration where parameters such as curve radius, number of lanes, length of the road section and others can be specified. The placement of assets follows rules encoded in the generator, such that barriers or guardrails have to be present in curves or road signs having a fixed distance to the side of the roads. For the maximum curvature of the road geometry, we integrated the prescriptions from Germany's RAA code for highway design (RAA, 2008). The desired length of the road segment can be specified, and the centreline is generated according to the given curvature, with a set of constraints for retaining consistency between curved and straight subsection. The centreline is initially generated as polyline since it is the most basic representation possible. Though less efficient than parametric polynomial representations, we can thereby ensure that the parameters can be integrated into any applicable format, either directly, or by fitting a parametric representation to the initial polyline in the modelling module. The centreline is specifically generated point by point to ensure the absolute position as anchor points for the road, applying the curvature and height constraints after each iteration, which leads to consistency of the resulting centreline,

This contribution has been peer-reviewed. The double-blind peer-review was conducted on the basis of the full paper.
https://doi.org/10.5194/isprs-annals-X-4-W5-2024-103-2024 | © Author(s) 2024. CC BY 4.0 License.

by utilizing the respective directional vector. For start and end points additional constraints are applied for cases of connected section segments.

of the CityGML standard was developed for better usability reasons. Thereby, we iteratively improve modelling complexity and diversity of generated synthetic point clouds accordingly, while also enriching the versatility of the geometry information

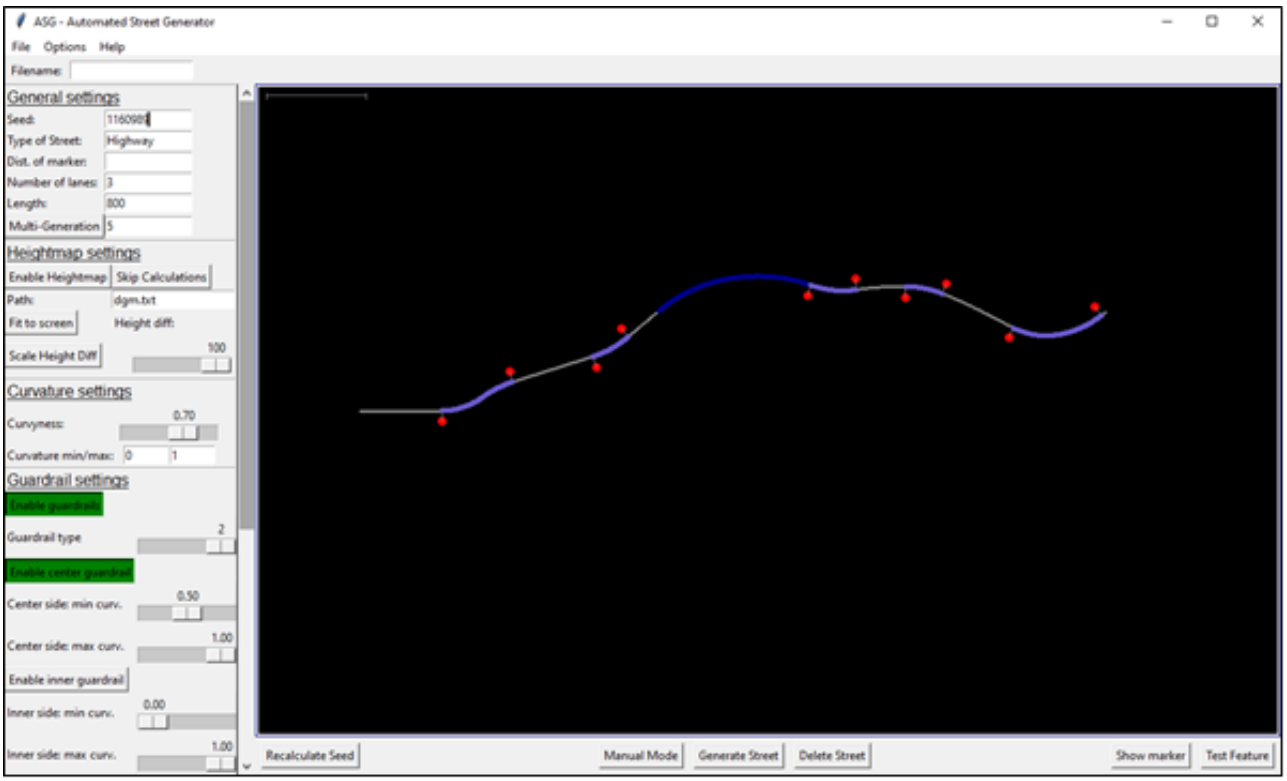

Figure 2: Road Generator GUI for Defining Road Scenes

In the manual mode the generation starts with drawing the road axis onto the canvas in the GUI. Guardrails can be interactively activated on the left side, the right side or on the median independently. After that, road furniture objects such as shield gantries or speed limit signs can be placed alongside the road, where the real position snaps to the set orthogonal distance to the road edge automatically. Additionally, in the current version of our generator, a digital terrain model can be imported and is visually represented with colour coding on the canvas to allow for integrating vertical changes of the road. This, however, only affects the longitudinal direction, while the transversal slope of the road remains unchanged. Since the change in side slope only partially follows the local curvature of the road and can differ with respect to drainage systems in specific environments, we neglect this aspect in the current version though aim at integrating it in future versions.

4.2 Data Interface

The resulting drawing is transformed into a JSON-based format for our metadata frame that forms an interface between the geometry generator and the modelling module we developed. The JSON metadata frame comprises centreline definitions, number of lanes on either side, median thickness, road furniture locations and types. We aim at implementing the object class structure defined in the schema of (Crampen et al., 2023b) in future versions of the geometry generator and aligning the structure of the JSON interface more closely to the structure defined in (Ledoux et al., 2019), where a JSON-based encoding integrated in our JSON interface. The current structure of this interface is depicted in Figure 4.

The general settings allow for the integration of a scaling factor for higher coordinate precision, with regard to floating point coordinates and an offset for georeferencing of the road section in a geodetic coordinate reference system and its assets similar to LAS format, which is used to exchange point clouds efficiently. As mentioned above, the current version mainly focuses on the customization of road furniture objects and the road space, therefore the layers and according thicknesses of the substructure are currently fixed in the modelling module, as well as the occurrence of vegetation like trees and small gras areas or rocks and vehicles on the road, which's positions are randomly chosen in the modelling process. Roads are defined by a set of attributes like class, median width, surface thickness and an arbitrary number of lane objects, which in turn have attributes like lane width, direction, which is oriented from start to end point of the generated axis and width of road markings. For point objects in the road furniture section, the asset is specified with its class, a position and the driving direction. As stated before, this structure is constantly evolving towards supporting more object classes and therefore enabling more versatile scenes to be generated. We additionally allow for changing the representation of functional units like road space or road furniture assets according to the concept in (Crampen et al., 2023b). However, complexity leads to higher effort in customizing the desired scene and makes the manual approach less easy to use. Therefore, we aim at providing different layers

of customizability in order to leave it optional to generate large numbers of roads with small differences as well as more complex changes, which in turn have to be implemented in our modelling module.


```
{
    "settings": {
        "general_settings": {
            "offset": {
                "x": 0,
                "y": 0,
                "z": 0
            },
            "scaling": {
                "x": 0.01,
                "y": 0.01,
                "z": 0.01
            }
        },
        "output_settings": {
            "instancing": true,
            "subsurface": true,
            "signs": true,
            "guardrails": true,
            "surrounding": true,
            "vehicles": true
        }
    },
    "streets": [
        {
            "type": "Highway",
            "mid_section_width": "1.5",
            "surface_thickness": "0.2",
            "lanes": [
                {
                    "direction": "normal",
                    "width": "3.5",
                    "marking_width": "0.15"
                },
                {
                    "direction": "reverse",
                    "width": "3.5",
                    "marking_width": "0.15"
                }
            ],
            "coords": [
                {
                    "x": 0.9999921602692798,
                    "y": 0,
                    "z": 0.0038746199084600304
                },
                "..."
            ]
        }
    ],
    "road_furniture": {
        "signs": [
            {
                "type": "speed",
                "pos": {
                    "x": 182.6243387194927,
                    "y": 0,
                    "z": 11.095776024413576
                },
                "side": "normal",
                "offset": 0
            },
            "..."
        ]
    }
}
```


Figure 3: Example of Interface Structure in JSON-Format

4.3 Modelling Module

Our modelling component was implemented using the Software Houdini (Houdini Development Team, 2024) developed by Side FX. It is a software for procedural modelling, which allows us to incorporate the information from our interface at different steps in the modelling procedure to create smaller or larger changes in the result. We have chosen Houdini, since it provides a direct interface to the Unreal Engine (Unreal Engine Development Team, 2019), which will be used in the next step to develop the synthetic point cloud generation scheme. As discussed in section 2 procedural modelling is a great fit for our target of generating a wide variety of models, however the more input parameters the modelling procedure allows to vary, the higher the risk of errors in the resulting model. If for example side slopes of the road change too quickly, this can result in the whole model being erroneous, because errors at one step in the process propagate in the same large way that valid changes lead to a largely different model. For a complex modelling process like ours, it is nearly impossible to completely avoid configurations, where the result is unrealistic. Still, when using 100 random configurations in our road generator, we reach more than 95% valid results, with regard to appearance. When choosing more extreme values in the configuration, clearly visible errors occur in approximately 2% of cases, that will affect point cloud generation. During modelling, we distinguish between point and line objects. Line objects are the lanes, the road markings, the lateral area, such as embankments and barriers or guardrails, while point objects are positioned via a single point and include road signs, trees, rocks, shield gantries, vehicles and signal posts. As discussed in section 4.1, point object positions are referenced to a point on the centreline and its distance, while line objects are referenced with a starting point and an end point in the same way. The shapes of the different objects in the highest Level of Geometric Representation are exchangeable, pre modelled assets for point objects, with orientation being defined according to the centreline direction at their position and through a cross-section for line objects, which is exchangeable as well. For guardrails and barriers start and end points, we technically place a point object in order to make the starting point more realistic. The module is structured into five subtasks, lane creation, lateral area generation for embankments and vegetation, road sign generation, safety installations and a header module for changing the Level of Geometric Representation. As an example, Figure 4 depicts the process for lane generation in Houdini. It can be noted that, depending on the specific demands, we have developed several submodules, for integrating road damages like ruts for example. These submodules, however, are still a matter of change, since we aim at allowing a variety of post-processing options for adapting the base model created with our main module to a use case specific model, that can be employed in Digital Twin applications in the future. To visualize modelling, Figure 6 shows chosen steps of the process. Figure 6 shows the model compared to the generated road in the road generator GUI.

The final goal is to use the same modelling module we developed for this contribution from the perspective towards augmenting real-world survey data with synthetic point clouds, for final model generation as well, where the interface is filled with information extracted from real point cloud data, thereby closing the loop of improving modelling capabilities for data augmentation and Digital Twin models accordingly.

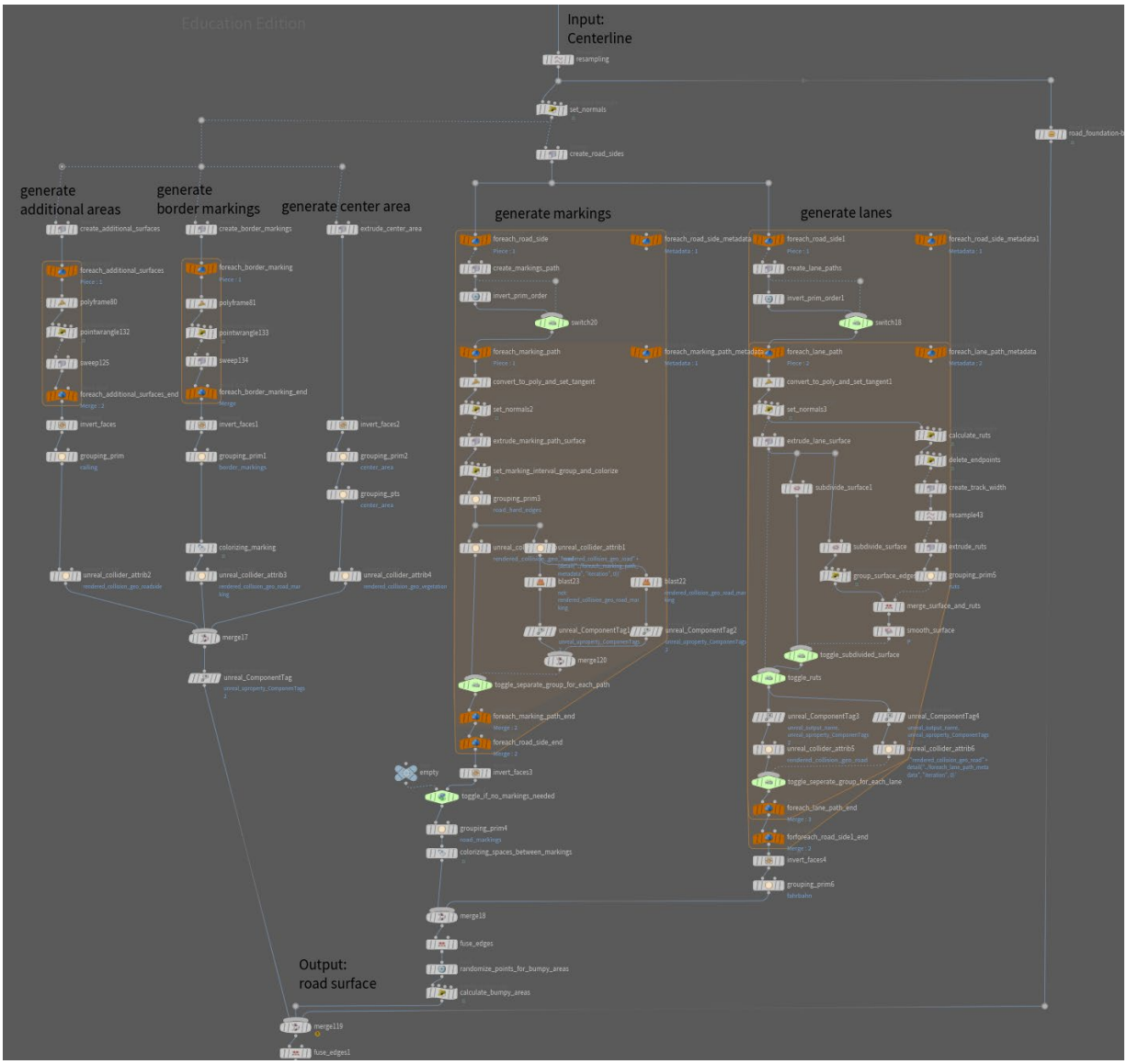

Figure 4: Modelling Submodule for Lane Generation

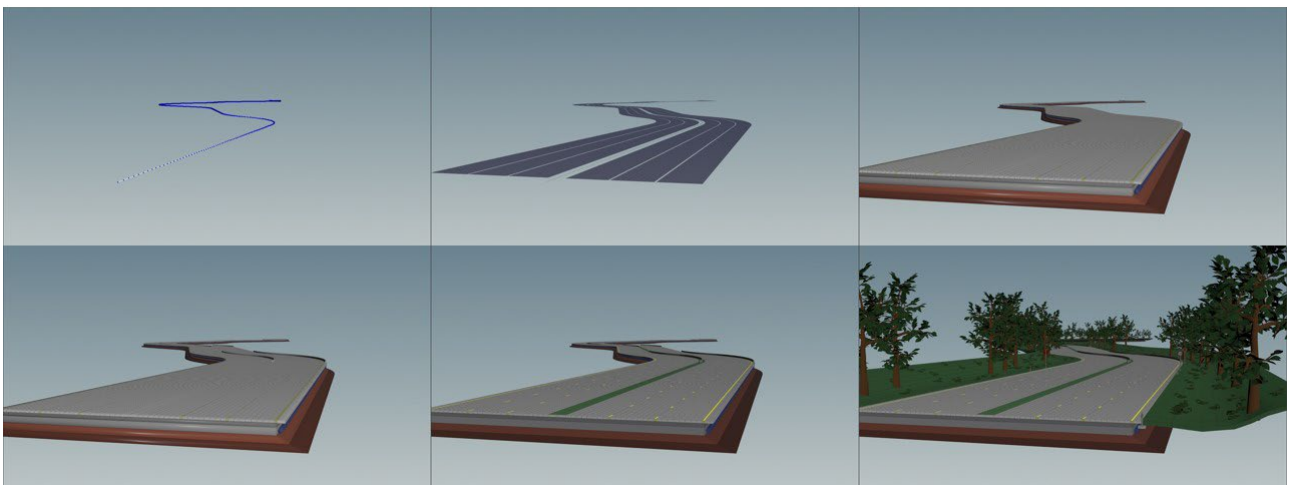

Figure 5: Modelling Process of a Road Section

5. Discussion

In the current version the geometry generator supports highways without lane transitions or junctions, since generically formulating modelling rules for junctions with a wide variety of possible installation of merging roads and lanes according to the local driving rules is very complex, would require several parameters like the crossing point of roads, angles, start and end of the junction, style of road merge and several more, making it nearly impossible to quickly define within the road generator, with the fact in mind that all the used parameters have to be accurately extracted from a point cloud in the future to close the feedback loop. Precise modelling of such lane departures requires the integration of topological relations with predecessor/successor relationships like existing in standards like OpenDrive (ASAM, 2023), GDF5.1 (ISO 20524-1:2020) or RoadXML (Chaplier et al., 2010), which are used for driving simulation and navigation. Environments created within these standards however either use simplified geometry to minimize the potential errors, if generated automatically like we mentioned in section 4.3 or employ manual modelling.

In our work we focus mainly on highways, which technically have no junctions, though highway entrances and exits can be seen as intersections, that are currently implemented into our modelling approach. Similarly, lane transitions are another important characteristic of highways, which are not easy to formulate and consistently model either. We will incorporate them using additional parameters, such as defining a start and end point of a lane on the centreline as well as a transition direction and start and end point of the transition itself. Accordingly, road markings have to be adjusted to follow the status of the lanes separated by markings. It is obvious that many features have to be added in order to create more realistic models, nonetheless it has to be kept in mind that all added parameters of the modelling approach have to be extractable from reality capturing data or other highly available geospatial data sources to be useful for later model derivation from real-world roads. To deal with that problem step-by-step, we design the whole process to be iterative, since for very specific parameters there exists no current solution yet. This is why we gradually improve modelling capabilities according to our progress in the other steps, to build a solid foundation that is upwards compatible. There are many steps in the complete Scan-to-Twin process, that change the outcome in a potentially substantial way, so validation of several steps down the line is essential. By completing the feedback loop developing the process of model generation and combining it with a synthetic point cloud generation workflow, we aim at reaching our first main goal to improve semantic segmentation of point clouds. However, at the same time, we also enable validating geometry extraction, by using the information of our interface as ground truth in that step. Even tough, in that way validation is only available for synthetic roads, we believe that it supports improving methods in that step and hope to also gain insight into necessary improvements in modelling throughout the optimization of previous steps. Figure 7 shows two examples of simple road models generated with our approach, proving that we are capable of generating diverse road sections even based on the same underlying axes. Through the capability of customizing models and improving occurrences of minority classes and underrepresented scenes, we are convinced of our approach's potential to support semantic segmentation, when generating synthetic point clouds with them. Another important objective for future work is developing high resolution textures and material definitions for the models, which would both make them more realistic for use cases, but would also enable more realistic point cloud generation, since scanner features like intensities and number of returns could be derived, and also realistic colouring could be achieved in the synthetic data. Lastly, we will incorporate direct DSM or DTM integration, to make the surrounding ground surface more realistic as well.

6. Conclusion

It has become obvious that there lies great potential in developing a virtual process running in opposite direction of the real-world sequential workflow, since it allows for several useful interfaces between real-world workflow and simulated workflow. In a way, one could argue that this approach already resembles the Digital Twin idea, but in the process perspective.

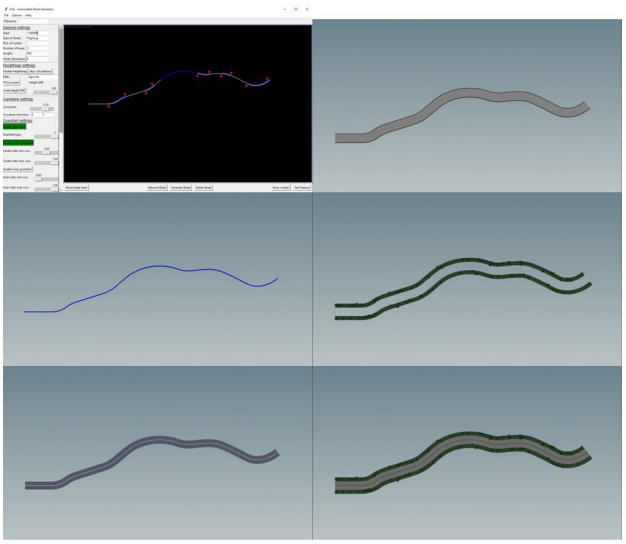

Figure 6: Top-Down View of a Generated Road Compared to the Road Generator

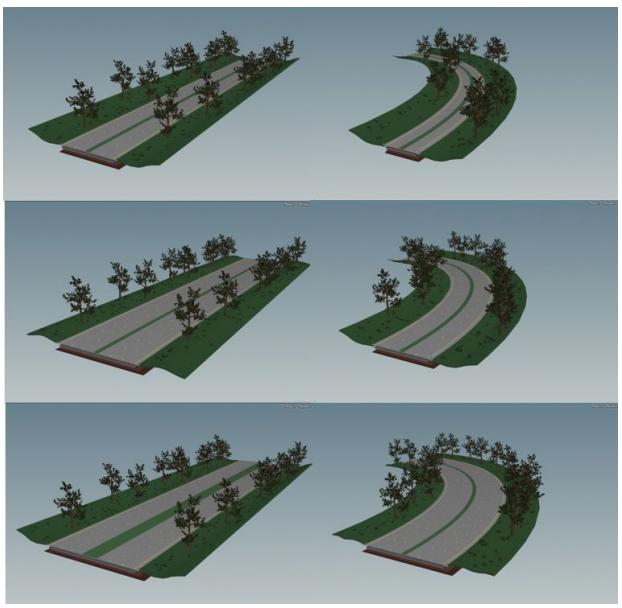

Figure 7: Simple examples of automatically generated road models

Given the Scan-to-Twin workflow is regarded as the physical component and the simulation layer represents the digital component, the sensor data is represented by the input data at the start of the whole workflow, in form of real-world point

clouds, while the connections are formed first at the start where synthetic data is incorporated into the data processing step and in the model generation step. In the future even more connections could be established, symbolized by the dashed arrows in Figure 1 at each step of the workflow. The overall goal is optimizing the Scan-to-Twin workflow on multiple levels to generate accurate digital representations for the Digital Twin of the road. Referring to the analogy of a digital twin, automation of optimization steps in the envisioned system, would be groundbreaking, while obviously being difficult to establish comprehensively, due to the high complexity of the chosen automatic process.

Aknowledgements

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

References

Asam, 2023: OpenDRIVE, Version 1.8.0, https://www.asam.net/standards/detail/opendrive/ (30.01.2024)

Chaplier, Julien; Nguyen That, Thomas; Hewatt, Marcus; Gallée, Gilles, 2010: Towards a standard: RoadXML the road network database format. In: Proceedings of: Driving Simulation Conference Europe.

Crampen, David; Blankenbach, Jörg 2023a: LOADt: Towards a Concept of Level of as-is Detail for Digital Twins of Roads. In: Proceedings of 30th EG-ICE International Conference on Intelligent Computing in Engineering.

Crampen, David; Hein, Marcel; Blankenbach, Jörg, 2023b: A Level of as-is Detail Concept for Digital Twins of Roads - A Case Study. In: *Recent Advances in 3D Geoinformation Science.* DOI: 10.1007/978-3-031-43699-4.

Deschaud, Jean-Emmanuel; Duque, David; Richa, Jean Pierre; Velasco-Forero, Santiago; Marcotegui, Beatriz; Goulette, François, 2021: Paris-CARLA-3D: A Real and Synthetic Outdoor Point Cloud Dataset for Challenging Tasks in 3D Mapping. In: *Remote Sensing* 13 (22), S. 4713. DOI: 10.3390/rs13224713.

ESRI, 2024: ArcGIS CityEngine, https://www.esri.com/en-us/arcgis/products/arcgis-cityengine/overview

Forschungsgesellschaft für Straßen- und Verkehrswesen, 2008: Richtlinien für die Anlage von Autobahnen. RAA. Ausgabe 2008. Köln: FGSV-Verl. (FGSV R1 - Regelwerke, 202).

Griffiths, David; Boehm, Jan, 2019: SynthCity: A large scale synthetic point cloud. http://arxiv.org/pdf/1907.04758v1.

Houdini Development Team, 2024. Houdini Version 19.5.605, SideFX.https://www.sidefx.com/products/houdini (30.01.2024)

Inan, Burak Alp; Rondao, Duarte; Aouf, Nabil, 2023: Enhancing LiDAR Point Cloud Segmentation with Synthetic Data. In: 2023 31st Mediterranean Conference on Control and Automation (MED). 2023 31st Mediterranean Conference on Control and Automation (MED). Limassol, Cyprus, 26.06.2023 - 29.06.2023: IEEE, S. 370–375.

ISO 20524-1:2020, 2020: Intelligent transport systems.

Justo, Andrés; Soilán, Mario; Sánchez-Rodríguez, Ana; Riveiro, Belén, 2021: Scan-to-BIM for the infrastructure domain: Generation of IFC-compliant models of road infrastructure assets and semantics using 3D point cloud data. In: *Automation in Construction* 127, S. 103703. DOI: 10.1016/j.autcon.2021.103703.

Ledoux, Hugo; Arroyo Ohori, Ken; Kumar, Kavisha; Dukai, Balázs; Labetski, Anna; Vitalis, Stelios, 2019: CityJSON: a compact and easy-to-use encoding of the CityGML data model. In: *Open geospatial data, softw. stand.* 4 (1). DOI: 10.1186/s40965-019-0064-0.

Li, Quanyi; Peng, Zhenghao; Zhang, Qihang; Liu, Chunxiao; Zhou, Bolei, 2020: Improving the Generalization of End-to-End Driving through Procedural Generation. http://arxiv.org/pdf/2012.13681v2.

Mathworks, 2024: Roadrunner, https://mathworks.com/products/roadrunner.html

Nguyen, H.H., B. Desbenoit, und M. Daniel. „Realistic Road Path Reconstruction from GIS Data“. Computer Graphics Forum 33, Nr. 7 (Oktober 2014): 259–68. https://doi.org/10.1111/cgf.12494.

Uggla, Gustaf; Horemuz, Milan, 2021: Towards synthesized training data for semantic segmentation of mobile laser scanning point clouds: Generating level crossings from real and synthetic point cloud samples. In: *Automation in Construction* 130, S. 103839. DOI: 10.1016/j.autcon.2021.103839.

Unreal Engine Development Team, 2024. Unreal Engine Version 5, Epic Games. https://www.unrealengine.com (30.01.2024)

TrianGraphiccs GmbH, 2024: Trian3DBuilder, https://trian3dbuilder.de/

Winiwarter, Lukas; Esmorís Pena, Alberto Manuel; Weiser, Hannah; Anders, Katharina; Martínez Sánchez, Jorge; Searle, Mark; Höfle, Bernhard, 2022: Virtual laser scanning with HELIOS++: A novel take on ray tracing-based simulation of topographic full-waveform 3D laser scanning. In: *Remote Sensing of Environment* 269, S. 112772. DOI: 10.1016/j.rse.2021.112772.

McCrae, James, and Karan Singh. Sketching Piecewise Clothoid Curves“. *Computers & Graphics* 33, Nr. 4 (August 2009): 452–61. https://doi.org/10.1016/j.cag.2009.05.006.

Xiao, Fanghong, Ling Tong, Yuxia Li, Shiyu Luo, und Jón Atli Benediktsson. „A General Spline-Based Method for Centerline Extraction from Different Segmented Road Maps in Remote Sensing Imagery“. *Remote Sensing* 14, Nr. 9 (26. April 2022): 2074. https://doi.org/10.3390/rs14092074.