DEPARTMENT OF AEROSPACE AND GEODESY

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis

Semantic-based Geometry Refinement of 3D City Models for Testing Automated Driving

B.Eng. Olaf Wysocki

DEPARTMENT OF AEROSPACE AND GEODESY

TECHNICAL UNIVERSITY OF MUNICH

Master's Thesis

Semantic-based Geometry Refinement of 3D City Models for Testing Automated Driving

Author:	B.Eng. Olaf Wysocki
Supervisors:	Univ.-Prof. Dr. rer. nat. Thomas H. Kolbe
	Dr.-Ing. Ludwig Hoegner
	M.Sc.M.Sc. Benedikt Schwab
Submission Date:	04.09.2020

With this statement I declare that I have independently completed this Master's thesis. The thoughts taken directly or indirectly from external sources are properly marked as such. This thesis was not previously submitted to another academic institution and has also not yet been published.

Munich, 04.09.2020 Olaf Wysocki

Acknowledgments

The thesis was completed during the coronavirus pandemic. Therefore, I would like to thank all the medical staff in Germany and all over the world for their fight. Thanks to them, people like me could continue their work and focus on their field of expertise.

I would like to thank the Chair of Geoinformatics at the Technical University of Munich, AUDI AG and the project team *SAVe* (Funktions- und Verkehrs-Sicherheit im Automatisierten und Vernetzten Fahren) for giving me the opportunity to write this master thesis. I would like to thank for supervision by the team of the Chair of Photogrammetry and Remote Sensing at the Technical University of Munich which significantly aided this multidisciplinary work. Especially I would like to thank my supervisors Univ.-Prof. Dr. rer. nat. Thomas H. Kolbe, Dr.-Ing. Ludwig Hoegner, and M.Sc. Benedikt Schwab whose advice has been indispensable to complete this thesis.

I am grateful for the *DAAD's* (Deutsche Akademische Austauschdienst e.V.) financial aid through my master studies at the Technical University of Munich which allowed me to fully focus on my academic development.

I would like to thank my partner Magdalena for her constant mental support throughout the years. Behind every successful man, there is a wise woman.

As this thesis brings me to the end of a certain stage of my life, I would like to thank my parents and sister without which this journey would not be possible.

Abstract

The testing of automated driving functions has to be improved in order to allow for a broad usage of automated and autonomous vehicles on public roads. The current approach assumes that a testing field for such cars can and should be a virtual representation of a real-world scene. This implies a need for a reliable, up-to-date, accurate, and semantically rich 3D models of a road space environment. Moreover, arising questions about satisfactory levels of semantics, temporal, and spatial resolution have no definite answers.

In order to create 3D maps, point clouds acquired in aerial and mobile mapping campaigns are often utilised. However, available automatic methods for 3D models creation do not completely fulfil demanded requirements. Those models either lack detailed geometry representations or have poor semantics. The models which are manually created cover those gaps but time-consuming modelling process prevents scaling of 3D maps for wider areas. The recent trends in 3D maps creation focus on reconstructing objects without taking into account semantics and geo-contextual information of already created 3D maps.

Therefore, the goal of this work was to create a method which allows to automatically enhance the geometry of existing 3D road space models by means of available point clouds. A workflow had to be easy to use and enable user-friendly customisation of an expected refinement level even for a non-expert user in the field.

The methodology is based on novel approaches from fields of geoinformatics and photogrammetry which are inevitable to achieve the goals of the project. The FME software serves as a backbone of the project where LASTools, Python scripts and the external software MeshLabServer are integrated. Thanks to that, the whole processing and reconstruction workflow is steered by one software. Validation of the methodology and visualisation of results are performed in the state-of-the-art city models managing tool 3D city Database suite and the game engine Unreal Engine which is used in automated driving simulators like CARLA. Additionally, the possible semantics enrichment of models representing roads is shown.

The city centre of Ingolstadt, Bavaria, Germany served as a testing ground for the methodology. The datasets of LoD1, LoD2, LoD3 buildings and roads from HD Map in the CityGML standard area were used.

Contents

1. Introduction

According to the Cambridge Dictionary the word *Refinement* has two meanings, namely *the process of making a substance pure* or *a small change that improves something* [Cambridge Dictionary, 2020]. Those two explanations precisely describe this Master Thesis project. On the one hand, the focus is to filter and select only relevant subsets of point clouds depicting specific objects (thus *the process of making a substance pure*) and on the other hand the geometries that should be created already exist but they are coarse and should be more detailed (thus *a small change that improves something*).

1.1. Motivation

Currently, 3D models of cities are gaining popularity around the world. Numerous municipalities in Europe, North America, Asia, and Australia introduced 3D Geographic Information Systems (GIS) for their cities. This trend enabled the improvement of accuracies of spatial-based analysis (like noise propagation in an urban environment) and enhanced visual attractivity of city models enabling new applications (like line-of-sight presentation) [Biljecki, Stoter, Ledoux, et al., 2015]. One of the most utilised standards in 3D city models is CityGML [Angermann, Donaubauer, Graw, et al., 2019]. It allows to visualise objects and store respective semantic information. In order to create a geometrical representation of such objects within a town's border, there is a need for spatial information. Geoinformation is very often delivered using Airborne Laser Scanning (ALS) or aerial imageries which e.g. assure spatial accuracy of buildings of around few meters and usually very low temporal accuracy varying from months to years [Angermann, Donaubauer, Graw, et al., 2019]. The Level of Detail 2 (LoD2), which supports modelling of buildings having walls and roof shapes, is an established standard for a city models creation. Recently, Level of Detail 3 (LoD3) becomes more interesting for municipalities because it enables more realistic visualisation of objects and more complex and accurate spatial analysis [Angermann, Donaubauer, Graw, et al., 2019]. At this level, details like windows, doors can be modelled. However, this kind of complex geometry has to be created based on highly accurate spatial information.

As the autonomous driving technology is evolving more and more spatial data is and will be acquired by Mobile Laser Scanning (MLS). This trend impacts the most the number of spatial data within road spaces as they are of high interest for automated vehicle positioning [Albrecht, Kraus, Zimmermann, & Stilla, 2019; Wysocki, 2019] and car sensors testing [Schwab & Kolbe, 2019]. Therefore, the availability of such data, their spatial and temporal resolution increases. Companies like 3D Mapping Solutions, HERE, TomTom or Audi AG send their mobile scanner fleets to gather spatial information from around the world at cm accuracy

level and those vendors guarantee a high rate of survey repetition [HERE, 2020; TomTom, 2020]. As one of the examples of this trend can serve the project Driving Dataset Tutorial where certain areas are possible to download free of charge and the vendor provides a tutorial on how to utilise such datasets [Audi Electronics Venture, 2020]. Moreover, autonomous cars in the future will be able to gather information and pass it to a database instantly which means that temporal resolution will achieve near real-time level [Chellapilla, 2018; HERE, 2020]. Taking into consideration those trends there should be a link between geospatial and automotive branch which will allow enriching city models with spatial information gathered by MLS with its high accuracy and temporal resolution and in the future by automated cars themselves. On the other hand, standards like CityGML can be a very helpful source of semantic and spatial information for autonomous and automated vehicle simulations. Thus, the two models can contribute to each other.

The complexity of different spatial scenarios is disabling fully automated LoD3 (or higher) reconstruction on a big scale. Moreover, it is hard to validate the results of such reconstructions as detailed ground-truth datasets are hardly available. With the increasing number of MLS fleets, the ground truth information for the validation of road areas reconstruction could serve as a solution to this problem.

The other aspect is a temporal resolution of utilised data and required details for the LoD3 standard. ALS data have mostly month to year temporal resolution which is enough to create LoD2 buildings. Also, the aerial position of acquiring point clouds results in many occlusions which often do not allow to recreate properly e.g. buildings façades [Wang, Peethambaran, & Dong, 2018].

1.2. Research question

Nowadays, most of the LoD3 buildings, city furniture, and other highly detailed CityGML objects are created manually. This requires a highly skilled operator in a specific tool to create such geometries and assign attributes to an object afterwards. This solution is error-prone as it is a subjective approach. Moreover, even if an item is created it has certain deviations and simplifications in comparison to the real-world situation.

Furthermore, road space objects within High Definition Map (HD Map) are currently described by simple geometries like cylinders and polyhedrons which replicate in strong deviations in the geometrical representation of objects. Those models are not sufficient for testing sensors and their behaviour on a road.

This thesis attempts to answer the question of how to refine the geometries of the CityGML objects derived from the HD Map to obtain minimised deviations from the real-world situation depicted by dense Light Detection And Ranging (LiDAR) measurements. This objective arises the following research questions and hypotheses which should be answered within the scope of this thesis.

Research questions and hypotheses:

1. How to implement a reproducible automatic surface reconstruction workflow for non-professionals?
2. *A priori* road space information (e.g. obtained from OpenDRIVE datasets) can enhance the results of point cloud segmentation algorithms
3. How can Mobile Laser Scanning point clouds having intensity values be utilised in the surface reconstruction of semantic vector objects?
4. To what extent and how can the workflow be parametrised by a user?

1.3. Structure and content

The structure of this thesis presents as follows. In Chapter 2 theoretical background of presented problems is explained. Chapter 3 describes a concept as well as implementation details of the solution. Within Chapter 4 all tests and practical examples are shown. The discussion of results can be found in Chapter 5 of this thesis. Chapter 6 sums up the presented work and gives an outlook to further possible developments for the project. Within the Appendix FME Workspace files, Python scripts and configuration files are added.

2. Fundamentals

Nowadays, models having road space scenes are developed in parallel for the needs of industrial sectors like automotive [VIRES Simulationstechnologie GmbH, 2020b], architecture [Autodesk, 2020], urban planning [Angermann, Donaubauer, Graw, et al., 2019; Gröger, Kolbe, Nagel, & Häfele, 2012], civil construction [Autodesk, 2020]. Not all of them have the road and its vicinity as the main component. Very often those models are developed independently from each other even if their standards overlap in certain areas – up to now, there is no one road space model solution which serves all different branches of industries needs and also the cooperation between vendors could be barely observed. As such example can serve CityGML and OpenDRIVE standards developed mainly for cost-effective sustainable maintenance of 3D city models [Gröger, Kolbe, Nagel, & Häfele, 2012] and for traffic and sensor simulations to describe entire road networks w.r.t. all data belonging to the road environment [VIRES Simulationstechnologie GmbH, 2020b], respectively. The CityGML allows to model roads with semantic and geometrical representation on different levels of details as roads are an undividable part of the urban area. On the other hand, OpenDRIVE was developed to only support modelling of roads networks which reflects in highly developed topological network relation but has limits in the geometrical representation of road-side objects and semantic information.

According to Frost & Sullivan, fully autonomous driving systems will be available from the third decade of the 21st century [Frost & Sullivan, 2018]. Currently, a few fully automated (L4) solutions exist like bus shuttles in the city of Sion, Switzerland where the bus does not need a driver. However, it drives on only two predefined routes within a town centre and can reach a maximum speed of around 20 km/h [Team SmartShuttle, 2018]. In 2017, Audi AG announced an L3 car with all the necessary systems for hands-off highway driving [Audi AG, 2017]. Nevertheless, in many countries, the legislation process allowing such a car to drive on public roads has not been released yet [Frost & Sullivan, 2018; Taylor, 2017]. For example, in 2018, the United Kingdom had no legislation allowing for L3 or more advanced self-driving vehicles [de Prez, 2018] and the similar issue was in the United States [Davies, 2018]. Those examples show two main interconnected challenges of automated function development. First, the complexity of traffic scenarios, which in the case of a highway could offer a relatively small amount of events to enormous possibilities within an urban area (see Figure 2.1). This, however, has a link to the second challenge – a process of obtaining a homologation that depends on governmental regulations. So, even if a technological stack allows us to have a ride with a driverless or semi-driverless vehicle there is still a need for a credibility assessment of such a system by governing bodies. Thus, authorities, in order to assure the safety of their citizens, have to have certain proof of a car system reliability which can be obtained through testing and simulation.

2.1. Testing of automated vehicle functions

Vehicle simulations have various needs as there exist different stages of car automation. The level of vehicle automation describes self-driving capabilities of a car (see Figure 2.1). The word autonomous referrers to level 5 and automated to range L2 – L4 within the scope of this project.

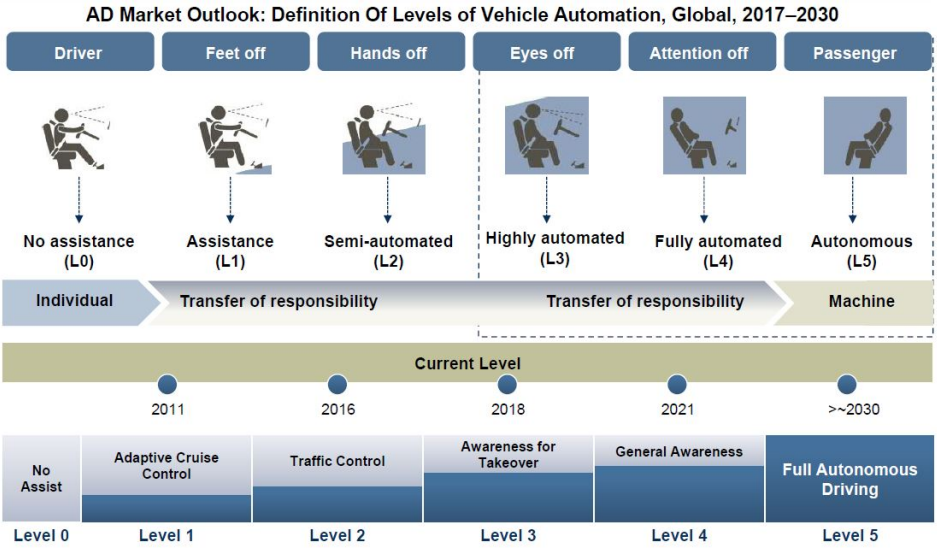

Figure 2.1.: Levels of automation [Frost & Sullivan, 2018]

There are certain measures depicting the level of complexity of road scenarios. According to Schuldt [Schuldt, 2017], there are several categories which can define this level:

- Lack of transparency (incomplete observations of the system and thus the need for indicators usage to extract relevant information)
- Vehicle own momentum (system only without taking into consideration other actors)
- Interconnectedness (interaction with other road users)
- Novelty (depends solely on the system thus constant in scenario description)
- Number of conditions per element (assumed to be constant across all scenarios)
- Number of actors (number of road users)
- Openness of the target situation (lack of clear objectives)
- Polytely (presence of multiple simultaneous goals)

Each category can be further scaled from 1 (low) to 5 (very high) describing qualitative measure. The intricacy of different road scenarios is shown in Figure 2.2 [Schuldt, 2017].

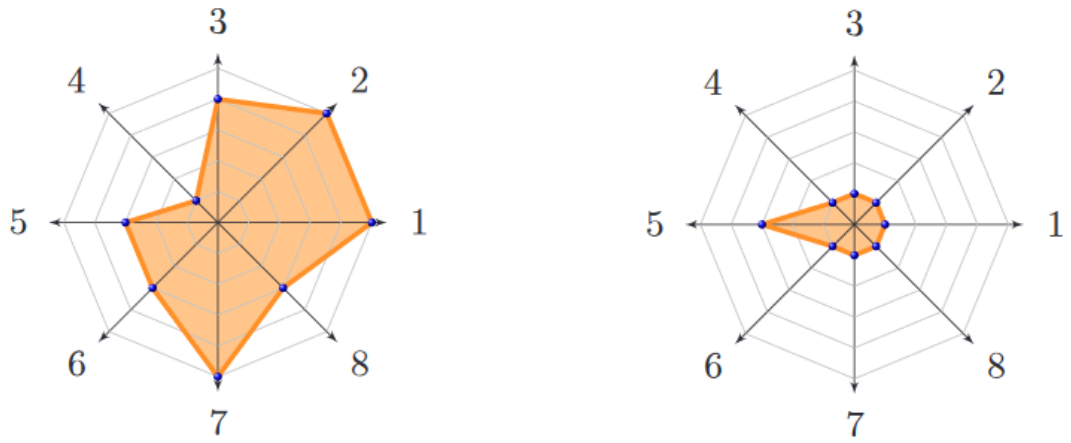

Figure 2.2.: Spider charts representing measures of scenarios complexity. Road in urban area, with signposted crossroads (left) and highway with little traffic (right). 1-lack of transparency, 2-vehicle own momentum, 3-interconnectedness, 4-novelty, 5-number of conditions per element 6-number of actors, 7-openness of the target situation, 8-polytely [Schuldt, 2017]

The experts in the field agreed that the right method to deliver such verification is to use Everything-in-the-Loop (XiL) approach - see Figure 2.3 [Riedmaier, Nesensohn, Gutenkunst, et al., 2018]. Although the facilities like test town in Singapore for driverless vehicles with traffic lights, bus stops, skyscrapers and a rain machine to recreate its extreme weather conditions exist [KPMG International, 2019], there is a major shift to test and simulate most of the system components in virtual testbeds which are cost-effective, safe and reproducible [Riedmaier, Nesensohn, Gutenkunst, et al., 2018]. Moreover, it is hardly possible to simulate as many and as various circumstances in real-world like in a virtual testbed. For example, it was estimated that to decrease the expected distance between two fatal accidents on a highway it is needed to conduct test driving through 6.6 billion kilometres [Wachenfeld, Junietz, Wenzel, & Winner, 2016]. Furthermore, virtual testbeds are capable to simulate changing weather conditions. Different pedestrian or cyclists actions would need hundreds of actors to play a specific role in a scenario which is an expensive and time-consuming effort.

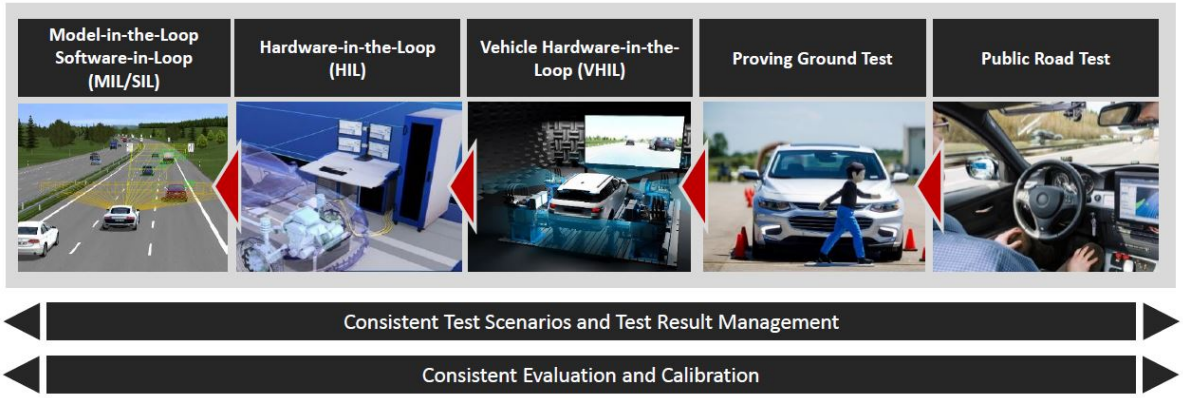

Figure 2.3.: Everything-in-the-Loop [Riedmaier, Nesensohn, Gutenkunst, et al., 2018]

However, the important requirement before a virtual testbed can be used is to validate it. Artificial test ground is validated against sensor responses which should be adequate to the real-world situation. This means that virtual testbed should comprise a minimal set of reality to be utilised. To achieve this goal real test drive can be compared with a virtual test drive by applying the same scenarios and evaluating results [Riedmaier, Nesensohn, Gutenkunst, et al., 2018]. In order to simulate vehicle functions, virtual testbeds have to contain essential dynamic information such as pedestrians or other cars likewise static objects like buildings or street lamps. Moreover, current models can represent digital-twin or completely fictional scenes [Schwab & Kolbe, 2019].

Should the validity requirement be fulfilled the scenario-based testing could be performed. The level of abstraction should be high for functional scenarios (item definition and the hazard analysis and risk assessment during the concept phase of the ISO 26262) and decrease towards concrete scenarios (derived from the logical scenario by a selection of a concrete value from a parameter range) through logical scenarios (derive and represent requirements for the item during the system development phase) and inversely for a number of scenarios [Menzel, Bagschik, & Maurer, 2018].

The sensors are of the critical importance of every car with automated functions. They percept a scene and based on their transmitted information the car system makes a decision. Thus, with each upper level of automation, they are more responsible for positioning, ride, manoeuvres and thus safety and final reliability. Types of instruments used for perceiving the environment are shown in Figure 2.4.

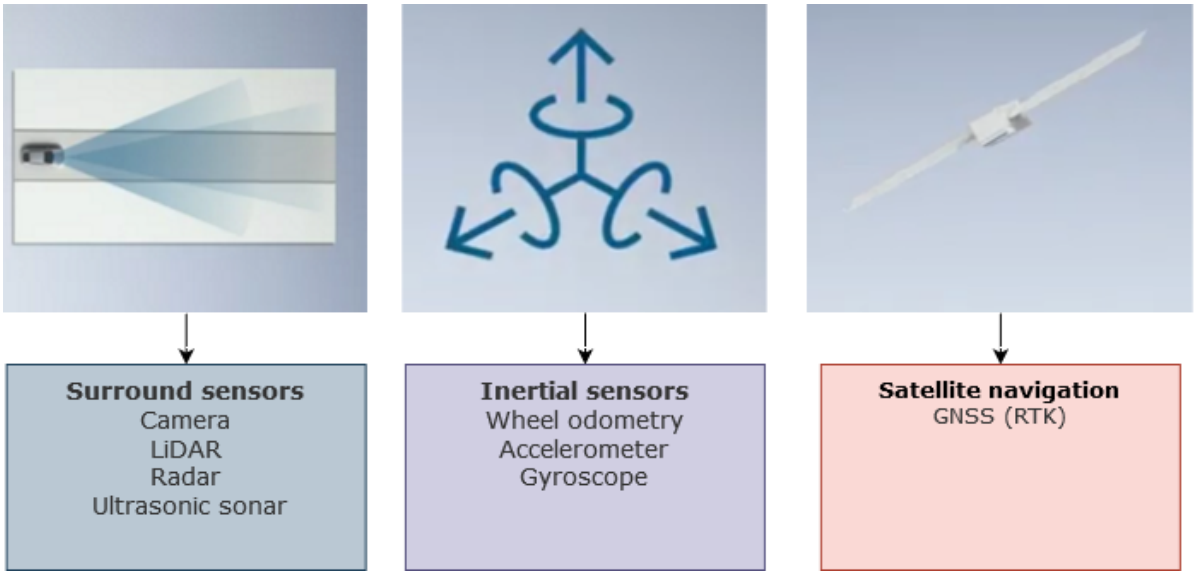

Figure 2.4.: Types of sensors used in an automated car [DPCcars, 2018; Wysocki, 2019]

Consequently, road space models should support sensor simulations. Nowadays, there are two main categories to pursue environment perception simulation as indicated in Figure 2.5.

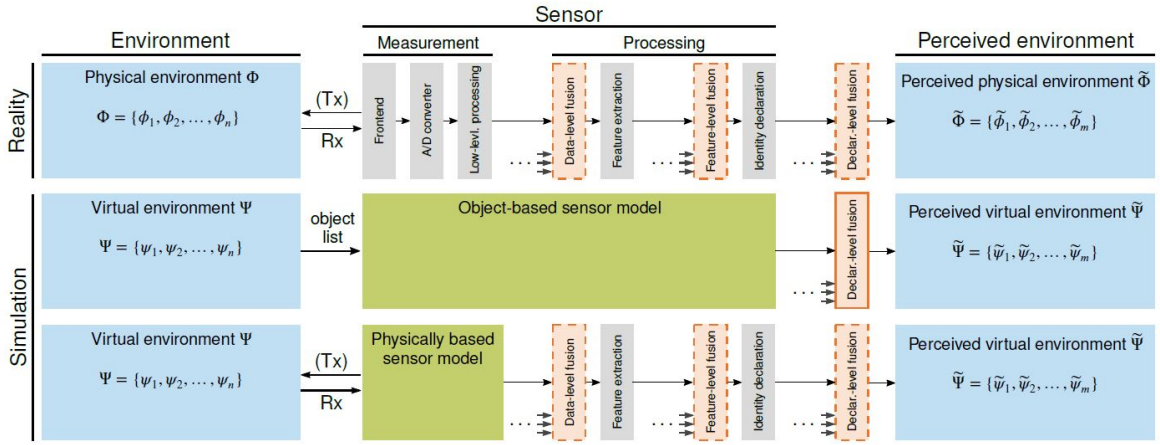

Figure 2.5.: Sensor based simulations [Schwab & Kolbe, 2019]

An object-based sensor model is a computationally inexpensive approach that skips parts of raw data fusion and focuses on an object list described by bounding boxes. This generalisation lets to discard irrelevant information like trash laying on the street and speeds up the process of final voting (declarative level fusion). Thus, perceived virtual environment objects are a subset of the virtual environment at the end of the process. In order to simulate possible sensors errors and delays, stochastic models are used e.g. Gaussian noise. A physically based sensor model method allows for such a generalisation but also for a fusion of raw data from

sensors. This slows down a process and does not allow for real-time processing however allows for better accuracy in terms of multipath propagation, wave propagation like scattering, reflection, or refraction which is not possible to test in the object-based environment [Schwab & Kolbe, 2019]. Nowadays, both methods are investigated by researchers. Also, both methods have advantages and disadvantages which will presumably lead to choosing an approach depending on current researchers' needs. Probably, to perform a certain test-case for which an object-based method will be sufficient there is no need to utilise physically based sensor model. Nevertheless, the latter approach promises more realistic results, especially in complex environments like urban scene [Hirsenkorn, Subkowski, Hanke, et al., 2017; Schwab & Kolbe, 2019]. In the end, there should be a standardized framework available [Hanke, van Driesten, Hirsenkorn, et al., 2017].

2.2. Data availability and modelling standards

However, there is an impediment in this approach – available data that will allow to fully utilise possibilities of the physically based approach. In order to simulate the interaction of the emitted sensor signal with environmental elements, object's roughness has to be known. Moreover, waves acts differently depending on the objects' materials properties. A glass surface reflects wave differently than a stone. There are models on the market of road spaces that have the potential to comply with those requirements.

The OpenDRIVE 1.6 road network model fulfils requirements of object-based sensor simulation as it allows for modelling of road space objects and can easily yield 3D bounding boxes to driving simulation. It also contains objects like lanes and it's borders, traffic signs and lights, or road-side objects. However, the format was not created for physically based sensor simulations and therefore it's capabilities are highly limited in this approach [Schwab & Kolbe, 2019]. On the other hand, the CityGML standard offers solutions that can directly aid this sensor simulation method. The classes like vegetation, buildings, traffic signs, and lights correspond to objects within a road space environment. Furthermore, a possibility to generically extend attributes of objects and adding information about objects' materials could be essential in simulations. The major challenge for this standard is modelling of such data which nowadays is mostly made manually. There are still limitations in an automatic generation to overcome [Beil & Kolbe, 2018]. Manual modelling is a time-consuming and cost-expensive task which is biased by a subjective depiction of reality by an expert. Furthermore, it will not assure quick and reliable customisation of a model to meet the requirements of sensor models. In case of changes in a scenario and components, the operator would have to manually prepare hundreds of slightly varying models which is cost-ineffective and in the end, excludes this standard to utilise it in automated driving functions tests. Within the scope of this Master's Thesis, attempt to fill the gap between developing of highly detailed model manually and the needs of geometric representation for automated driving simulations will be made.

2.2.1. OpenDRIVE

The OpendDRIVE is a modelling standard created for driving simulations. The OpenDRIVE standard has been in the market since 2006. The project started with the initiative of VIRES Simulationstechnologie GmbH, and since 2018 has been managed by Association for Advancement of international Standardization of Automation and Measuring Systems e.V (ASAM). It is an open format. Nowadays, the OpenDRIVE 1.6 serves also as a reference map and can be referred to the new term for maps for automated cars - HD Map [VIRES Simulationstechnologie GmbH, 2020b]. It is widely used by leading automotive companies, driving simulators vendors (products like Virtual Test Drive, CarMaker, Dyna4 or non-vendor CARLA) as well as scientific bodies [VIRES Simulationstechnologie GmbH, 2020a]. According to the Technical University of Munich, OpenDRIVE standard *facilitates an efficient exchange of data between universities and industrial partners* [VIRES Simulationstechnologie GmbH, 2020b]. The format is based on XML. Furthermore, the road geometry is described analytically which avoids a trap of bottleneck - does not limit further processing like discretisation.

OpenDRIVE 1.6 datasets could be geolocated and utilise Coordinate Reference System (CRS). However, every object has a location relative to the reference line (see Figure 2.6) similarly like in the concept of Linear Referencing System (LRS) - possible also to realize by most of the GIS tools [Esri, 2020; MPA Solutions, Servizio Gestione Strade, & Ufficio Controllo e Tecnologie Stradali, 2020; Safe Software, 2020a].

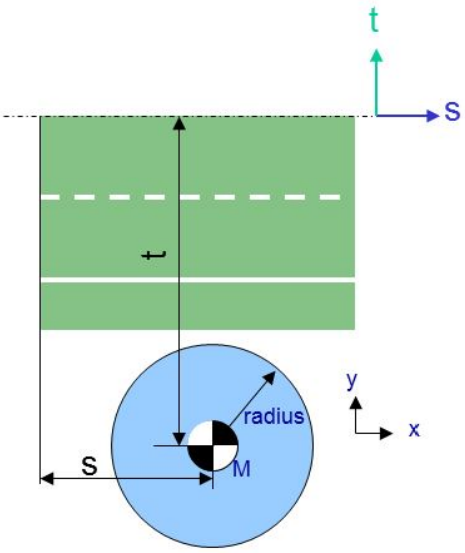

Figure 2.6.: The linear referencing in OpenDRIVE 1.6, example represents tree location definition. Each object is related to a road axis and therefore its position can be globally localized through localization of road axis [VIRES Simulationstechnologie GmbH, 2019]

The XML instance can inherit various road space items like [VIRES Simulationstechnologie GmbH, 2019]:

- Plane elements, lateral / vertical profile, lane width etc.
- Various types of lanes
- Junctions and junction groups
- Logical inter-connection of lanes
- Signs and signals
- Signal controllers (e.g. for junctions)
- Road surface properties
- Road-side objects
- User-definable data beads

OpenDRIVE database can be produced out of MLS point cloud but also other measurements techniques like in Figure 2.7 is shown.

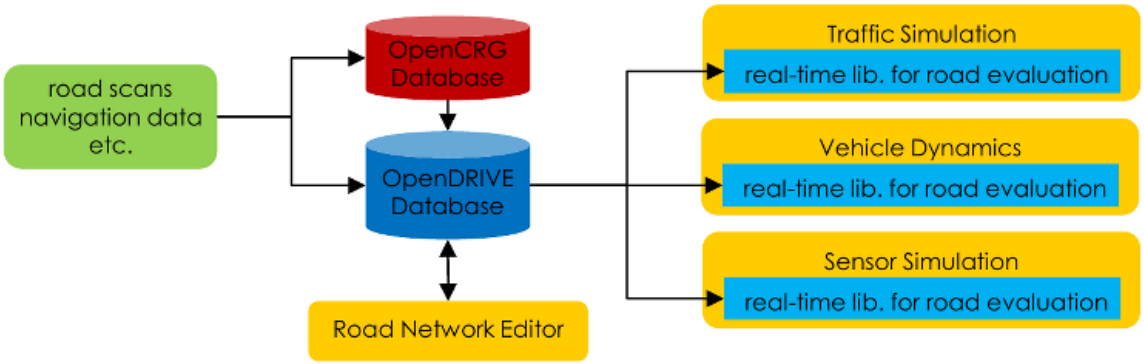

Figure 2.7.: Sources of OpenDRIVE datasets and their applications [VIRES Simulationstechnologie GmbH, 2020b]

Furthermore, the OpenDRIVE database serves as a core module in simulators like Virtual Test Drive (VTD) which is used in automated driving systems tests (object-based sensor model) [VIRES Simulationstechnologie GmbH, 2020c]. The OpenDRIVE 1.6 standard is suited to object-based sensor models and as for this purpose, it is sufficient. Geometries of road-side objects are simplified and are represented as cylinders, cuboids [VIRES Simulationstechnologie GmbH, 2019]. An example can be seen in Figure 2.8 where street lights and traffic signs are visualised.

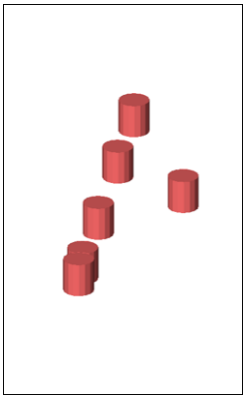

Figure 2.8.: A geometric representation of street lights and traffic signs in OpenDRIVE 1.6

Due to its limitations, the chances of using OpenDRIVE directly as a basis for physically based sensor models for testing the automated driving system are low [Schwab & Kolbe, 2019]. OpenDRIVE inherits valuable information about essential road assets like lane topology. Nevertheless, the lack of interoperability (common service communication, geometry model) and thus inconsistency between application-specific standards (e.g. CityGML) or proprietary standards (e.g. FBX) stands as an impediment in further development. Therefore, current studies focus on automatic parsing and translation of OpenDRIVE to other formats like Lanelet [Althoff, Urban, & Koschi, 2018] or within GIS tools [Scholz, 2019].

2.2.2. CityGML 2.0

CityGML is an application schema for the Geography Markup Language version 3 (GML3) based on XML [Kolbe, 2009]. The CityGML standard is an open data format for the storage of city models but also landscapes [Fiutak, Marx, Willkomm, & Donaubauer, 2018]. The responsible body is Open Geospatial Consortium (OGC) which releases official encoding standards [Open Geospatial Consortium, 2020a]. The representation of objects can start with Level of Detail 0 (LoD0) which allows for 2.5D representation of geometries (elevation and footprint) up to Level of Detail 4 (LoD4) where interiors and exteriors of objects (e.g. buildings) are possible to be modelled (see Figure 2.9). There are several modules introduced like Bridge, Building, CityFurniture, CityObjectGroup, Generics, LandUse, Relief, Transportation, Tunnel, Vegetation, WaterBody [Gröger, Kolbe, Nagel, & Häfele, 2012].

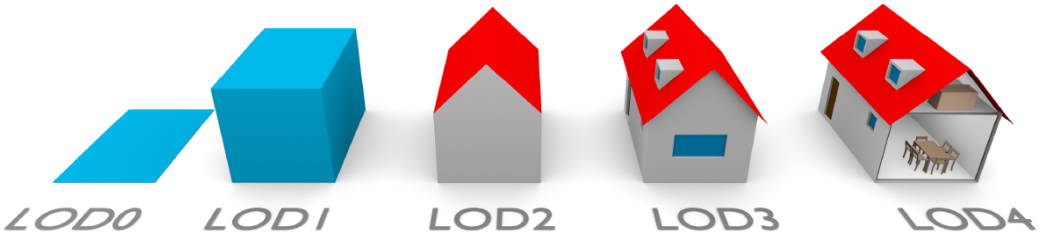

Figure 2.9.: Different Levels of Details in the CityGML standard [Biljecki, Stoter, Ledoux, et al., 2015]

CityGML is widely used by cities to efficiently manage their databases and deliver 3D visualisation of urban areas [Open Geospatial Consortium, 2020a]. Most applications are focused on buildings and different LoDs of those. Level of Detail 1 (LoD1) and LoD2 are a standard in most big municipalities like Berlin, Helsinki, Vienna, New York [Angermann, Donaubauer, Graw, et al., 2019] and smaller cities like Poznan in Poland [Geopoz, SHH, & virtualcitySYSTEMS, 2020]. However, LoD3 objects are barely available in web-portals. If they exist, they mostly cover specific landmarks within the city but do not serve as a base of 3D GIS of a municipality. This is due to the time-consuming process of developing of LoD3 objects which are mostly created manually [Angermann, Donaubauer, Graw, et al., 2019]. Lower LoDs are generated mostly automatically or with little manual interference based on ALS or aerial imageries which assure absolute 3D point accuracy of up to 1-2 meters [Angermann, Donaubauer, Graw, et al., 2019] which is sufficient for LoD2, LoD1 and LoD0 generation [Gröger, Kolbe, Nagel, & Häfele, 2012]. It is also possible to introduce other data e.g. from Unmanned Aerial Vehicle (UAV) or auxiliary information from cadastre. However, most of them do not fulfil suggested requirements mentioned in CityGML Encoding Standard which demands at least 0.5 m absolute accuracy to deliver LoD3 objects [Gröger, Kolbe, Nagel, & Häfele, 2012]. The full list of modelling accuracy suggestions is shown in Figure 2.10. Moreover, LoD2 buildings are often texturized to increase the attractivity of visual representation in web portals. These textures are mostly acquired from aerial oblique images and are automatically draped on semantically split buildings components [virtualcitySYSTEMS GmbH, 2018].

	LOD0	LOD1	LOD2	LOD3	LOD4
Model scale description	regional, landscape	city, region	city, city districts, projects	city districts, architectural models (exterior), landmark	architectural models (interior), landmark
Class of accuracy	lowest	low	middle	high	very high
Absolute 3D point accuracy (position / height)	lower than LOD1	5/5m	2/2m	0.5/0.5m	0.2/0.2m
Generalisation	maximal generalisation	object blocks as generalised features; > 6*6m/3m	objects as generalised features; > 4*4m/2m	object as real features; > 2*2m/1m	constructive elements and openings are represented
Building installations	no	no	yes	representative exterior features	real object form
Roof structure/representation	yes	flat	differentiated roof structures	real object form	real object form
Roof overhanging parts	yes	no	yes, if known	yes	yes
CityFurniture	no	important objects	prototypes, generalized objects	real object form	real object form
SolitaryVegetationObject	no	important objects	prototypes, higher 6m	prototypes, higher 2m	prototypes, real object form
PlantCover	no	>50*50m	>5*5m	< LOD2	<LOD2
…to be continued for the other feature themes					

Figure 2.10.: Proposed absolute accuracy requirements of CityGML models by OGC [Gröger, Kolbe, Nagel, & Häfele, 2012]

Within the CityGML standard road and city furniture modelling is also possible [Beil & Kolbe, 2018]. The proposed division of levels of details is shown in Figure 2.11. However, similarly to LoD3 buildings, it's a time-consuming task and therefore it does not stand as standard for 3D city or landscape models [Angermann, Donaubauer, Graw, et al., 2019]. Each vector object e.g. road or building can be globally localized using coordinate systems independently.

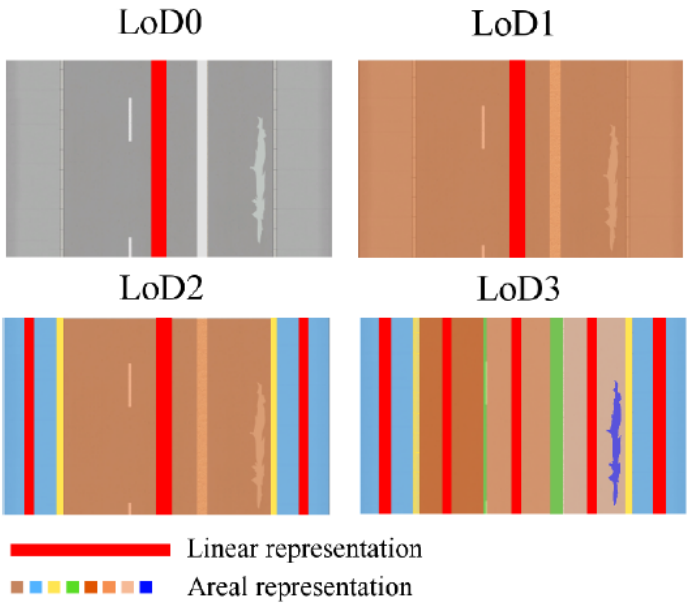

Figure 2.11.: A proposed representation of a road in the CityGML standard [Beil & Kolbe, 2017]

2.2.3. CityGML 3.0

At the beginning of 2020, Kutzner et al. gave an overview of the new, revised version of CityGML standard - CityGML 3.0 [Kutzner, Chaturvedi, & Kolbe, 2020]. The official CityGML 3.0 version will be released in the foreseeable future [Kutzner, Chaturvedi, & Kolbe, 2020]. However, the revised version of the conceptual Unified Modeling Language (UML) model, the Geography Markup Language (GML) encoding (XML schemas) are freely accessible [Open Geospatial Consortium, 2020b].

The new version of the OGC standard rebuilds to cater to new demands of applications. The overview of revised modules is shown in Figure 2.12. The most important changes in thematic modules concerning this project were made in modules Building and Transportation. In general modules like the Versioning could be of much interest [Kutzner, Chaturvedi, & Kolbe, 2020].

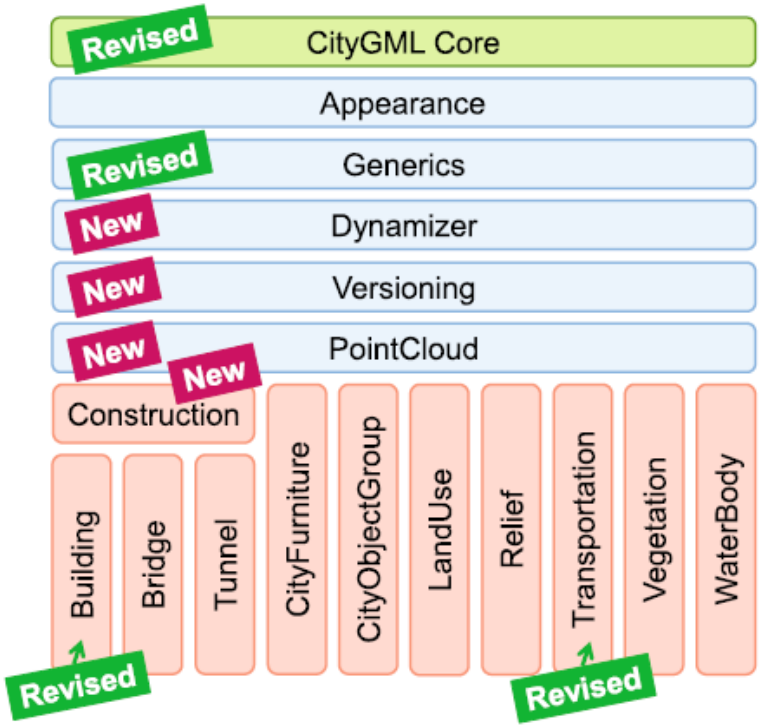

Figure 2.12.: An overview of the CityGML 3.0 standard. Pink boxes show varying thematic modules while blue and green ones specify general concepts [Kutzner, Chaturvedi, & Kolbe, 2020]

The Transportation module was revised to serve for applications in traffic and driving simulations fields, driving assistance systems and in the autonomous driving branch [Kutzner, Chaturvedi, & Kolbe, 2020]. The most important changes within this module like new possibilities to explicit modelling of manholes, holes, road markings can be highlighted.

Furthermore, the concept of Versioning can be the key to the refinements of geometries and semantics in the future. This module open possibilities for modelling of different city models versions. Therefore, the time factor can be introduced (see Figure 2.13).

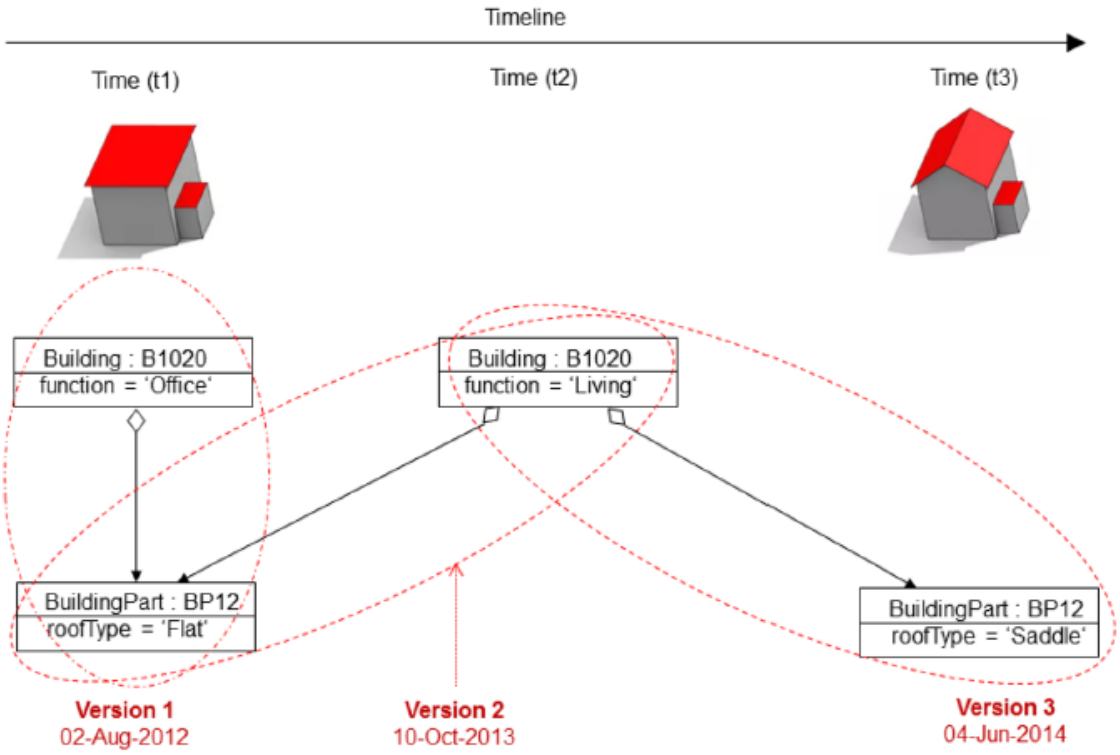

Figure 2.13.: Versions of buildings in the new CityGML 3.0 as an example of the Versioning concept [Kutzner, Chaturvedi, & Kolbe, 2020]

Also, the semantic concept of *space* and *space boundaries* is introduced in CityGML 3.0. Space Boundary depicts the borders of Spaces. The Space is an object which can be represented by 3D borders in the real world. The further subdivision can be made to *physical spaces* and *logical spaces*. Logical space is an abstract volumetric object. For example a city district or military zone. Physical space is a space that is bounded by a physical object and can be further subdivided into *occupied spaces* and *unoccupied spaces*. The space that is occupied means that a volumetric object already exists in this place and thus occupies this space. In turn, unoccupied space does not occupy any parts of the urban environment (see Figure 2.14).

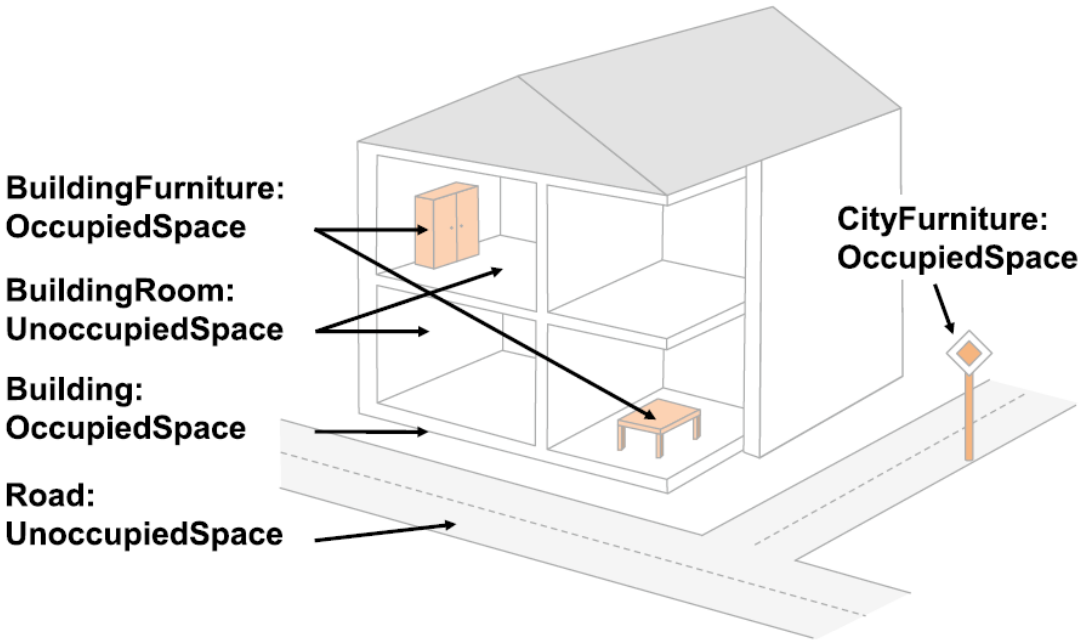

Figure 2.14.: Occupied and unoccupied space in the revised CityGML 3.0 standard [Kutzner, Chaturvedi, & Kolbe, 2020]

2.3. Point cloud processing

The point cloud is a set of unstructured points having representation in a 3D space (e.g. coordinate system) possibly with additional attributes like intensity or RGB value.

2.3.1. Segmentation

The segmentation of a point cloud means partitioning the point cloud into distinct clusters [Point Cloud Library, 2020]. Conventional segmentation methods utilise supervoxels and region growing algorithms [Papon, Abramov, Schoeler, & Worgotter, 2013; Point Cloud Library, 2020; Vo, Truong-Hong, Laefer, & Bertolotto, 2015]. Also, edge-based algorithms focuses on changes in local surface properties which mostly stands for normal, gradients, principal curvatures or higher derivatives where segmentation was based on threshold exceeding [Grilli, Menna, & Remondino, 2017]. Those methods are fast however results are not accurate. Region growing algorithm yields better results than the edge-based approach however seed points (initial values) are of much importance in this approach. The algorithm starts to grow a region in a seed point, then it expands based on a predefined criterion. It is essential to set a proper function based on which region should be expanded to achieve demanded results. Therefore, the operator needs pre-knowledge about a dataset [Grilli, Menna, & Remondino, 2017]. Other methods utilise K-means clustering or are based on hierarchical clustering [Grilli, Menna, & Remondino, 2017].

2.3.2. Machine Learning & Deep Learning in outliers filtering

Recently, research activity in the field of point cloud processing has been focused on the utilisation of deep learning and machine learning methods. This trend can be seen as a result of the availability of high computational resources and detailed datasets. Many deep learning works are focused on structured datasets like image sequences or speech recognition. However, point clouds represent in most cases an unordered set of vectors and thus not many solutions were released in this topic [Qi, Yi, Su, & Guibas, 2017]. First, deep learning classification techniques were focused on a transformation of datasets to 2D space and then performing classification. This simplification is also reflected in the shrinkage of possibilities - like scene understanding. However, the introduction of PointNet [Qi, Su, Mo, & Guibas, 2017] allowed for new studies in the field of 3D point cloud classification and segmentation (see Figure 2.15). It allows consuming unordered point sets as input. This novel approach allowed to further develop methods in the field like PointNet++ [Engelmann, Kontogianni, Hermans, & Leibe, 2017; Huang, Xu, Hong, et al., 2020; Zhou & Tuzel, 2017].

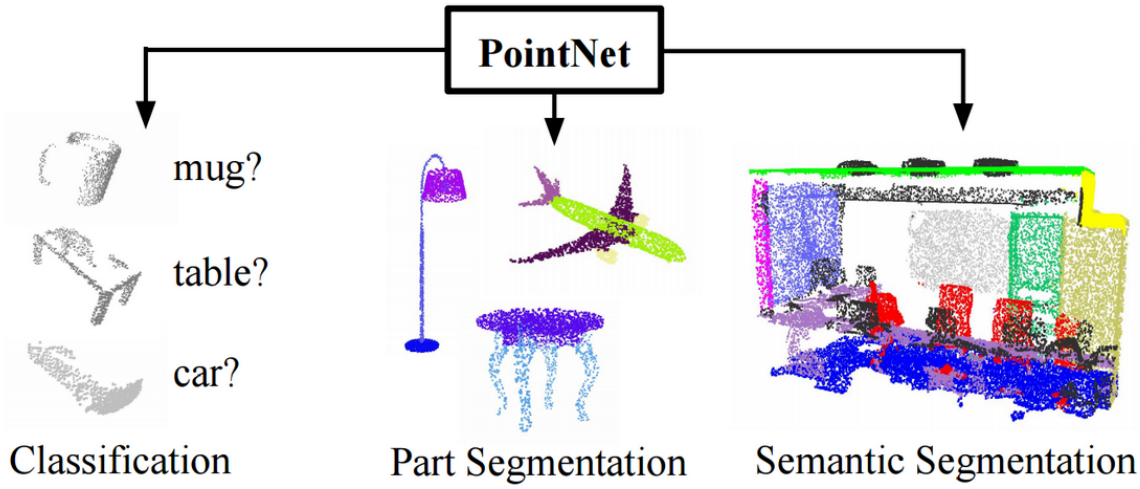

Figure 2.15.: PointNet overview [Qi, Yi, Su, & Guibas, 2017]

However, deep learning assumes to consume raw datasets and output the desired result [Griffiths & Boehm, 2019]. Thus, the solution does not expect to use a user's knowledge about a dataset nor already created models' semantics. Depending on the method's application it can be an advantage or disadvantage. For example, a model trained on buildings located in Asia will presumably not generalise well for buildings in Europe. Also, varying acquisition geometries and densities of training datasets stand as an impediment for wider usage of deep learning methods. On the other hand, algorithms like RANdom SAmple Consensus (RANSAC) utilise a user's knowledge in filtering out outliers [Ulrich, 2019].

RANSAC is an algorithm which searches for the best plane among a 3D point cloud. In parallel, it reduces the number of iterations [Yang & Forstner, 2010]. This algorithm has high robustness versus outliers (even for more than 50%) and the possibility to classify observations

to inliers and outliers. The objective of the algorithm is to fit a model n to measurement data S. Control parameters are t (distance threshold – correct classification of a large majority of inliers), T (minimum number of inliers), N (maximum number of selected minimum sets) [Ulrich, 2019]. The example of the RANSAC application is shown in Figure 2.16.

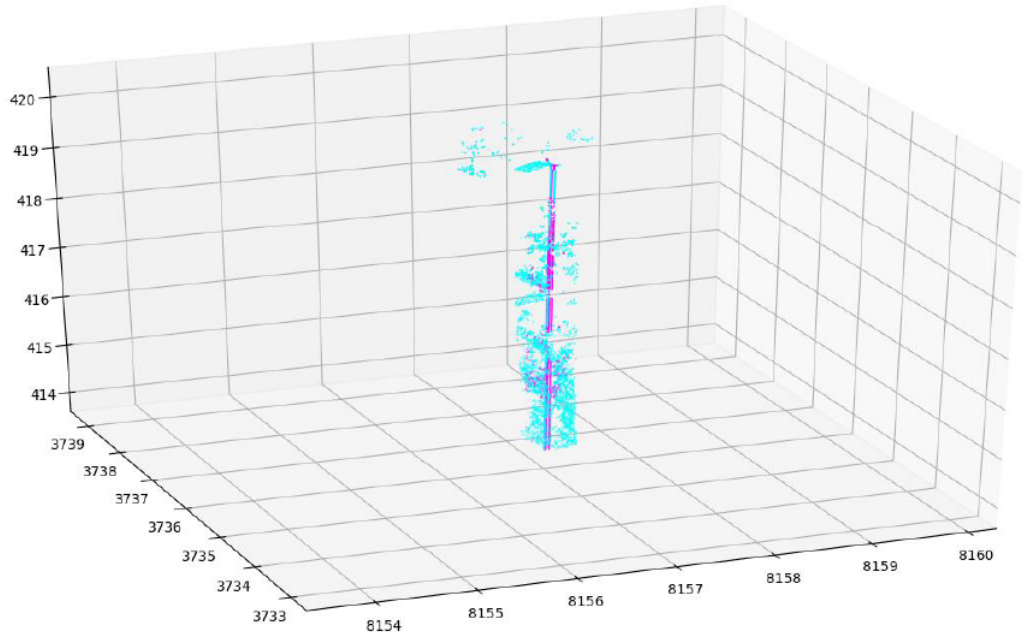

Figure 2.16.: RANSAC algorithm and detection of street lamp poles. Inliers (purple) and outliers (blue) [Wysocki & Albrecht, 2019]

2.3.3. Surface reconstruction algorithms

The reconstruction of 3D surfaces from point clouds samples is a well-researched problem in the Computer Vision field. It allows for processing of scanned data to mesh surfaces, to fill in holes of surfaces, to remesh existing models, and to stitch models [Kazhdan, Bolitho, & Hoppe, 2006].

There are several approaches to tackle the problem of reconstruction. One of those is a utilisation of the Poisson equation and its derivatives. The Poisson equation has been already used in several fields like tone mapping of high dynamic range images or seamless editing of image regions [Kazhdan, Bolitho, & Hoppe, 2006].

In the context of surface reconstruction the Screened-Poisson proved to be the state-of-the-art solution for 3D reconstruction problems [Kazhdan, Bolitho, & Hoppe, 2006; MeshLab, 2020].

The idea behind the Poisson, according to the creators Kazhdan et al., is to utilise *Poisson problem, which seeks the indicator function that best agrees with a set of noisy, non-uniform observations* [Kazhdan, Bolitho, & Hoppe, 2006]. The Poisson reconstruction perceives data globally. The whole provided dataset is processed without resorting to heuristic partitioning or blending [Kazhdan, Bolitho, & Hoppe, 2006]. It creates watertight surfaces from input point cloud subsets with a calculated orientation of those [Kazhdan & Hoppe, 2013].

The Screened-Poisson algorithm is a revised version of the Poisson surface reconstruction method. It was introduced by Kazhdan and Hoppe in 2013 [Kazhdan & Hoppe, 2013]. The method enhances the processing speed by complexity reduction of the algorithm. The *screening* is defined not over the whole domain but a sparse subset of points. Furthermore, thanks to improvements the algorithm should not over-smooth the input data.

The method is implemented by state-of the-art tools and software like The Computational Geometry Algorithms Library (CGAL) [CGAL, 2020b] or MeshLab [MeshLab, 2020]. The possible results of surface reconstruction are shown in Figure 2.17.

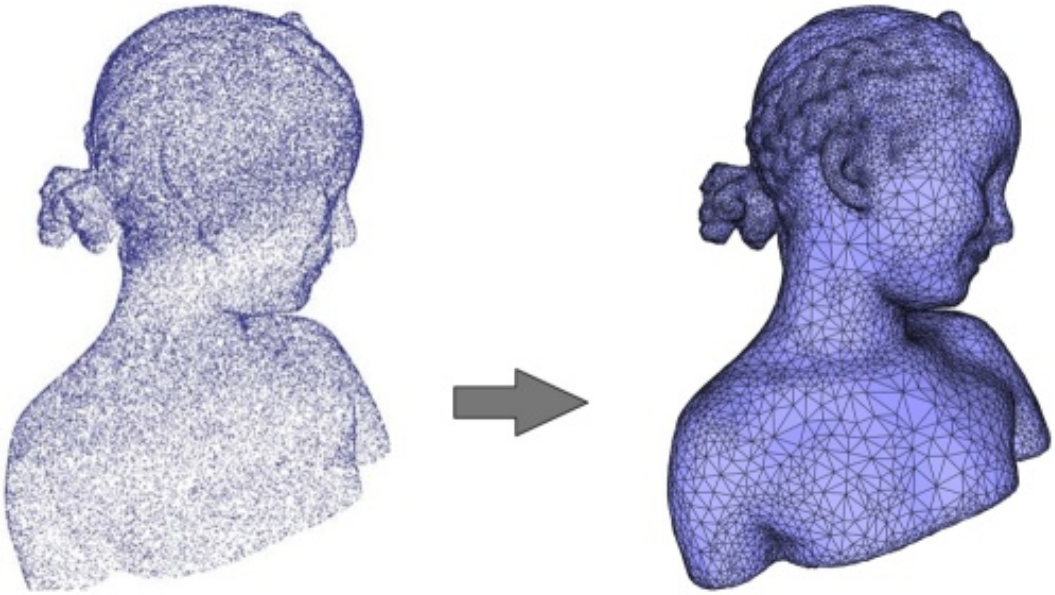

Figure 2.17.: Poisson reconstruction with 120K input points [CGAL, 2020a]

A common pipeline of the Poisson surface reconstruction starts with an analysis of an input point set. Then, the simplification and outliers removal is applied in order to allow for calculation of normals and final reconstruction. The last step is to provide a mesh structured model. In Figure 2.18 the overview of a common pipeline is shown.

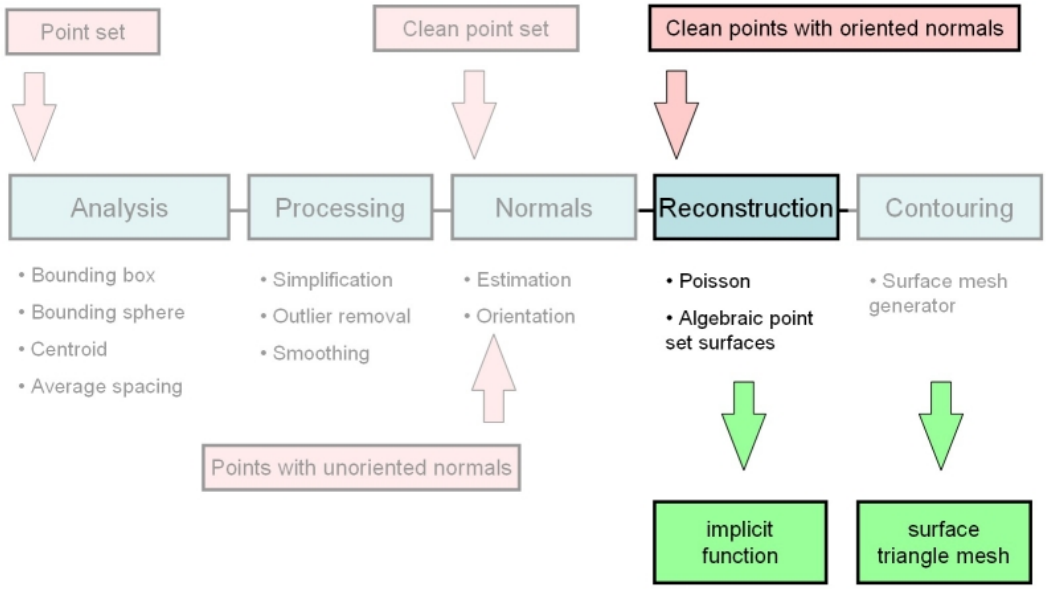

Figure 2.18.: Common pipeline to apply the Poisson reconstruction. Edited illustration of the CGAL's team [CGAL, 2020a]

2.4. Recent trends in the reconstruction of city models

The CityGML's support of coarse to very detail semantic and geometrical information encourages further applications development. Recently, the potential to utilise this standard in automated driving testing has been observed [Beil & Kolbe, 2018; Schwab & Kolbe, 2019]. However, the most important issue in the utilisation of highly detailed models (at least LoD3) is the manual work to produce details of objects and also validate them against a real scene. The increasing availability of highly accurate spatial data from MLS and ALS inspired researchers to create automatic workflows to reconstruct objects. In parallel, machine learning and deep learning branches noted very fast development in recent years. Moreover, algorithms to reconstruct surfaces are widely accessible as open-source libraries and also embedded in many tools [Angermann, Donaubauer, Graw, et al., 2019]. Those trends brought new light to LoD3 objects modelling and reconstruction of objects in general. In 2019, Wen et. al presented an approach of automatic reconstruction of a LoD3 building model using a fusion of different point cloud sources (ALS, Terrestrial Laser Scanning (TLS)) together with oblique imageries. Also, segmentation and plane extraction algorithms were used [Wen, Xie, Liu, & Yan, 2019]. The advantage of data acquired terrestrially over aerially is shown in Figure 2.19. It is also possible to use LoD2 building models with textures as a base for creating LoD3 buildings. This approach requires a Deep Learning architecture embedded [Hensel, Goebbels, & Kada, 2019]. The increasing availability of mesh models and free of charge LoD2 structures allowed to propose a workflow to integrate mesh models with semantic CityGML objects which resulted in accuracy improvements of solar potential analysis [Willenborg,

Pültz, & Kolbe, 2018]. Furthermore, Gruen et al. proposed and compared solutions of LoD3 semantics enrichments with an aid of LoD2 and without LoD2 as well as usage of procedural modelling [Gruen, Schubiger, Qin, et al., 2019]. The advantage of having MLS or TLS data is a possibility to recreate façade of buildings and depict details of roads effectively. Within the project VarCity, many researchers tried to find a solution for current city reconstruction problems [Swiss Federal Institute of Technology in Zurich, 2017]. One of the issues addressed by Bodis-Szomoru et al. was a fusion of complete but inaccurate airborne point cloud with detailed but incomplete street-side point cloud in order to create mesh representing city structures [Bodis-Szomoru, Riemenschneider, & Van Gool, 2016]. The other tackled issue was a complete reconstruction in 3D space based on point clouds which proved promising results [Martinovic, Knopp, Riemenschneider, & Van Gool, 2015]. Moreover, works towards image-based surface meshing approach to cover large city areas at the level of a CityGML LoD3 were conducted and showed promising results additionally allowing to swap images with 3D point clouds [Bodis-Szomoru, Riemenschneider, & Van Gool, 2015].

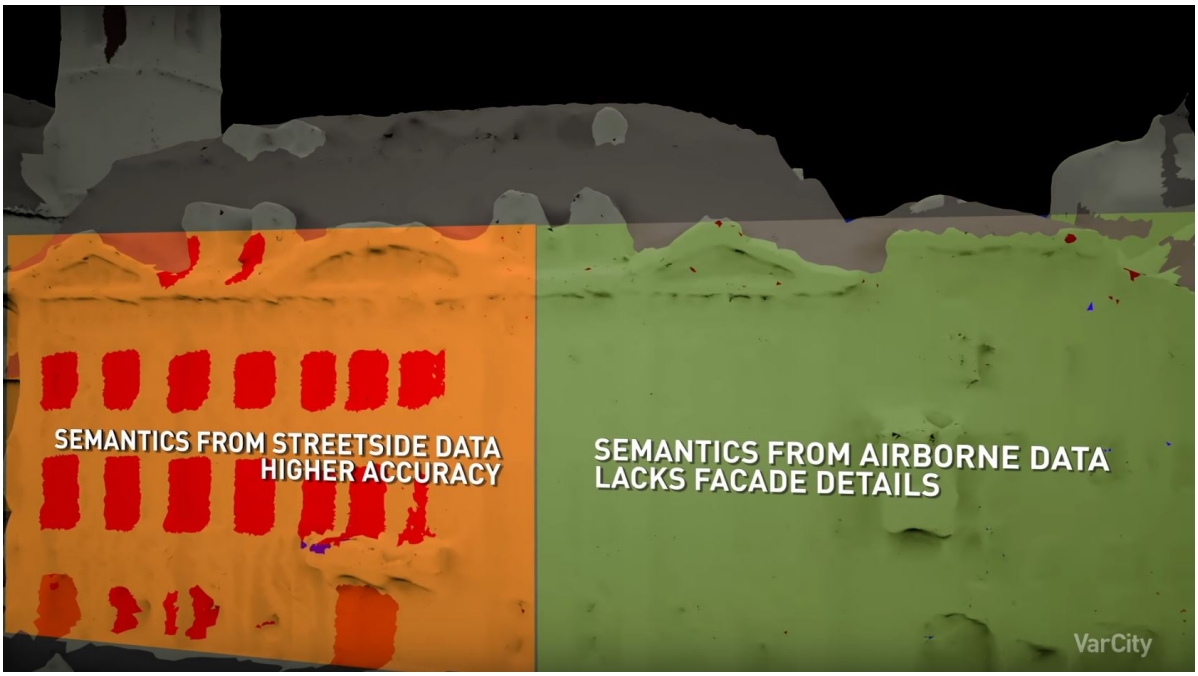

Figure 2.19.: Advantage of data acquired terrestrially over aerially – VarCity project [Swiss Federal Institute of Technology in Zurich, 2017]

2.5. Tools

The project aims to create a reproducible workflow. In order to do that several tools are gathered under an umbrella of one software. The software Feature Manipulation Engine (FME) caters the role of integration platform with direct links to LASTools, MeshLab Server, Python libraries (The ElementTree XML API, pyntcloud). Thus, it is possible to run the whole

processing part as an end-to-end solution just by specifying a few parameters. Moreover, FME, and thus the workflow, allows to write out data to specific data formats (e.g. CityGML) which can be consumed by UnrealEngine and 3DCityDB-Web-Map-Client.

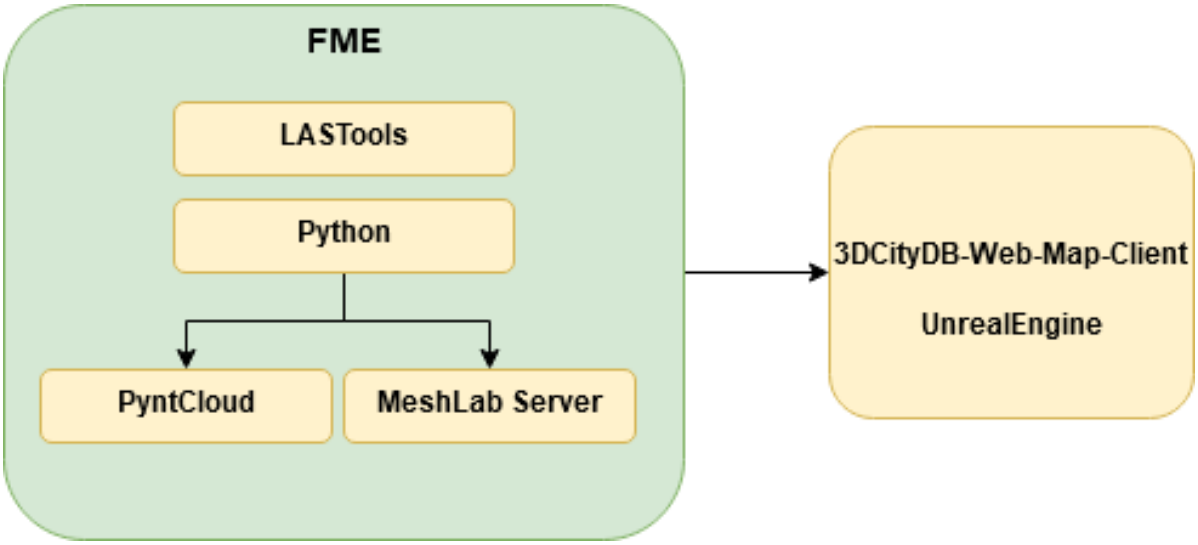

Figure 2.20.: Tools utilisation overview

2.5.1. FME

The Feature Manipulation Engine (FME) is a tool developed by Safe Software [Safe Software, 2020b]. It is a data integration platform which allows connecting more than 450 different applications by tailored templates and functions. Moreover, it is capable of the creation of reproducible workflows through FME Workspace files. More than 500 built-in functions can be used to manipulate data and perform complex spatial analysis. Functions are tailored to be used for not only GIS tasks but also Computer Vision, Machine Learning and others. Within a Workbench, it is possible to create custom functions and call external commands which enable the integration of other software, tools. Additionally to FME Workbench, the FME Inspector is shipped, which is a tool to explore data from files as well as intermediate products during the transformation process. FME is freely available for student purposes [Safe Software, 2020b].

2.5.2. LASTools

LASTools is the flagship product of the rapidlasso GmbH company. The software is easy to acquire, integrated into popular GIS software like ArcGIS, QGIS and FME. Moreover, it is also possible to download a standalone version which can be shipped with Graphical User Interface (GUI). Functions of software allow performing efficient point cloud processing like classification, segmentation, tiling, filtering etc. [rapidlasso GmbH, 2012].

LASTools have been integrated with FME software by Safe Software Lab and it is possible to seamlessly add LASTolls' functions to the workflow of the FME Workbench via FME Hub [Safe Software Lab, 2017].

2.5.3. MeshLab & MeshLab Server

MeshLab is open-source software for processing 3D meshes. The tool allows to edit, clean, generalise, translate, reconstruct, and explore meshes. Furthermore, it ships a function to consume raw point cloud data and perform a surface reconstruction. Furthermore, the software allows for preparing models for 3D printing [MeshLab, 2020].

MeshLab Server is a version of MeshLab which can perform operations in a batch mode. The selected set of operations to perform can be saved as an XML configuration file and triggered via command line [alemuntoni, 2020].

The main advantage of this software package is that one can perform testing of essential processing steps via MeshLab's GUI and easily export it as a configuration XML file for the MeshLab Server batch processing. Therefore, it is possible to seamlessly change certain parameters in the script file using e.g. programming languages like Python and its tailored XML parsing libraries. This, however, enables a convenient way to steer software performance and standard processes like reading, writing and processing.

2.5.4. Python, pyntcloud & ElementTree XML API libraries

Python is a modern programming language widely used by academia, industry, and IT enthusiasts. Python enables the development of web applications, GUI development, data processing and is utilised in many other fields [Python Software Foundation, 2020].

One of the reasons behind Python's popularity is the easiness of building extending libraries. One of those libraries is pyntcloud. This package allows working with point clouds within the Python environment. It offers a wide spectrum of possible reading and writing point cloud formats as well as functions for manipulation of those. Pyntcloud is composed of several modules offering functions like filtering, sampling, voxel creation and RANSAC algorithm [Castro, 2020].

This tool enables enhancements of the FME software which lacks point cloud processing possibilities in terms of machine learning algorithms. However, FME offers an option to integrate Python libraries within the software.

In order to parse XML data efficiently, the standard Python library is used - ElementTree XML API. This package allows to parse, manipulate and build XML files. It ensures the standard XML structure with roots and elements which allows to easily navigate and extract specific elements.

This package enables extending FME functionalities w.r.t. efficient XML parsing. The library can be integrated via native FME Python terminal and perform user-oriented changes in the script file of MeshLab Server.

2.5.5. Unreal Engine

Unreal Engine is an advanced real-time 3D creation tool. The software is developed by Epic Games and it is available without a cost and royalty-free under non-profit and for internal use license. It servers in fields like visualisation, entertainment, simulations. It has already gathered a large community of professionals and enthusiast of 3D graphics [Epic Games, 2020].

The Unreal Engine software caters the role of a backbone of automated driving simulators like CARLA [Dosovitskiy, Ros, Codevilla, et al., 2017; Lozé, 2019].

Many packages and plugins created by Epic Games and the community extend possibilities of Unreal Engine. For example, it is possible to import FBX, COLLADA, OBJ files directly to the working environment. However, the tool still does not support non-cartesian coordinate systems.

2.5.6. 3D City Database suite

The 3D City Database suite consists of three main parts: 3D City Database, 3D City Database Importer/Exporter (Importer/Exporter), 3DCityDB-Web-Map-Client. Those products are developed by the Chair of Geoinformatics, Technical University of Munich, virtualcitySYSTEMS GmbH and M.O.S.S. Computer Grafik Systeme GmbH. The suite is free of charge and open-source [Chair of Geoinformatics, Technical University of Munich, 2020].

The whole suite enables database storage of city models in CityGML 1.0 and 2.0 standard with its possible ADEs. It is compatible with Oracle as well as PostgreSQL/PostGIS solutions. The 3D City Database Importer/Exporter allows for seamless importing and exporting CityGML models to the database. Also, allows exporting files to formats like KML/COLLADA/glTF. Furthermore, the validation of CityGML syntax is possible. The 3DCityDB-Web-Map-Client tool can be perceived as CesiumJS 3D virtual globe extension that enables efficient exploration of large 3D models, shadow casting for 3D objects and terrain, layer management and sharing a web-map through an URL [Chair of Geoinformatics, Technical University of Munich, 2020; Yao, Nagel, Kunde, et al., 2018].

The solution is used in major cities in Europe and around the world like Berlin, Rotterdam, Zurich, Helsinki, and Singapore. [Chair of Geoinformatics, Technical University of Munich, 2020].

2.6. Datasets

The goal of this Master Thesis was to create refinements for automated vehicles functions testing, thus the data cover roads space objects.

Taking into consideration the broader application range of the solution, it is needed to evaluate the method for different types of available road space models. Therefore, the method can be applied for LoD1 objects of different types (vertical-like and horizontal-like). Those models shipped essential semantics to the workflow which enabled less computational

expensive processing and allowed for labelling of reconstructed geometries. As the basis for recreation of surfaces serves point cloud data acquired in ALS and MLS campaigns.

The case study area is located in the city centre of Ingolstadt, Bavaria, Germany and the extent of data encompasses a polygon of around 0.5 km x 0.5 km. All presented illustrations were made within this test scenario area if not stated differently.

2.6.1. City models

HD Map, LoD1

As HD Map comprises most relevant road space objects it serves as the basis for workflow creation. The 3D model was shipped as the OpenDRIVE standard. The dataset contains geometries like trees, street lamps, fences, buildings and roads which are located within a road space. In this work, only buildings and roads are taken into consideration as geometries to be refined. Thanks to existing OpenDRIVE to CityGML converter r:trån [Schwab, 2020; Schwab, Beil, & Kolbe, 2020] it was possible to load data directly into the GIS software in the CityGML standard.

The OpenDRIVE has coarse geometries with rather poor semantics, therefore, it was translated into CityGML LoD1.

HD Map consists of 87 buildings and 94 roads for the test scenario.

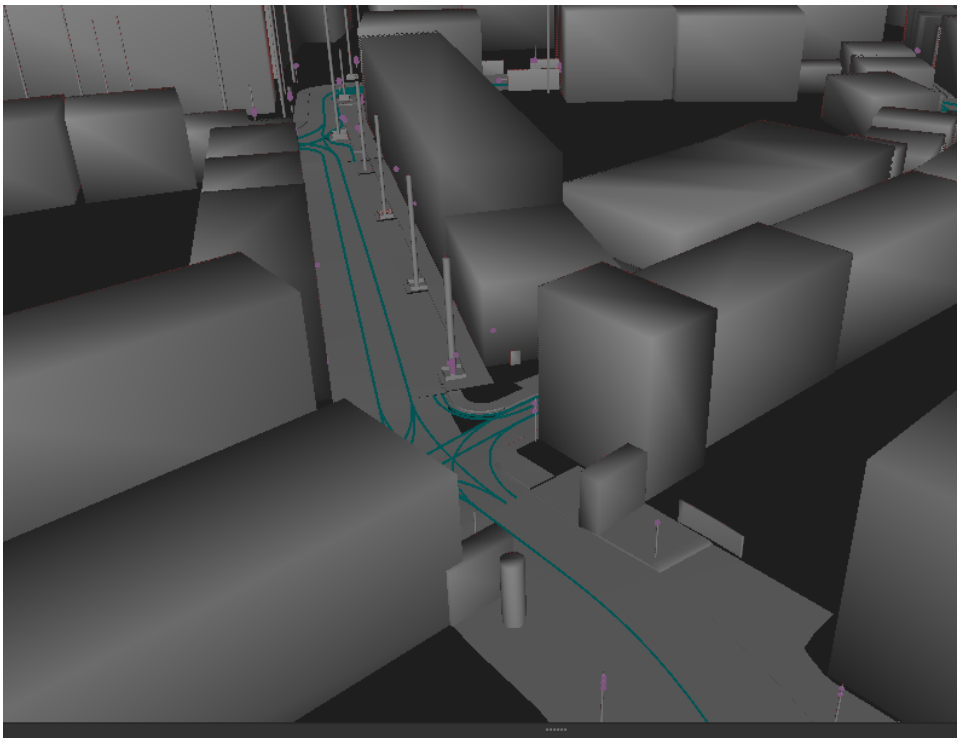

Figure 2.21.: OpenDRIVE objects (transformed into CityGML LoD1 models)

Governmental data, LoD2

Bavarian State Office for Survey and Geoinformation (ger. Landesamt für Digitalisierung, Breitband und Vermessung) (LDBV) office delivers LoD2 buildings models for the whole Bavaria state based on cadastre and aerial point cloud data [Landesamt für Digitalisierung, Breitband und Vermessung, 2020].

The 3D LoD2 buildings are used as comparison models for refined geometries. The main advantage of such models are modelled roof structures and footprints acquired from the national cadastre as well as buildings functions.

LoD2 city model consists of 599 buildings for the test scenario.

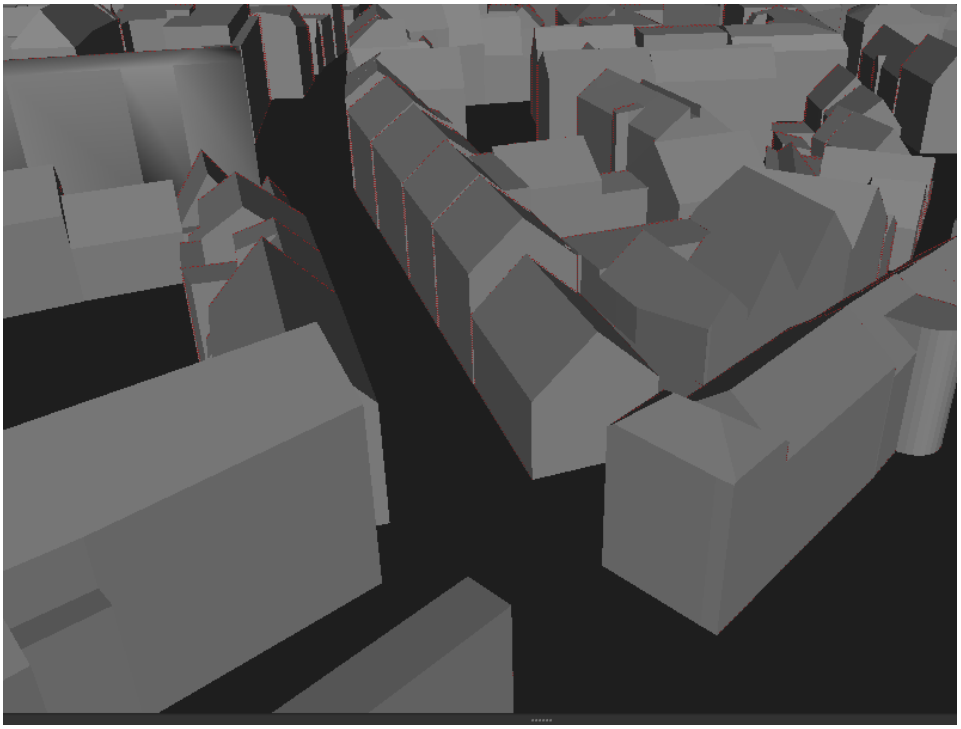

Figure 2.22.: LoD2 buildings

Manual modelling, LoD3

The manually modelled buildings were created based on MLS point clouds. Those buildings inherit spatial features like windows and doors. Also, they are the most detailed vector objects within the workflow. However, as they were manually modelled, some incorrectness in classification and modelling can occur. The 3D LoD3 buildings are used as comparison models for refined geometries.

LoD3 city model consists of 28 buildings for the test scenario.

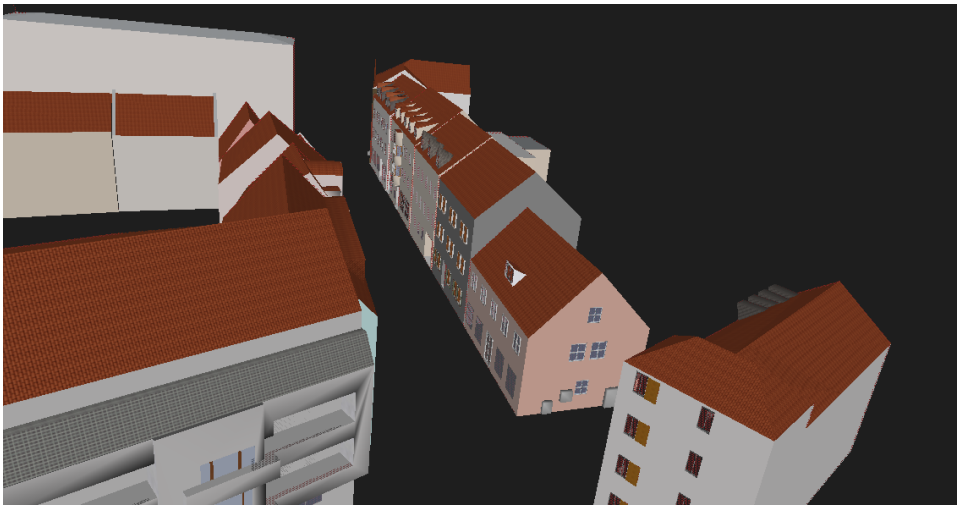

Figure 2.23.: Manually modeled LoD3 buildings

2.6.2. Point clouds

Point clouds utilised within this workflow can be divided based on the type of acquisition into ALS and MLS.

Airborne Laser Scanning

Point cloud obtained by ALS used within this projects is delivered by LDBV. Type of acquisition puts limitations to this point cloud in terms of coverage of vertical structures and objects covered by higher structures (e.g. trees covering pavements). Therefore, the measuring points not only fall on the surface of the ground, but also on the objects on them, e.g. trees or buildings. The point cloud has automatically classified points into [Landesamt für Digitalisierung, Breitband und Vermessung, 2020]:

- Ground points
- Object points
- Nonassignable points near the ground
- Building points

Those classes can be used as a hint while refining the objects' structures. However, due to the sparsity of point cloud, the classes were not utilised. Moreover, low coverage of vertical structures disabled this dataset to be useful in terms of vertical-like structures reconstruction. Only, in case of horizontal-like structures, this data complement MLS point clouds in areas where MLS field of view was obstructed (e.g. parked cars) [Landesamt für Digitalisierung, Breitband und Vermessung, 2020].

To sum up, this data should be perceived as a supportive source of information while having MLS point cloud data which should be then used as the main source of spatial information. Furthermore, the usage of this kind of data is helpful only in terms of horizontal-like objects.

It has to be underlined that the dataset is shipped in European Petroleum Survey Group (EPSG) 25832, DHHN2016 CRS which results in different height systems between datasets and thus in height deviations. For this particular area of Ingolstadt, a shift of 46.5 m reduces vastly the height differences. The small deviations are further minimised within the presented workflow.

The median density for an area of interest is around 6000 points per 25 m x 25 m tile.

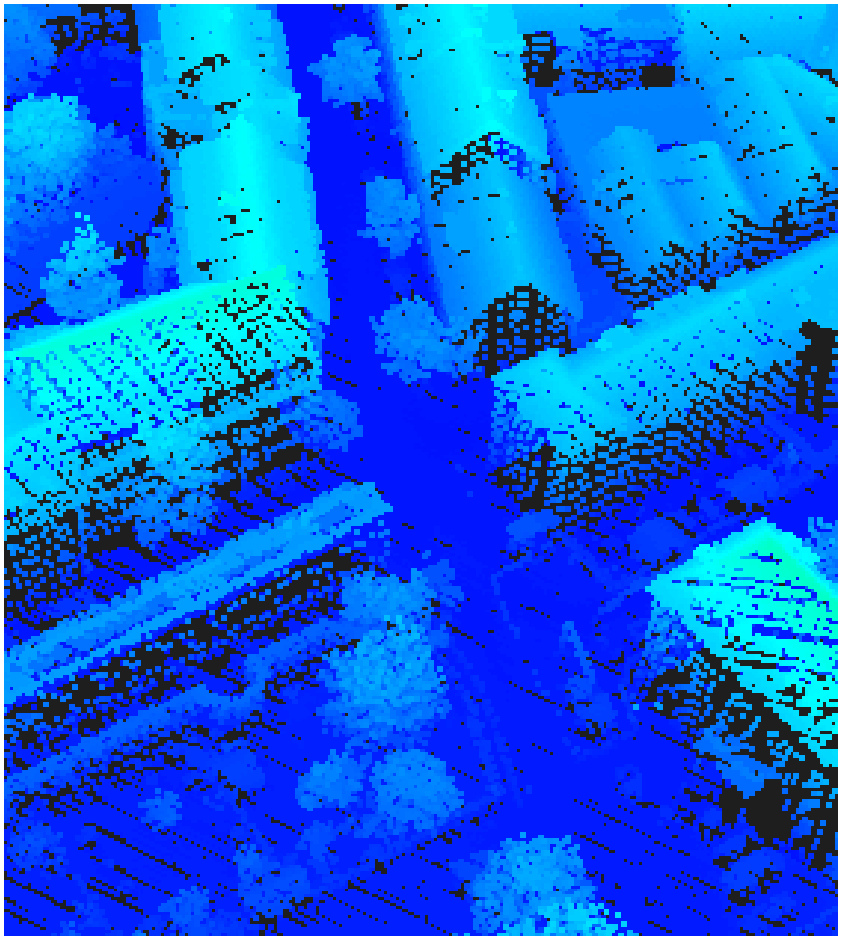

Figure 2.24.: ALS point cloud

Mobile Laser Scanning

Point cloud obtained by MLS utilised within this project is delivered by 3D Mapping Solutions GmbH [Coduro, 2018]. Dense point cloud subsets acquired at the street level cover structures that face main streets where the acquisition had taken place. This does

not necessarily mean that only façades are covered but also e.g. buildings' backyards and buildings' side walls. However, type of acquisition puts limitations to this type of point cloud in terms of coverage of elements located at the top of structures as well as objects which are covered by features located closer to the MLS acquisition location. For example, a tree on a pavement can obstruct a field of view of a scanner for a building located behind the tree.

The median density for an area of interest is around 91 500 points per 25 m x 25 m tile.

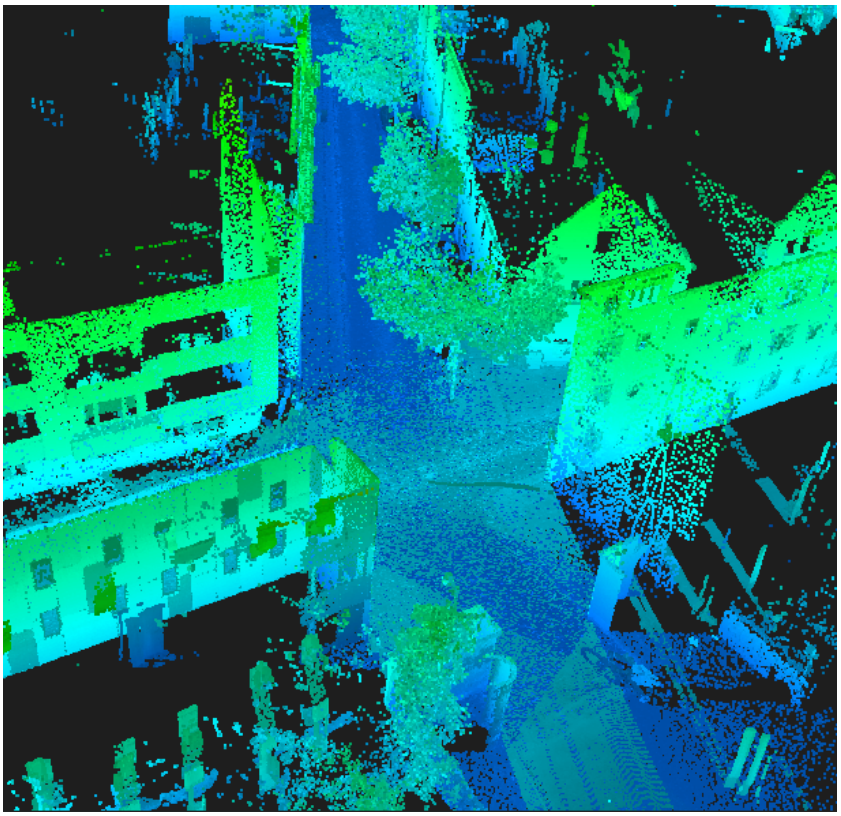

Figure 2.25.: MLS point cloud

3. Methodology

The workflow is designed to use highly dense point cloud data from laser scanning (MLS), aerial point cloud data (ALS) and CityGML data acquired from varying sources (manual modelling, governmental data, HD Map). The processing of data is a fusion of cutting edge solutions from Geoinformation (FME, CityGML) and Photogrammetry (LASTools, RANSAC) fields. The visualisation of results is performed in a web (3DCityDB-Web-Map-Client) and a game engine (Unreal Engine) and those stand as a benchmark for the workflow utilisation. An overview of the architecture design is shown in Figure 3.1.

The methodology is designed to allow for a reproducible application. The method should serve as an end-to-end solution for refinements of selected geometries for simulations of automated driving functions. Moreover, the workflow allows for user-oriented customisation by the introduction of parameters which are fine-tuned by default but also possible to be changed by an operator.

The method can be applied to vertical-like (e.g. buildings) and horizontal-like objects (e.g. roads). Those terms are explained in more detail in the following sections of this chapter. It is believed that after slight customisation the solution should perform decently for other road space objects.

All the sections are designed to be performed in the FME 2020.0 environment. Thus, all described steps are reproducible in FME version 2020.0. The tools that are integrated within the FME like LASTools should be compatible with the FME 2020.0 version and higher, scripts are designed for Python 3.6 and MeshLabServer is designed to work with 2020.03 release. The not FME-integrated tools are used for visualisation of results. Software versions like 3D City Database 4.0.1, the Importer/Exporter 4.2.0, the 3D Web Viewer 1.6.2, and Unreal Engine 4.24.3 are utilised.

The whole workflow is designed for urban areas. Preferable input datasets for processing are MLS, ALS and CityGML LoD1 objects. As the validation of results can serve CityGML LoD2, LoD3 objects, point clouds, meshes and other geodata sources. The MLS point should have density of around 90 000 points while ALS 6000 per 25 m x 25 m. However, this can be adjusted in the workflow.

The names of chapters are capitalised to indicate that it references to the specific section of the workflow, not the general term. For example, *Clipping* refers to described steps in this workflow while *clipping* to the operation of cutting of overlying layers known in GIS.

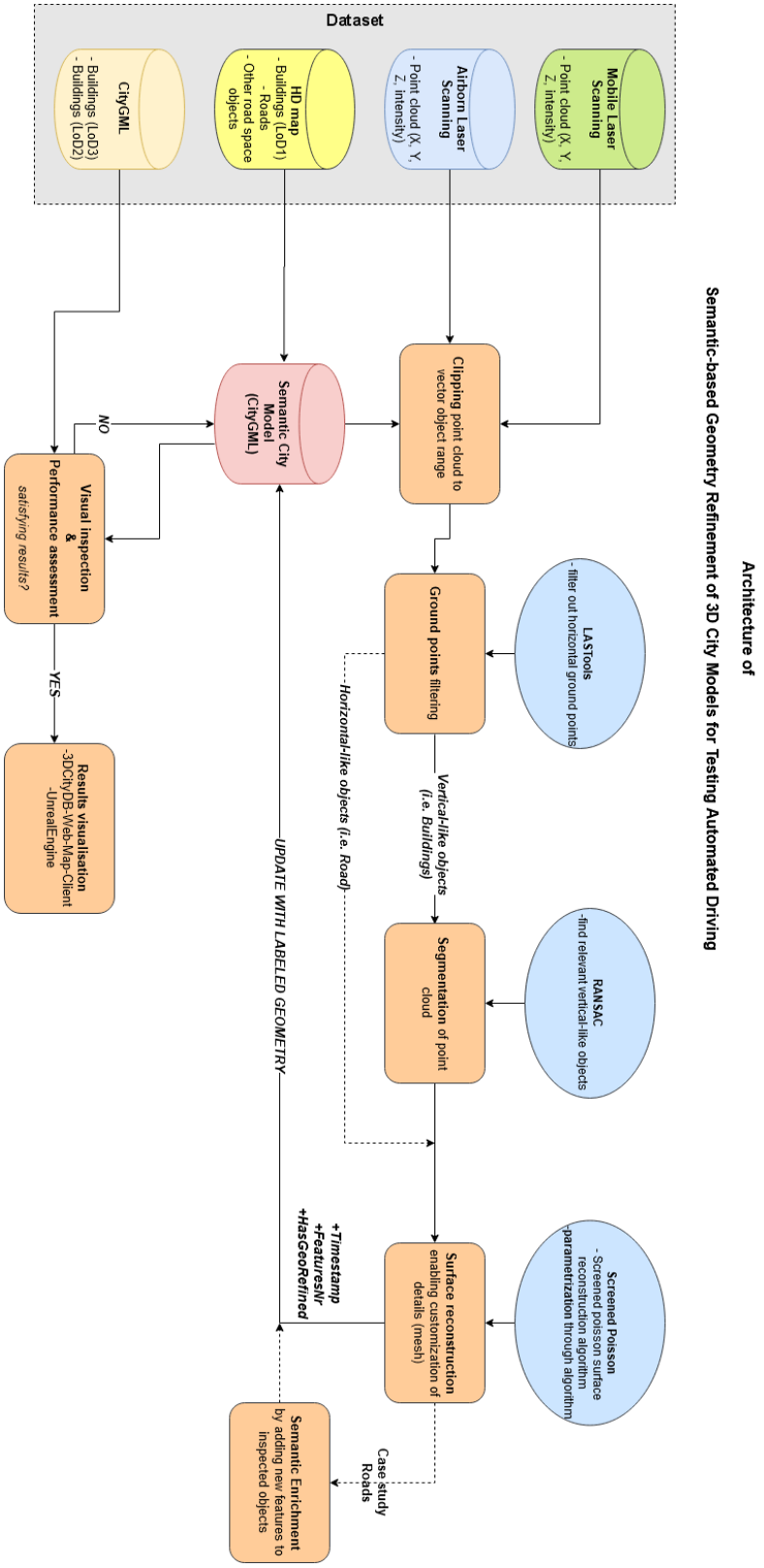

Figure 3.1.: An architecture overview

3.1. Clipping

Clipping and buffering are one of the most popular operations in the GIS field. Those methods can serve as starting points for further and more complex spatial analysis. The main advantage of those methods for this particular workflow is a high reduction of outliers with a low-cost computational operation. The buffering and the clipping should filter out not relevant parts of point clouds based on *a priori* information of absolute accuracy of the input vector city models.

The buffering and the clipping operation could be conducted in a 2D and a 3D space. The latter is introduced in this workflow as it is designed to process 3D objects. However, instead of a direct 3D buffer, a simplification is introduced. The standard 3D buffer creates a sphere-like object, this geometry, however, is computationally demanding for the clipper. In order to avoid that, a mixture of a 2D buffer and an extrusion is introduced. First, the 2D buffer is calculated and then the extrusion in positive and negative Z coordinate is calculated. Therefore, respective buffers perform in X, Y, Z direction (also Z positive and negative direction). In order to assure a buffer to be robust w.r.t. LoDs, the buffering always performs on oriented bounding boxes of buildings objects. Thus, e.g. LoD2 buildings buffers resemble LoD1 buildings. This allows simplifying buffering process which reflects in the faster computation of buffers and at the same time still decreases the number of outliers. The expected result of Clipping is shown in Figure 3.3 with a point cloud portion cut for a LoD1 object.

The project is designed to extensively utilise semantics of a city model. Thus, the buffer's range around each processing object varies depending on the type of an object (e.g. road, building) and its specific LoD (1,2,3). As the Clipping step should not erase inliers each value range is set according to the CityGML Encoding Standard as shown in Figure 3.2 [Gröger, Kolbe, Nagel, & Häfele, 2012]. This accuracy is possible to be changed within the FME Workbench in case of a known different accuracy of input models independent from LoD's suggested accuracy.

	LOD0	LOD1	LOD2	LOD3	LOD4
Model scale description	regional, landscape	city, region	city, city districts, projects	city districts, architectural models (exterior), landmark	architectural models (interior), landmark
Class of accuracy	lowest	low	middle	high	very high
Absolute 3D point accuracy (position / height)	lower than LOD1	5/5m	2/2m	0.5/0.5m	0.2/0.2m
Generalisation	maximal generalisation	object blocks as generalised features; > 6*6m/3m	objects as generalised features; > 4*4m/2m	object as real features; > 2*2m/1m	constructive elements and openings are represented
Building installations	no	no	yes	representative exterior features	real object form
Roof structure/representation	yes	flat	differentiated roof structures	real object form	real object form
Roof overhanging parts	yes	no	yes, if known	yes	yes
CityFurniture	no	important objects	prototypes, generalized objects	real object form	real object form
SolitaryVegetationObject	no	important objects	prototypes, higher 6m	prototypes, higher 2m	prototypes, real object form
PlantCover	no	>50*50m	>5*5m	< LOD2	<LOD2
…to be continued for the other feature themes					

Figure 3.2.: Guidelines for modelling specific LoDs [Gröger, Kolbe, Nagel, & Häfele, 2012]

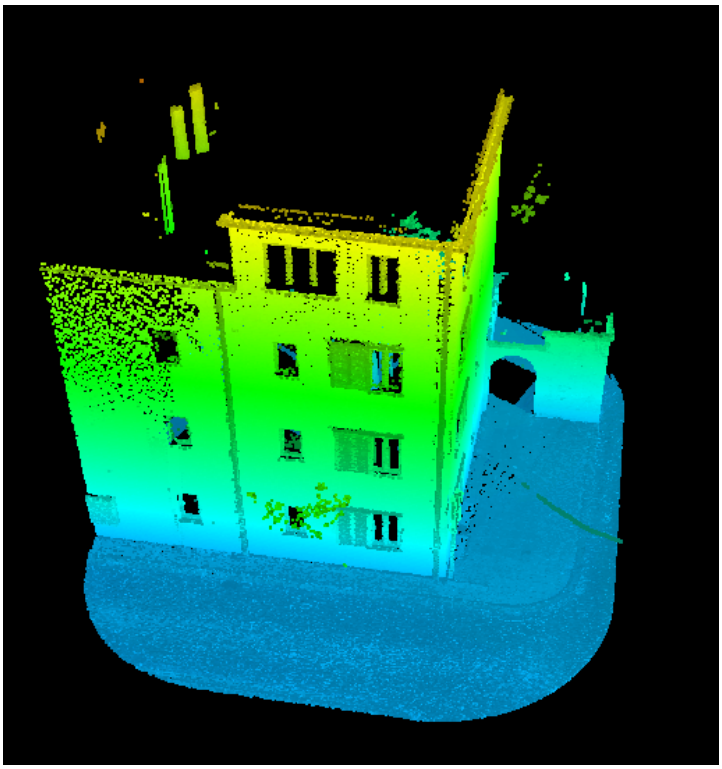

Figure 3.3.: The portion of point cloud cut by custom 3D buffer for a LoD1 building

3.2. Ground Points Filtering

The usage of semantic information included within vector objects allows to shrink areas of interests to a specific 3D ranges and thus limits the number of possible points in point clouds depicting specific object type. However, the method does not filter out all outliers.

At this stage, point clouds could be perceived as a depiction of horizontal-like and vertical-like objects and/or parts of those. For example, a road segment could be described as a horizontal-like object as it consists of horizontal-like parts. On the other hand, a building is a vertical-like object as it is a vertically extruded construction consisting of horizontal-like (in case of flat roofs) and vertical-like parts. However, as roads and buildings do not have an ideally plane surface either in the horizontal or vertical direction, the used terms have a suffix *-like*. Those terms are used in the next parts of this work.

This assumption allows pursuing filtering of horizontally aligned points in point clouds within a specific objects' range. Moreover, in the case of horizontal-like objects, it enables a way to filter out all non-relevant points and as a consequence directly yields data to the Surface Reconstruction step (see Figure 3.4). For the vertical-like objects, filtering algorithms are applied within the Segmentation step and due to the Ground Points Filtering the amount of those points is decreased.

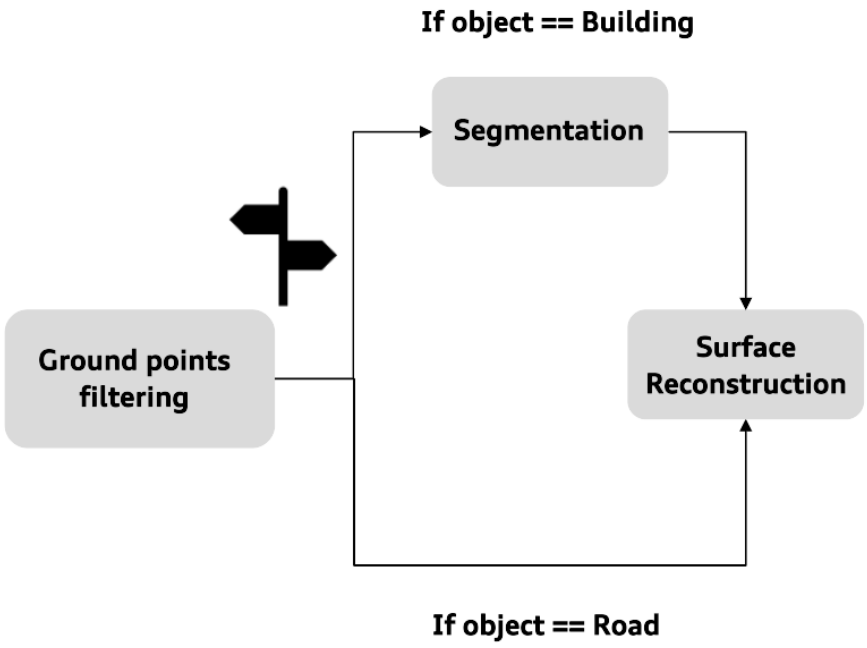

Figure 3.4.: Schema of the Ground Points Filtering step, the case studies building (vertical-like object) and road (horizontal-like object)

3.2.1. Horizontal-like objects

The LASTools.lasground [Safe Software Lab, 2017] tool is capable of segment points to ground points and non-ground points. This function is shipped as a custom transformer for FME and can be acquired via FME Hub. The LASTools.lasground tool is the *de facto* standard for the processing of point clouds. However, it is not possible to explain in detail the working principles of the algorithm as it is a closed standard. Settings are tuned to fit the needs of a type and a place of a point cloud acquisition. Therefore, flags indicating not an airborne point cloud and an urban environment are turned on [Isenburg, 2020]. This setting is selected for horizontal-like objects with combined point clouds both from ALS and MLS (see an example in Figure 3.5). This is possible because features like roads tend to have a smooth surface with no significant spikes and no more than one horizontal surface per feature.

Results obtained for those features show that almost all outliers are successfully filtered out (see Figure 3.6). This allows forwarding points of point cloud classified as ground points to the next step - Surface Reconstruction (see Figure 3.4).

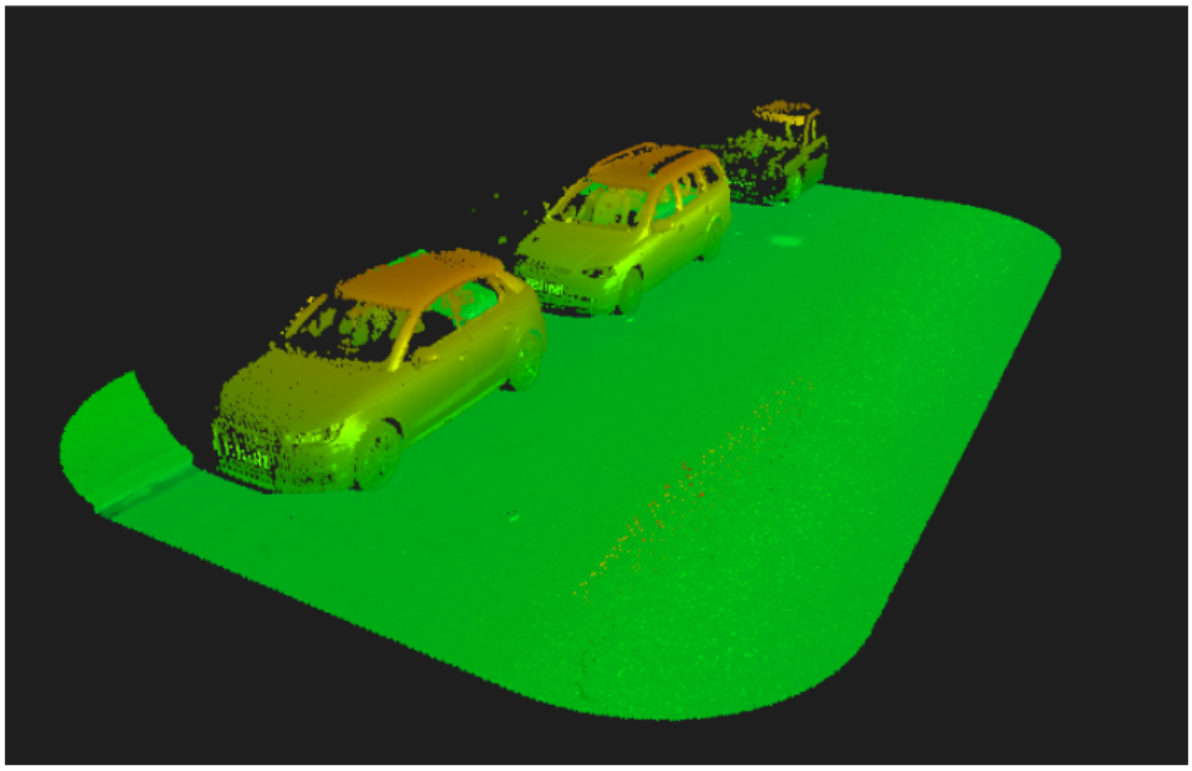

Figure 3.5.: Point cloud (ALS & MLS) cut to an area of interests of a road segment, before the Ground Points Filtering

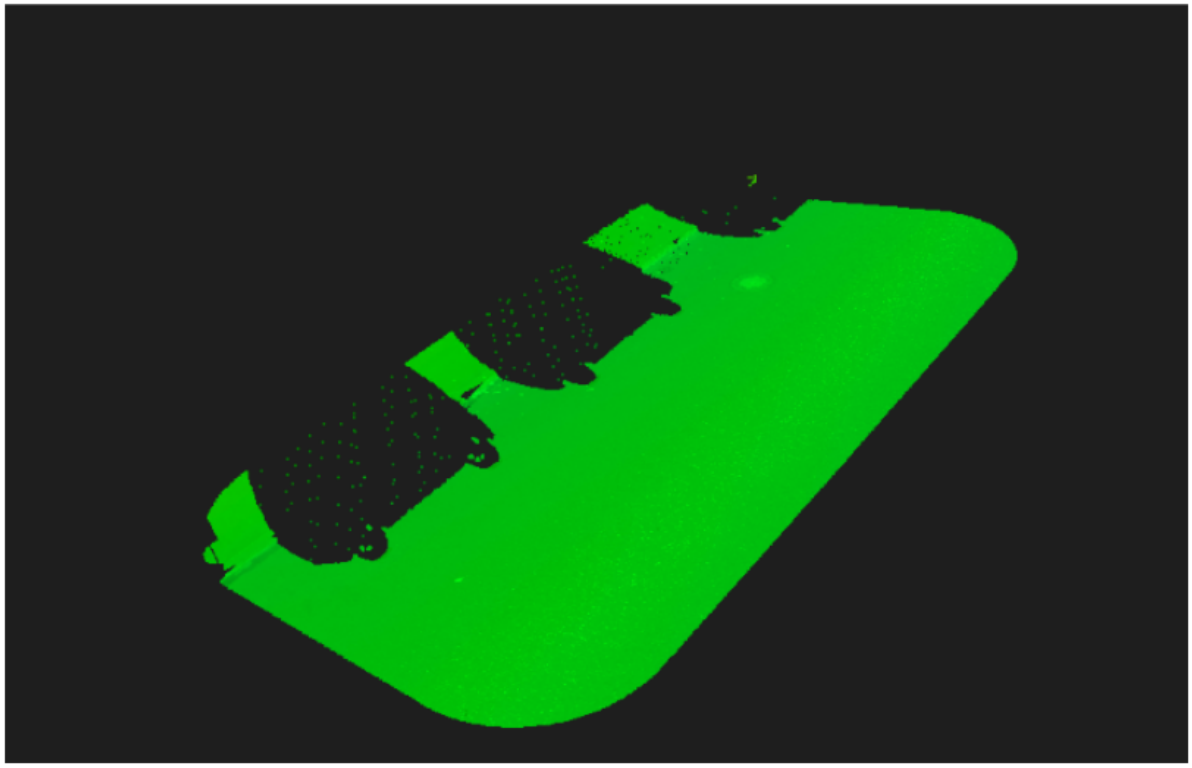

Figure 3.6.: Point cloud (ALS & MLS) cut to an area of interests of a road segment, after the Ground Points Filtering

One of the challenges with a combination of ALS and MLS data are differences in absolute heights which can occur due to height reference differences or registration errors. Hence, the step of height rectification is introduced into the general workflow.

The general shift calculated for the whole area can be sufficient for the whole investigated area to perform the Ground Points Filtering. Nonetheless, small (up to 0.5 m for areas at a scale of a city district) discrepancies might still exist. In order to rectify them, the second shift is performed, this time locally. The dense MLS point cloud serves as a reference for the sparse ALS point cloud within a specific feature extent. Therefore, the ALS data is shifted towards a calculated mode height of the specific MLS point cloud (see Figure 3.7). This step is repeated per each analysed horizontal-like feature.

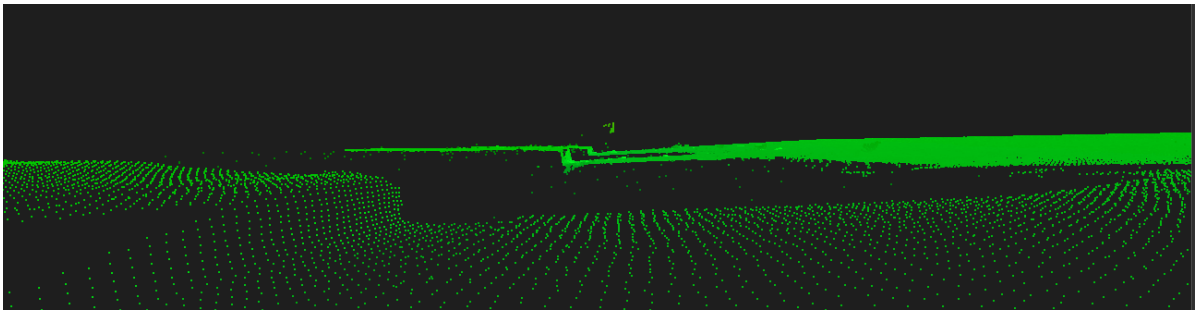

Figure 3.7.: One of *holes* in a MLS point cloud filled in with ALS points with a rectified height

The ALS dataset has a classification attribute which indicates what object is described by a specific subset of a point cloud. However, this classification might inherit noise and classification errors (see Figure 3.8). Therefore, *a priori* classification is not utilised in this step and the point cloud is reclassified as ground and non-ground classes by the lasground tool.

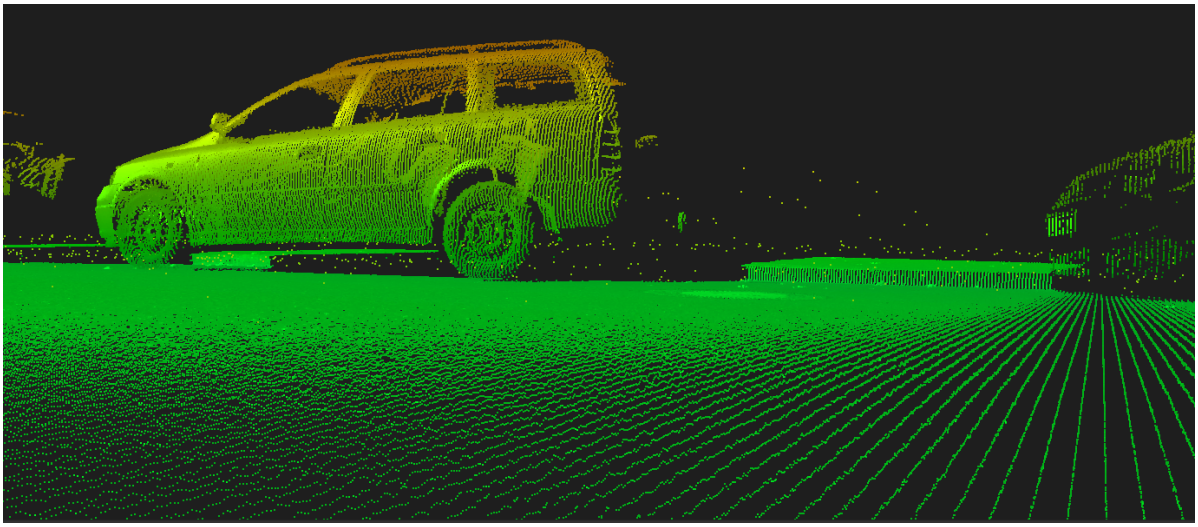

Figure 3.8.: Erroneous ALS point cloud filtered out in the Ground Points Filtering step. Those points are classified as *ground points* in a raw classification

The ALS point cloud is used in order to cover gaps resulting in filtering out outliers (like point clouds representing cars see Figure 3.5 and Figure 3.6). Without ALS the subsequent reconstruction of a surface would be possible through the use of interpolation algorithms but this results in less accurate surface representation w.r.t. to a real-world situation.

Generally, the LASGround solution works appropriately to find points in a point cloud that describe a ground surface. However, in case of noise present below the ground surface, the algorithm is not robust w.r.t. to noise. For instance, in the case of MLS data, the reflection of a signal allows mapping features that are not present at the ground surface but also underneath. This can be the case when a manhole is present on a road surface. This is because a manhole often has holes allowing for a signal to be transmitted through them.

This results in a classification of those noisy points as points describing ground surface and rejecting the points above which indeed describe a ground surface by the LAS tool (see Figure 3.9).

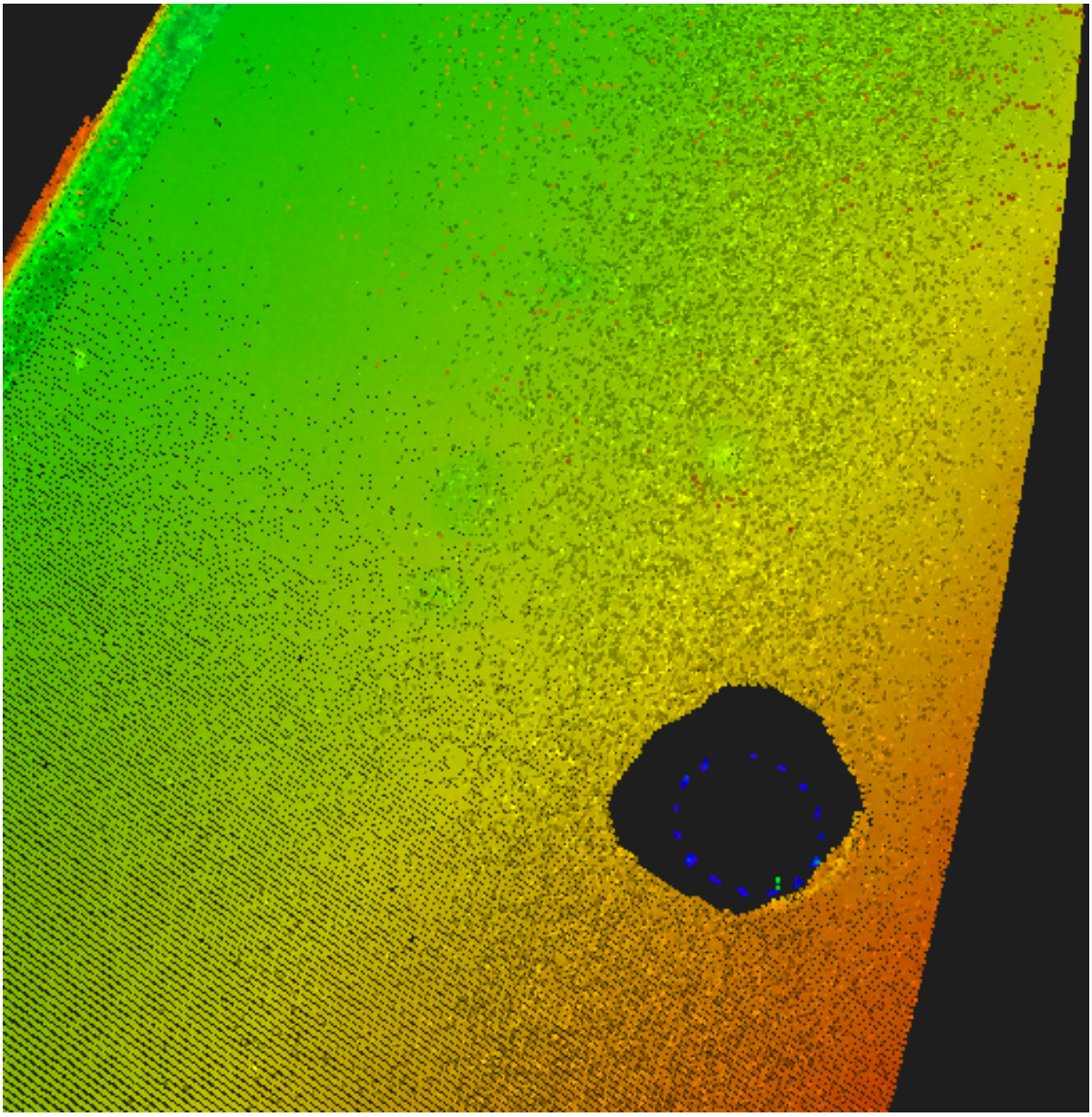

Figure 3.9.: The gap in dataset after a ground points filtering. Blue points classified incorrectly as a ground surface

In order to overcome that problem, the threshold is set to reject those noisy points. The process of erasing those points is controlled via parameter *Enforce manholes corrections* with values *True* or *False*. In case of the parameter set to *True* the algorithm splits point clouds

to 1 m x 1 m patches where a mode of height values is calculated. According to this mode value, a 10 cm value is subtracted from a mode and every point located below this threshold is rejected. This allows rejecting noisy data below manholes.

3.2.2. Vertical-like objects

In the case of vertical-like objects, the ALS point cloud is not processed. This is because the ALS covers all structures in the airborne field of view. Hence, it describes e.g. roof structures. This can result in false ground points detection (e.g. roofs as ground). Thus, the Ground Points Filtering operation for vertical-like objects is performed only for the MLS data with the same lasground tool settings as for horizontal-like objects.

The operation filters out most of the irrelevant horizontal structures in a point cloud subset (see Figure 3.10 where a situation before filtering is shown) but some outliers like trees still exist (see Figure 3.11)

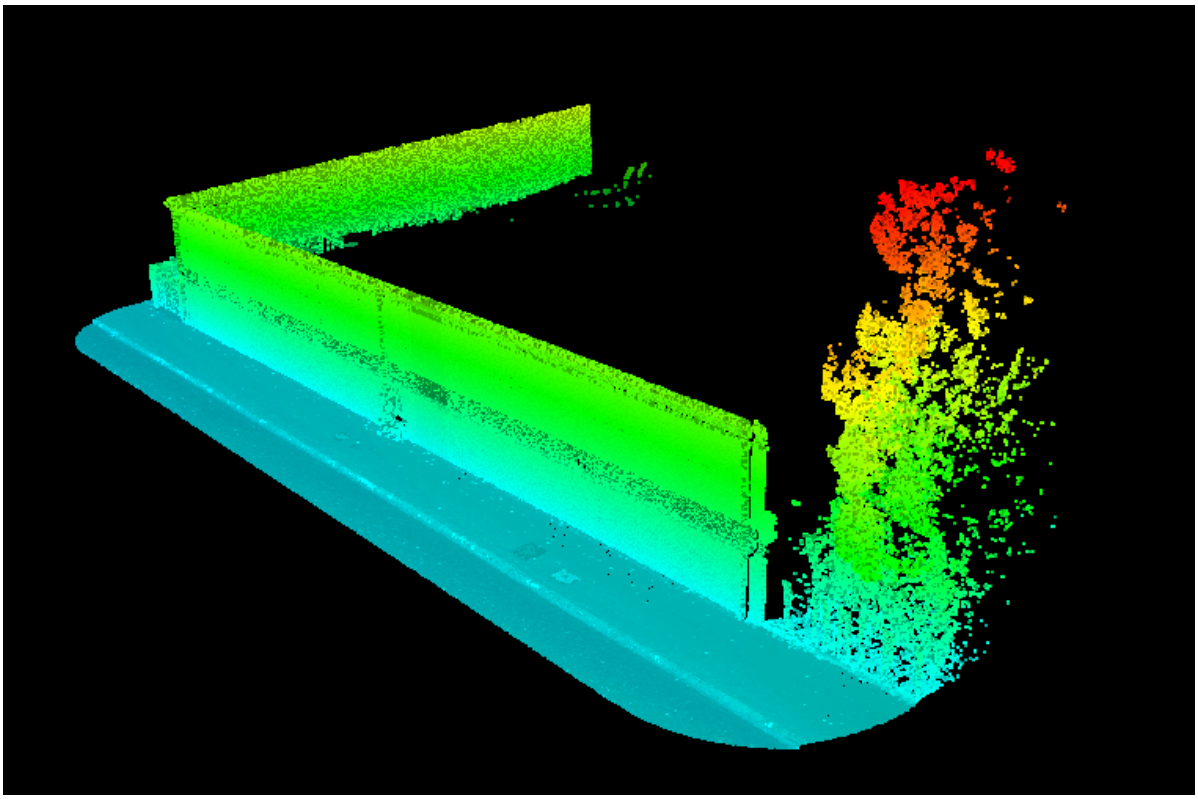

Figure 3.10.: Point cloud (MLS) cut to an area of interests of a building, before the Ground Points Filtering

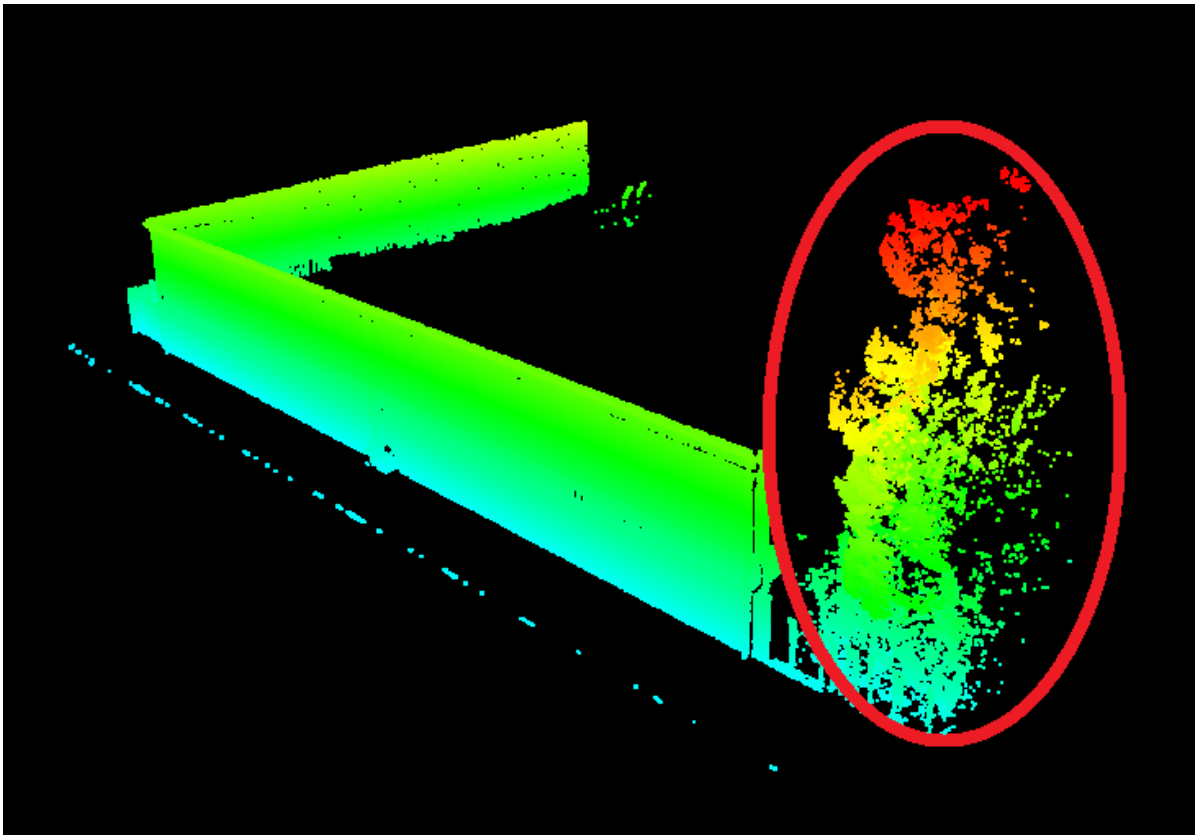

Figure 3.11.: Point cloud (MLS) cut to an area of interests of a building, after the Ground Points Filtering. The red circle indicates one of the outliers portion, a point cloud depicting tree

3.3. Segmentation

As presented in Figure 3.4 the vertical-like objects have to be additionally processed. Therefore, the Segmentation step is designed only for these types of objects.

Vertical segments of buildings from now on are called walls even though walls might consist of windows, doors and other structures. This term is introduced to simplify the description. Furthermore, the term *façade* is not used as it suggests that investigated structures are fronts of buildings which is not always a case.

First of all, the assumption that the MLS data acquired at the street level can only depict walls is made. Moreover, not all of the buildings' walls can be described by MLS point clouds as MLS is often mounted on a car that can only perform mapping within roads designed for vehicles. This may result in empty point cloud datasets for walls located behind an acquisition place e.g. in a backyard or walls that are adjacent, and thus they are not in a field of view of a scanner. Another possibility is that a wall facing the main road can be obstructed by other objects (trees, cars) that prevent the wall to be successfully reconstructed.

3.3.1. Buildings as groups of walls

Taking into consideration those assumptions and previous steps of the workflow it is possible to perceive a building as a group of walls.

Each wall has a specific subset of the point cloud which can depict it. Therefore, each input wall has a specific range within which a point cloud depicting the wall can be located. This range corresponds to the absolute accuracy of the input model (depending on LoD's accuracy) and can be applied as a buffer around each ground line segment of the wall similar to the Clipping step. The only difference is that not a simple buffer is introduced but an off-setter which made a rectangular area around each line. This is performed in order to decrease processing time in further processing steps.

3.3.2. Extraction of relevant subsets of point clouds depicting walls

Having areas of interests per each wall within a building it is possible to evaluate the influence of a point cloud acquisition location. MLS testing dataset of this project is obtained at the street level. Thus, only walls not covered by other objects like other buildings, trees etc. are described by point cloud subsets. In some cases, walls can be partially depicted by the point cloud. This can cause severe problems in the next step of the workflow - Surface Reconstruction as it needs relatively equally distributed point cloud at the wall surface in order to sufficiently perform the reconstruction. Therefore, it is important to reject all walls that are not sufficiently described by a point cloud subset. This allows to vastly decrease processing time as well as prevent yielding erroneous Surface Reconstruction results.

In order to do that the point cloud density per each wall is measured. The point cloud subset for each building is flattened to a two-dimensional space by erasing Z coordinates. Then, each point cloud is tiled to 2 m x 2 m polygons and a sum of points within the obtained squares is calculated (see Figure 3.13). Then, the method introduces a threshold that rejects sparse point clouds. Finally, the patches distribution over walls is checked. The expected results are presented in Figure 3.12.

The alternative approach could be a point cloud density calculation per wall's buffer. However, this solution disables an option to check whether a point cloud is spread evenly along a wall. There could be a situation that a portion of point cloud covers only a small part of a wall and this point cloud portion is densely populated in this part. Thus, based only on the buffer's density information the implemented algorithm will classify the wall as valid for further reconstruction but it will result in a not complete dataset for successful Surface Reconstruction in next steps.

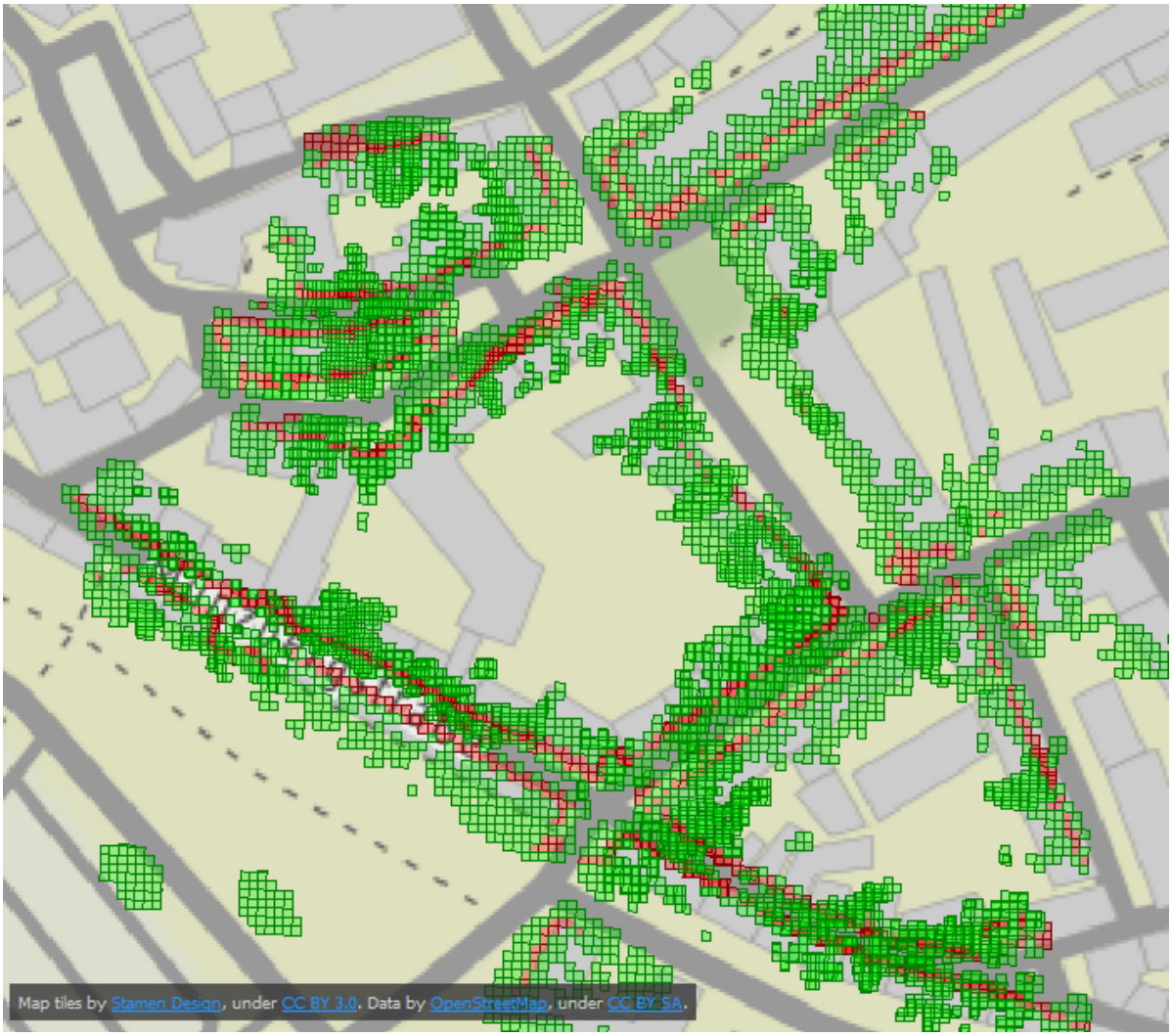

Figure 3.12.: 2D vector tiles depicting the density of point cloud subsets per buildings' group. Red tiles depict dense regions of a point cloud and green patches mark rejected and thus sparse point cloud areas

The attribute enables rejection of sparse and thus irrelevant patches - leaving only parts that have a high density of point clouds. In this case, 20 000 points per 2 m x 2 m polygon are set as a threshold (see Figure 3.14).

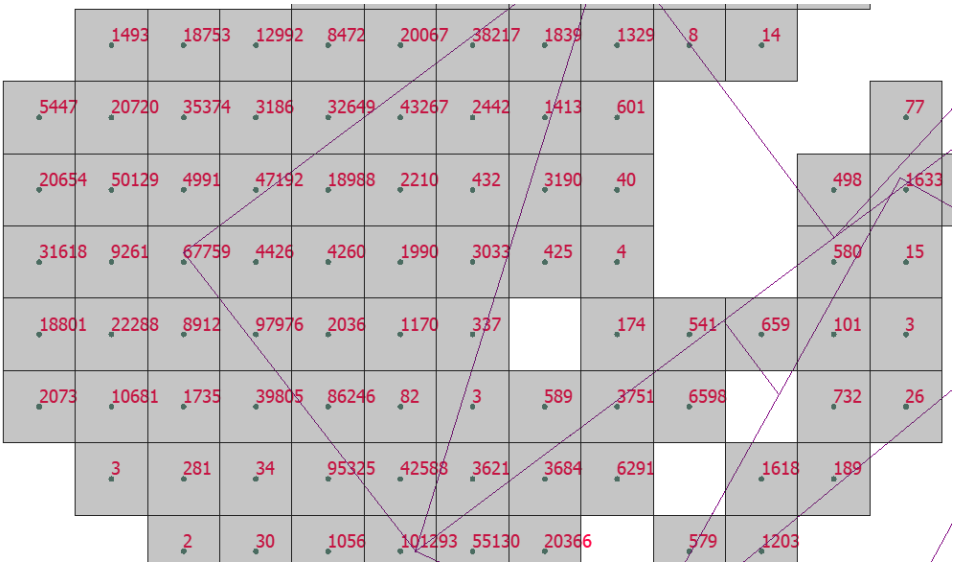

Figure 3.13.: 2D vector tiles depicting the density of point cloud subsets (grey-coloured rectangles with sum of points) per one randomly selected building

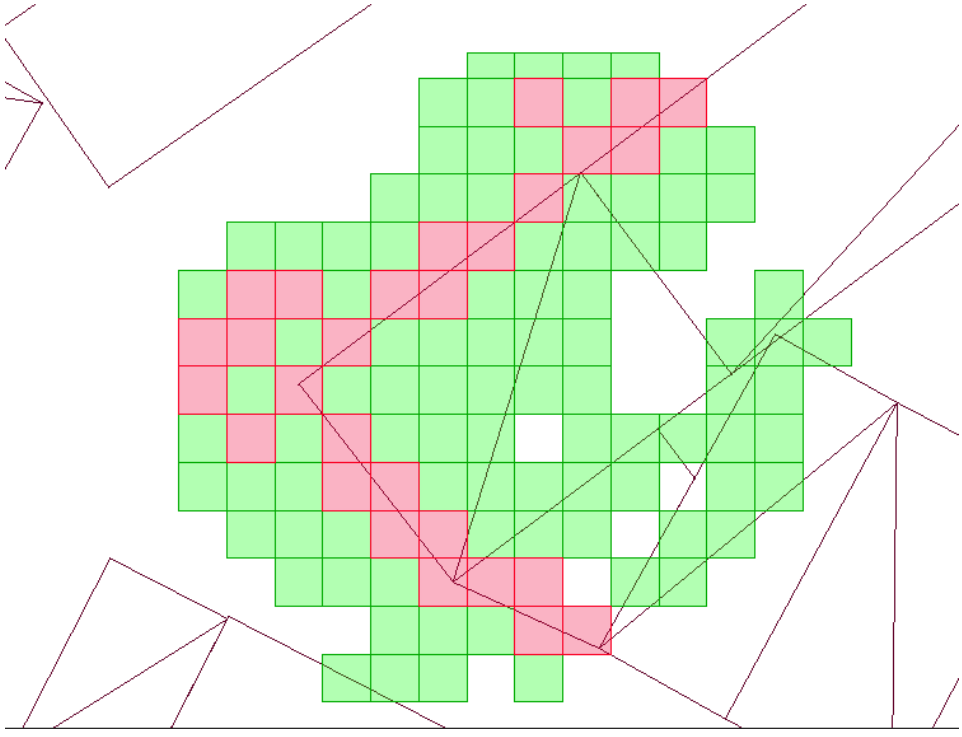

Figure 3.14.: 2D vector tiles depicting the rejected and accepted patches of point cloud subsets per one randomly selected building. Red tiles depict dense regions of a point cloud and green patches mark rejected and thus sparse point cloud areas. Buildings marked by dark edges

The tiles that meet the requirements are related to corresponding walls' areas of interests. The threshold here is set according to the LoD of the refined wall. For example, if a wall with an accuracy of LoD1 overlies or intersects dense tiles at least 5 times it is passed as a valid wall and subsequently processed (see Figure 3.15).

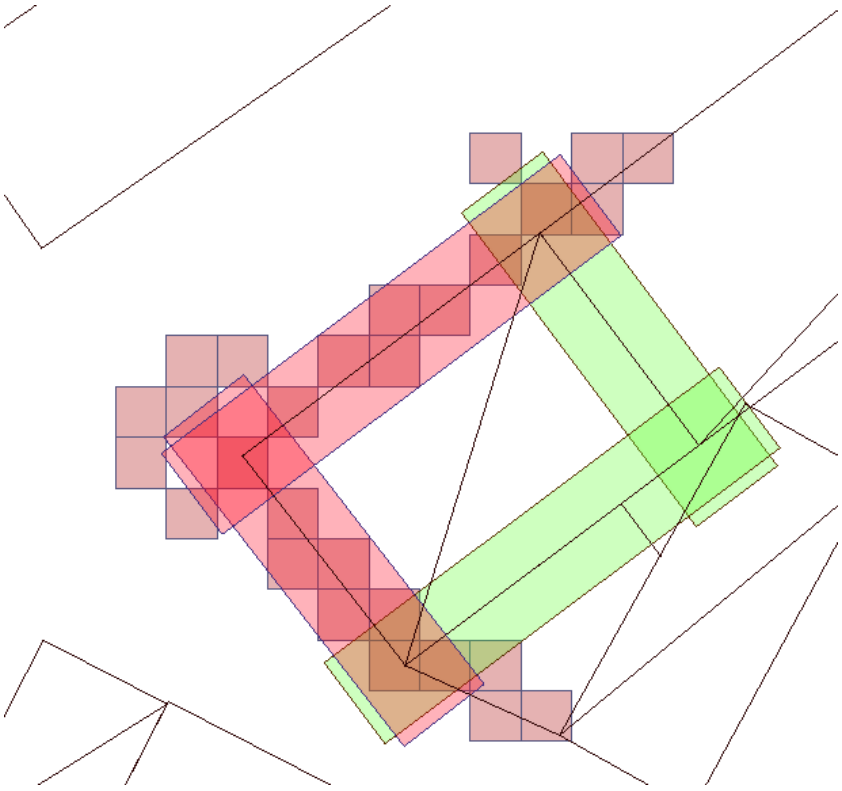

Figure 3.15.: Walls for further processing. Qualified walls marked red while rejected green. Tiles marking dense regions are red-coloured. Buildings marked by dark edges

Walls' regions that meet the previously stated requirements are used as cutting areas for point cloud subsets. Each point cloud subset depicting a specific building is cut by smaller, wall regions and assigned a unique wall id. The new unique id structure follows the rule *unique raw building id + index of wall within the building's group*.

The selection of walls enables the clipping of a point cloud subset. Within overlapping regions of walls, the point cloud is duplicated for each wall to prevent bias towards one wall. This duplication is processed in the next, more complex Segmentation step. Clipped subsets still may have outliers in the form of trees, cars etc. (see Figure 3.16). Therefore, more complex segmentation has to be applied.

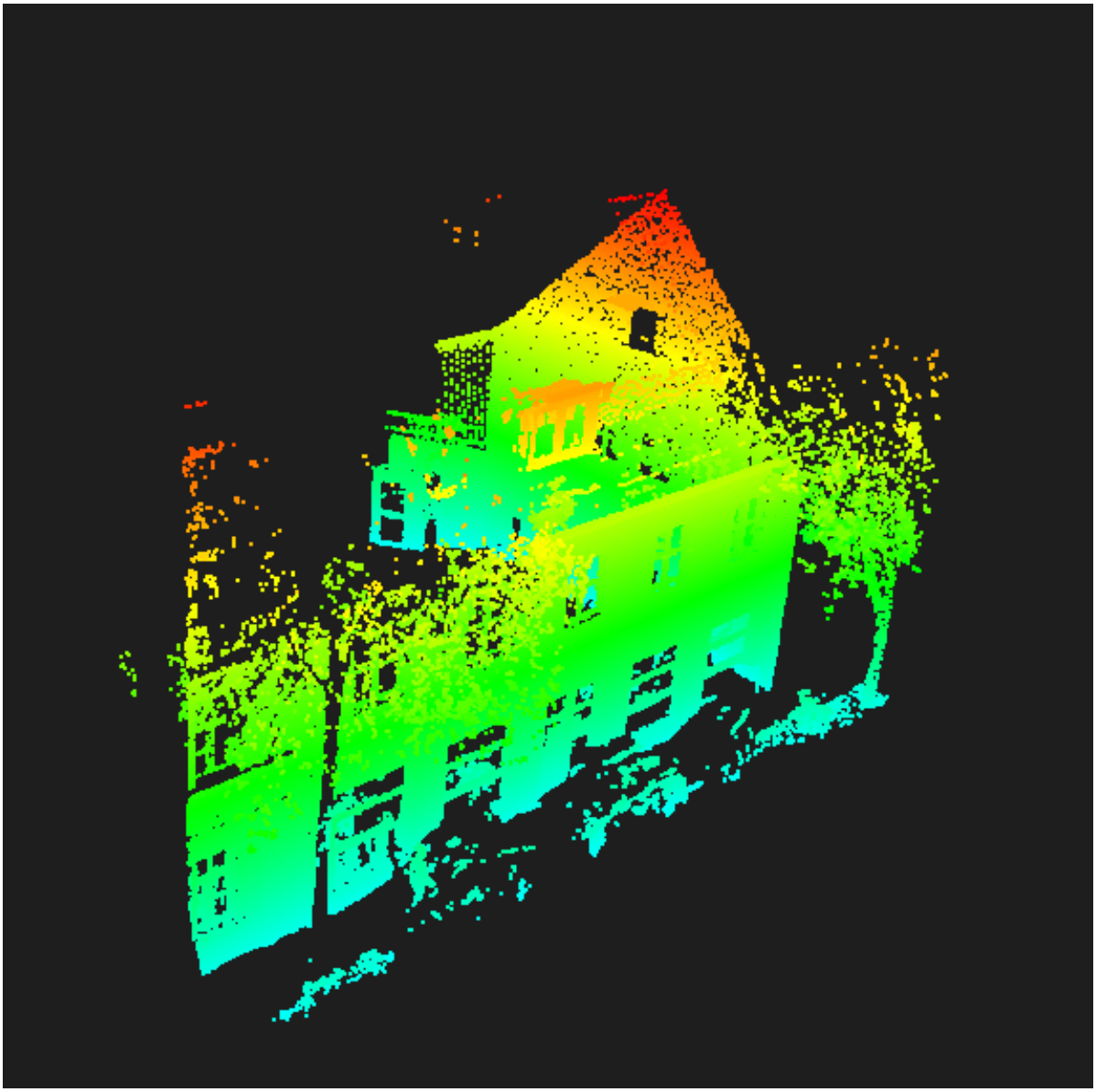

Figure 3.16.: A point cloud subset representing a building after selection of valid walls. Ground points already rejected and boundaries of walls' buffers visible. Outliers still present in the subset (e.g. trees, cars at the parking space)

3.3.3. Finding a plane to ultimately separate relevant point cloud subsets

As mentioned in the previous section, a more complex segmentation method has to be applied in order to select points from point cloud subsets describing only walls. In order to achieve that, the RANSAC algorithm is applied. The algorithm fits a plane into unstructured input data. Since point cloud subsets are unstructured spatial data describing planes like walls, the algorithm is used to find a wall within a given subset. The RANSAC algorithm fits only

one plane into one dataset. The splitting of the point cloud subset per each wall assured that RANSAC finds one single, main plane.

FME lacks of built-in function for the RANSAC algorithm. As one of the requirements for this work is the creation of an end-to-end tool the RANSAC algorithm has to be applied within the FME Workbench. The FME software has an option for integration of Python scripts through the PythonCaller function. As Python does not have a standard library for a fitting plane algorithm, the pyntcloud library is introduced. FME has several options for a package installation into the FME Python environment. However, due to compatibility reasons, the external Python 3.6 interpreter is utilised where a pyntcloud library and its dependencies are installed.

The RANSAC algorithm performs an operation on locally saved subsets of point clouds. Parameters of the algorithm are fine-tuned to this type of a dense point cloud.

- Model as a plane n and measurements data as an unordered set of points preprocessed in previous steps S
- The maximum number of selected minimum sets is set to 100 N
- The minimum number of inliers parameter is discarded as point cloud density and points in point cloud subsets per object are varying T
- The distance threshold is set to 0.1 t obtained by visual testing of the algorithm and high density of point clouds

After each iteration over subsets of point clouds, those are saved back locally. Then they are read out to the FME tool for further processing (see an overview in Figure 3.17).

The previous steps of the pipeline enable to shrink each area of interests and size of a point cloud. This allows for a successful application of a fitting plane algorithm. Without previous steps, the fitting plane algorithm would detect road surfaces, roofs, trees and other irrelevant objects represented by the point cloud. The illustration depicting a random building before RANSAC utilisation is shown in Figure 3.18 and for the scene after the application of the algorithm see Figure 3.19.

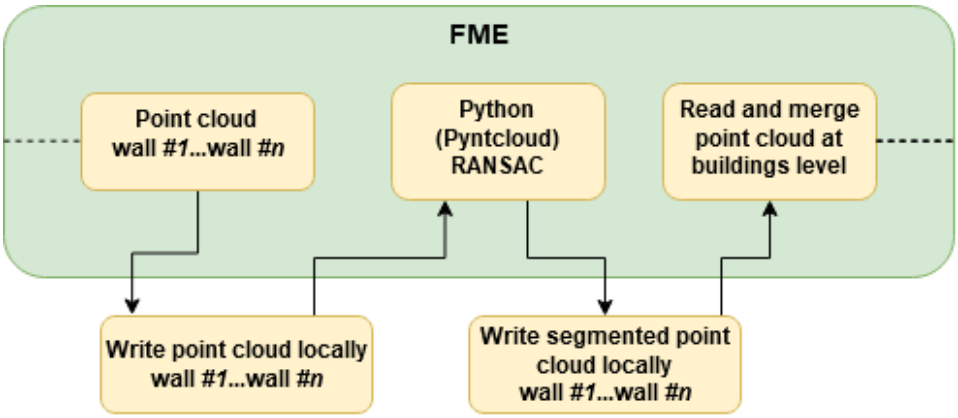

Figure 3.17.: An overview of the pyntcloud and RANSAC integration FME environment

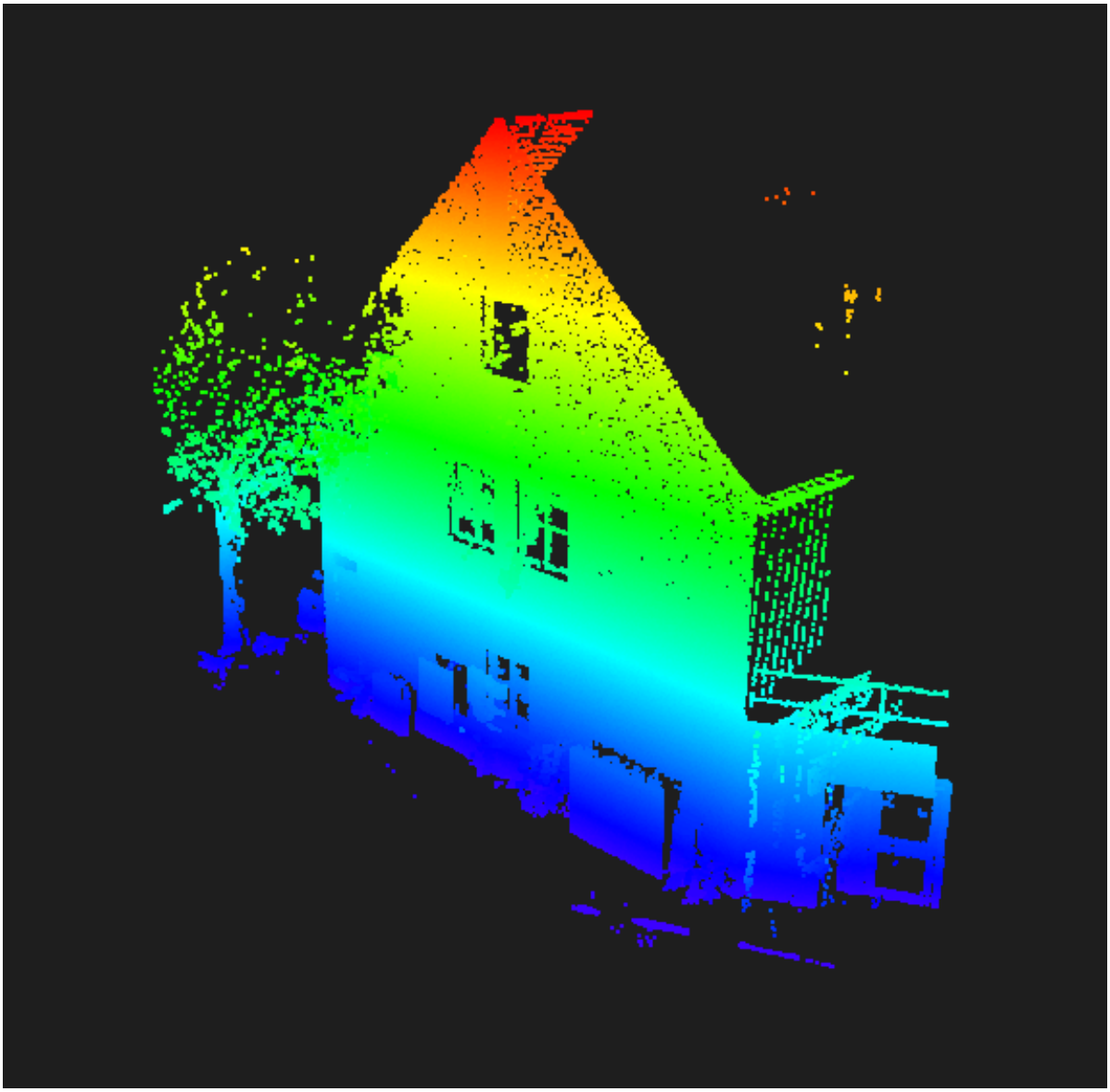

Figure 3.18.: A subset of point cloud depicting one of the building's walls before applying the RANSAC algorithm

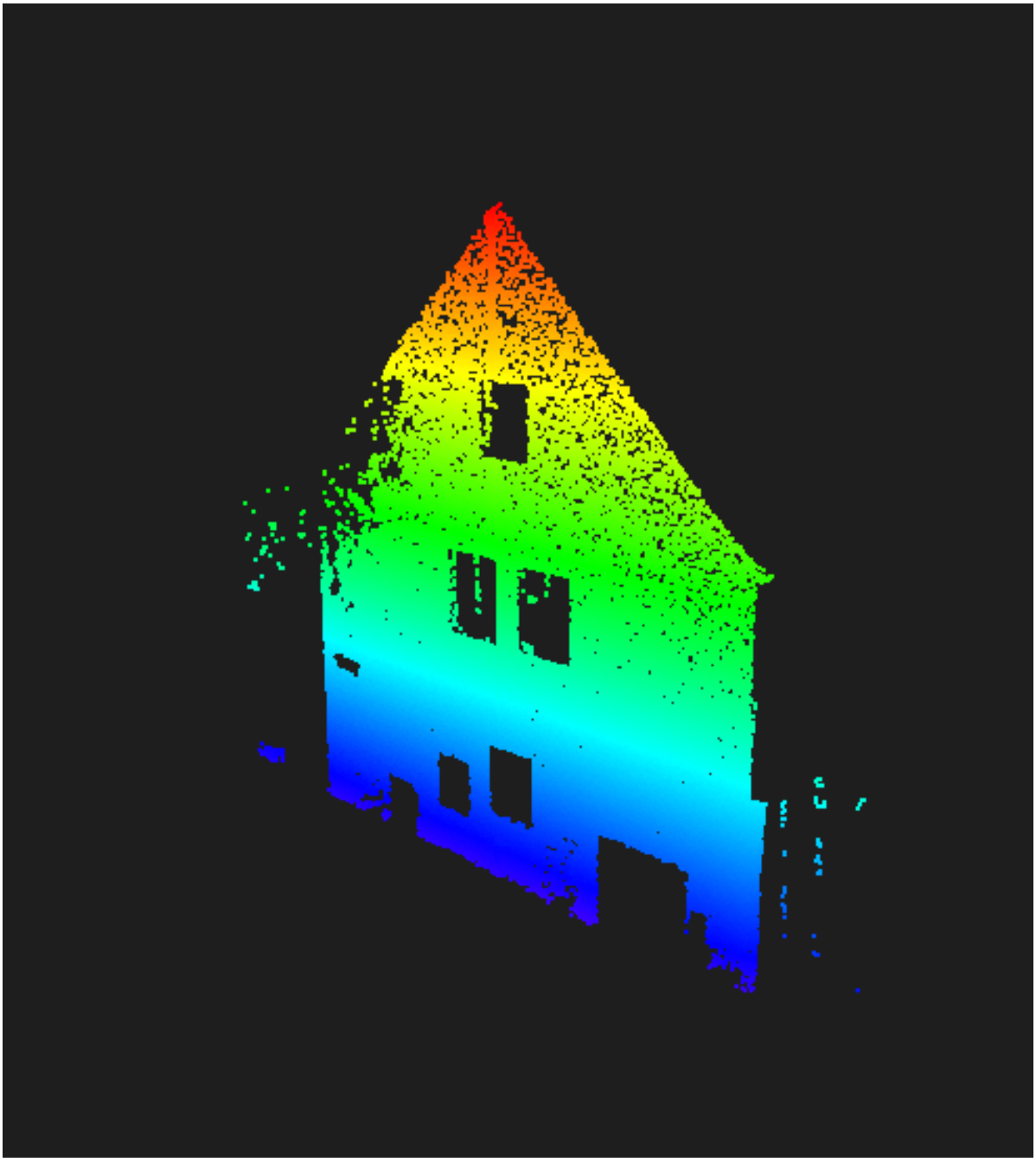

Figure 3.19.: A subset of point cloud depicting one of the building's walls after applying the RANSAC algorithm

After each point cloud describing a wall of a building is found, inliers can be aggregated back to the building level. This operation enables a successful reconstruction of the building's surfaces. It has to be underlined that from now on almost all points within point clouds should represent the building's walls. The implicit outliers filtering is applied in the Surface

Reconstruction section where distant, single points are labelled as irrelevant for a reconstruction. However, still, some noise like leaves or minor city structures may be present, thus the complete segmentation of point cloud is not achieved which however does not cause severe problems for the reconstruction.

3.4. Surface Reconstruction

In the previous steps, the processing of point cloud subsets in order to filter out non-relevant points for scene reconstruction is presented. Now, thanks to those results, it is possible to reconstruct a surface of specific objects. This section applies to vertical- and horizontal-like objects. However, some steps can slightly differ depending on the object type. In the case of no explicit headline, the solution applies to both vertical- and horizontal-like structures.

The goal of the Surface Reconstruction step is to reconstruct objects at specified fidelity which can be steered by a set of parameters. This function allows potential users to customise the Surface Reconstruction workflow to their needs.

The whole project workflow is designed to assure that even an inexperienced user in the field should be able to use the tool. That is why the Surface Reconstruction algorithm is integrated into FME and it is possible to steer the process via easy-to-use parameters accompanied by short text explanations. Moreover, default, fine-tuned values serve as suggestions for a potential operator. The typed-in or selected from the list parameters are translated into an XML document which is a configuration file for the MeshLab Server software.

3.4.1. Set a local coordinate system

The Surface Reconstruction part is the most demanding part of the workflow w.r.t. computational power. To decrease the processing time, objects' geolocation can be transformed into a local coordinate system where each object is located in the centre of the local coordinate system. Setting up of the origin of a new, local CRS in the feature's bounding box centre reduces processing time as it can operate with smaller, absolute numbers, and types - at a level of few meters instead of million meters.

The shift that has to be utilised to create a new reference system is calculated before the Surface Reconstruction process. After the Surface Reconstruction is successfully completed the corresponding shift is applied to locate objects in an appropriate, previous location.

Horizontal-like objects

The change of X, Y coordinates of the object is introduced by a direct shift to 0,0 (X, Y) coordinate and setting of a new local coordinate system. The 3rd dimension (Z-coordinate) has a decisive meaning for an orientation of the reconstructed surface of horizontal objects. The MeshLab tool (applied later on) calculates the orientation of surfaces by setting origin in 0,0,0 (X, Y, Z) of input data. To assure that the orientation of reconstructed objects is appropriate, an object is shifted by an additional, arbitrary, relatively large number in the

negative direction (e.g. -10 m) to assure that orientation of horizontal-like objects is facing upwards w.r.t. to the global coordinate system (see Figure 3.20).

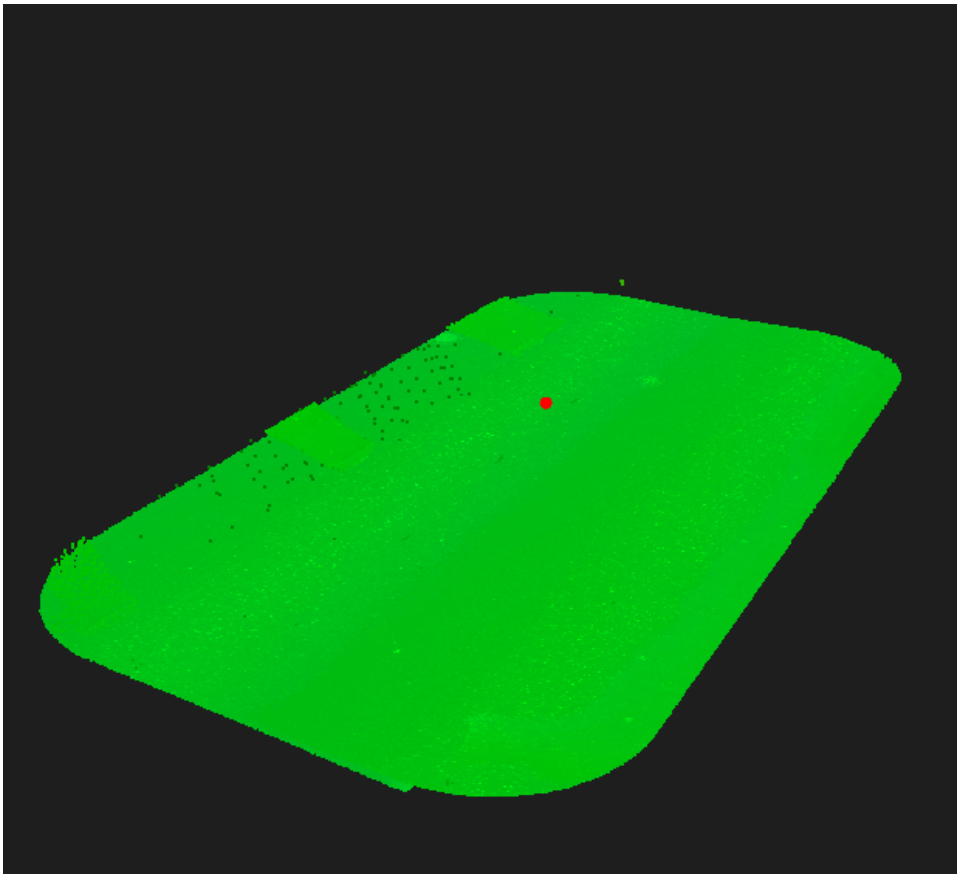

Figure 3.20.: The centre of a local coordinate system for a road segment. Red point indicates origin which is located 10 m above the data set

Vertical-like objects

The similar method can be applied to vertical-like objects. However here, each X, Y and Z coordinate can have an immense impact on the final reconstruction result. To prevent false orientation the shift is made according to the X, Y centre of raw buildings instead of processed point clouds. To avoid false faces at the bottom of objects their Z coordinates are shifted in the negative direction by 1 m similar to horizontal-like objects. This solution allows to orient reconstructed surfaces always towards the centre of the building currently processed (see Figure 3.21 and Figure 3.22). Then, after reconstruction is performed, the orientation can be simply swapped. This method allows to robustly reconstruct scene's orientation.

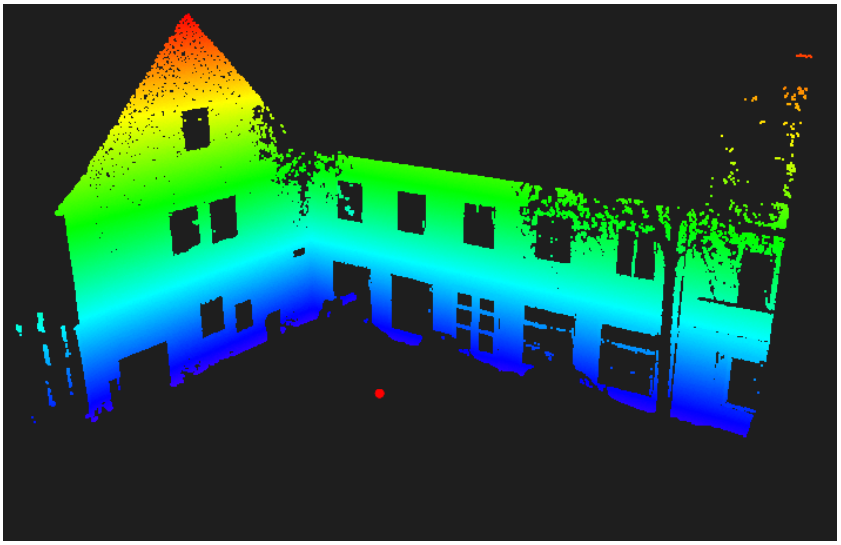

Figure 3.21.: The centre of a local coordinate system for a building. View from inside of the object. The red point indicates the local reference system origin

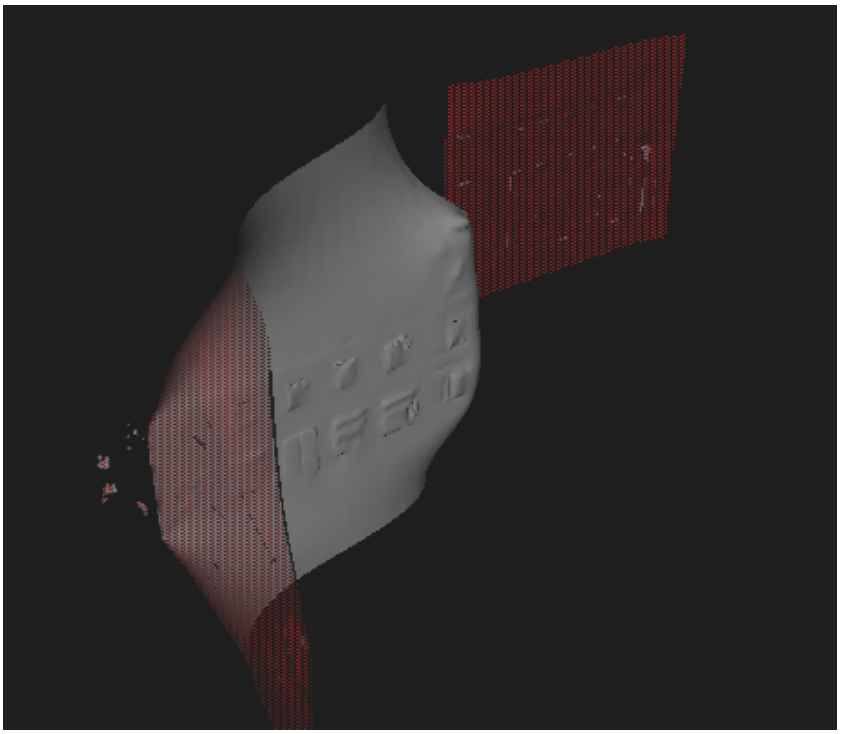

Figure 3.22.: An orientation of reconstructed surfaces. The front orientation (grey surfaces) and the back orientation (red dots). Now, simple swap allows to orient faces towards the desired, outward direction

3.4.2. Reconstruction of surfaces

The reconstruction of surfaces is performed by a mix of a configuration file and a command-line trigger which starts the MeshLab Server software (see Figure 3.23). In the MeshLab Server, the reading of a point cloud and the processing of point cloud subsets to meshes takes place.

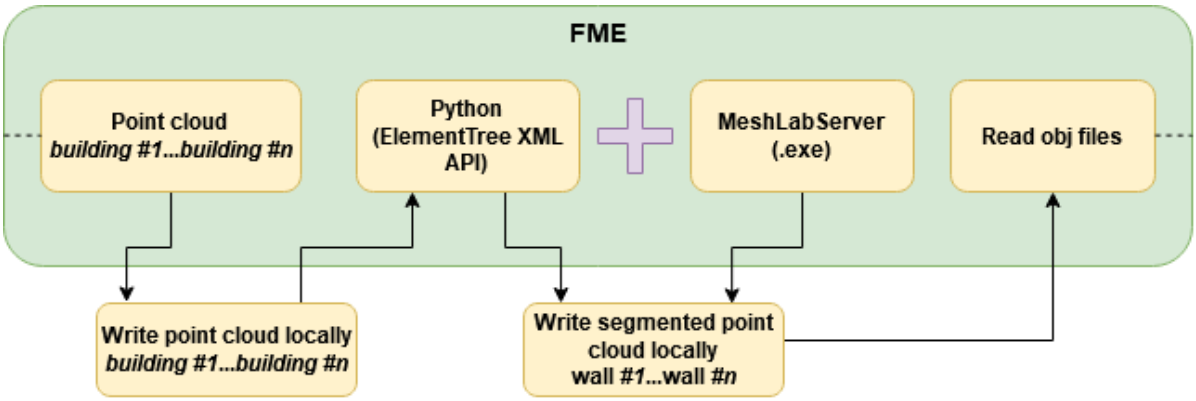

Figure 3.23.: An overview of the MeshLabServer integration within FME environment

A Python script incorporated into FME transforms a configuration template which is later parsed by MeshLab Server. The Python script is designed to allow for user input through published parameters of FME Workbench. This can be done using the GUI of FME and thus is an easy-to-use method. Those parameters should be defined before the workflow is run. However, fine-tuned parameters which are implemented as default values should also yield satisfactory results for most of the datasets having a similar characteristic. In total, there are eight parameters regarding the integration of Python, MeshLab Server and functions within MeshLab. The latter is responsible for the Screened-Poisson reconstruction algorithm and essential preprocessing steps in MeshLab.

The following pipeline applies within the MeshLab environment:

1. Load the point cloud as .XYZ text file
2. Compute normals for point sets w.r.t. to a viewpoint
3. Surface Reconstruction - Screened Poisson
4. Simplification - Quadric Edge Collapse Decimation

Not all parameters available within the pipeline can be modified by a user. Most of them are fine-tuned during testing and saved within a configuration file. Even though they are not customised via FME published parameters in case of occurring corner cases they could be changed manually within an XML configuration file or seamlessly added to the list of parameters. The four essential parameters were chosen to be enabled for customisation. The importance of those parameters is the main, decisive factor behind the choice. They influence

heavily the level of a final geometry refinement. This can be seen in the testing part of this project (see Chapter 4). The following parameters were chosen:

- Adaptive Octree Depth (Screened Poisson algorithm)
- Target number of faces (Quadric Edge Collapse Decimation algorithm)
- Percentage reduction (Quadric Edge Collapse Decimation algorithm)
- Post-simplification cleaning (Quadric Edge Collapse Decimation algorithm)

Generally speaking, Adaptive Octree Depth controls a resolution of the reconstructed surface (see Figure 3.24). A greater number results in a higher resolution of the output mesh. However, it comes with a great computational cost. It has to be weighted between computational time and final requirements for a 3D model. The value 10 is proposed in this workflow as a satisfactory depth for vertical- and horizontal-like city structures.

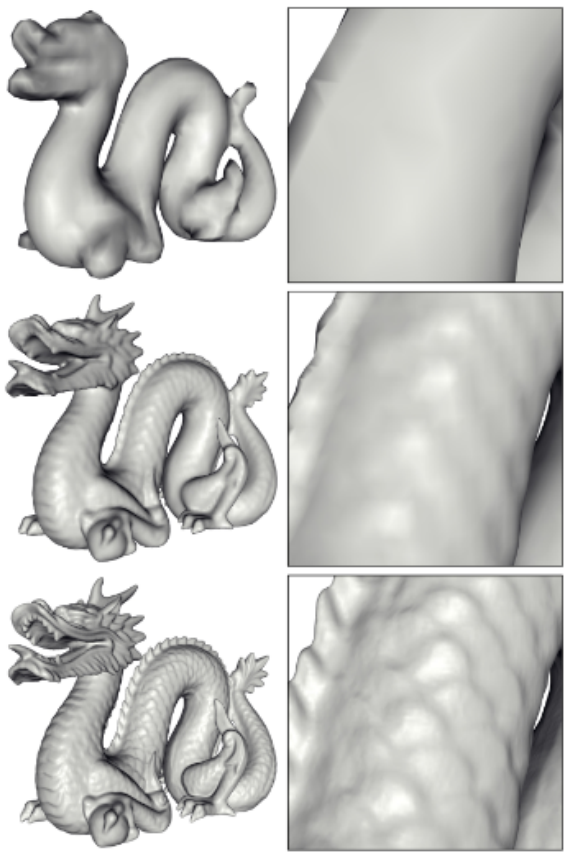

Figure 3.24.: Reconstructions of a dragon model at different octree depths 6 (top), 8 (middle), and 10 (bottom) [Kazhdan, Bolitho, & Hoppe, 2006]

The target number of faces and percentage reduction parameters cannot be used simultaneously. One can choose only one parameter while rejecting the other. Those are designed to control the level of details of the output mesh. At the first step, the octree depth parameter decides what should be the resolution of the reconstruction. This allows achieving high fidelity of the refinement. However, this can result in a very large mesh even for a relatively small object. The Target number of faces can control the algorithm to simplify a complex mesh to a fixed number of faces. This parameter is prone to errors as a user often does not know which value will result in a satisfactory result. Thus, the Percentage reduction is introduced. By typing-in, a percentage of a current mesh to be reduced to, the user, can utilise his knowledge about the expected result and estimate the needed value.

The flag Post-simplification cleaning with binary list True or False allows rejecting faces which are badly reconstructed. *Badly* means that unreferenced vertices, bad faces, and similar are erased. In most cases, this option yields satisfactory results. In case of uncommon settings or input data, the flag can be switched off.

3.4.3. Erasing not relevant faces

The Screened-Poisson algorithm enforces the creation of a continuous surface within a given range and distribution of input data. This stands as an advantage in case of unstructured data with the possibility of having holes or missing parts as it can easily interpolate those values (to some extent). On the other hand, it results in an augmentation of the extent. Therefore, the output mesh of a point cloud covers a larger area than the input data. In order to reduce that effect, certain measures can be taken. One solution can be an introduction of a threshold which will eliminate relatively large (w.r.t. to particular object) edges. This approach, however, can exclude desirable edges as the augmentation takes place within a dataset as well (e.g. in case of holes). Another other idea is to manually select not desirable parts of 3D models and delete them. The solution proves to provide satisfactory results but it is a time-consuming task which additionally needs a skilful operator with knowledge about the dataset being processed and software editing experience. Obviously, this approach is excluded from this work since the focus is on the automatic reconstruction of objects.

The method which is suitable for the task of a city models reconstruction is a shrinkage of the object's extent to the raw extent of an investigated model. This assures correct topological relation between adjacent objects within the city model and does not require manual editing by an operator. The final refinement depends on input geometries - this ultimately biases the result towards a raw geometry.

Horizontal-like objects

The cutting of non-relevant parts of horizontal-like objects stands as a relatively inexpensive computational task. The 3D reconstructed mesh object is split from mesh geometry to polygons which can be then cut by a corresponding 2D extent of a raw object (see Figure 3.25 and Figure 3.26). As horizontal objects are actually 2.5D objects, a 2.5D model is cut by a 2D extent which results in a relatively computationally inexpensive task.

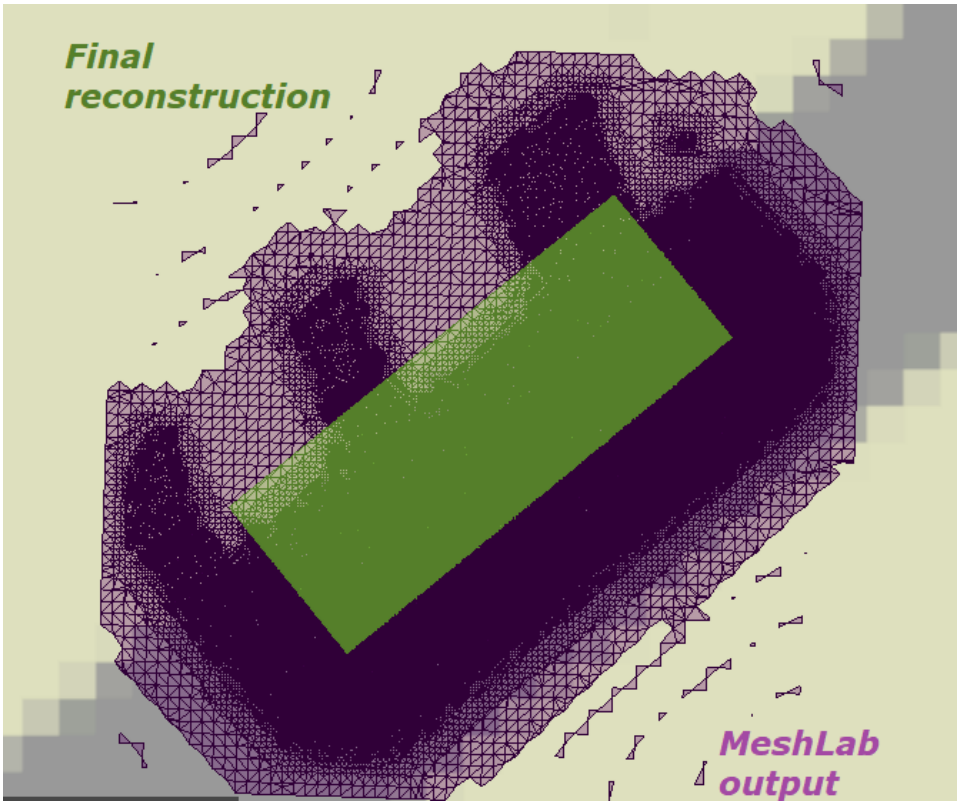

Figure 3.25.: The reconstructed mesh (purple-coloured) cut to the raw area of a reconstructed object (green-coloured) in 2D view

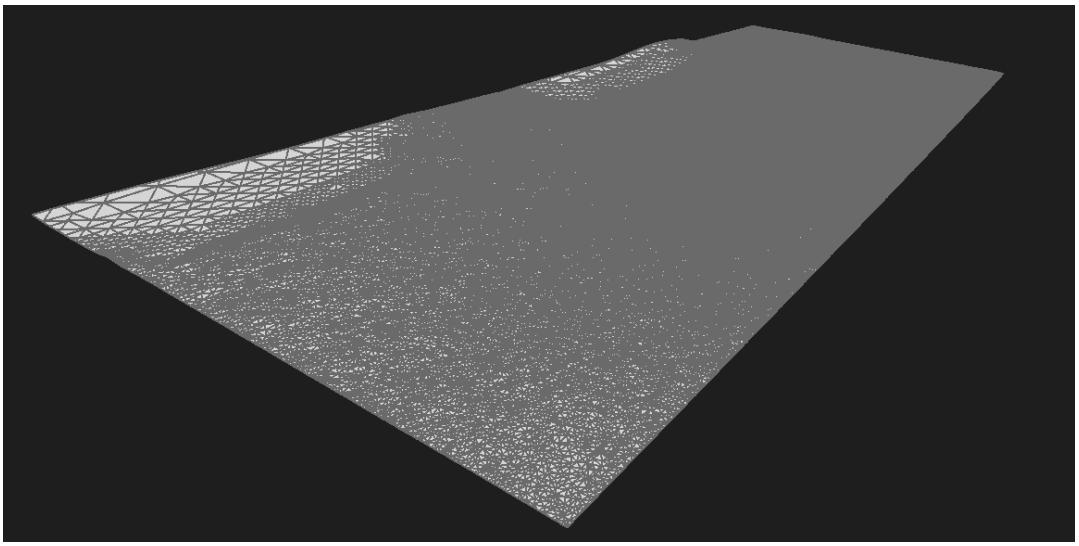

Figure 3.26.: Reconstructed mesh cut to the raw area of a reconstructed object in 3D view. 10% of simplification is applied and octree depth is set to 14

Vertical-like objects

On the other hand, vertical-like objects are pure 3D features. Therefore, the cut by a 2D extent of a vertical-like structure will not shrink the extent of e.g. a wall in terms of structure's height (see Figure 3.27). In order to prevent operation of a high computational cost, the cutting of 3D object by 3D extents is applied as follows:

1. Swap Y with Z coordinate within a raw and reconstructed object
2. Extract maximal possible 3D boundaries (see the Clipping section in Chapter 3) of the raw model and clip out outstanding parts in the reconstructed object - from now on called GeneralMeshCutter (see Figure 3.29)
3. Relate remaining reconstructed parts with a raw model to acquire semantics and relate single structures of raw buildings (e.g. walls) with single reconstructed parts (e.g. polygons constructing walls)
4. Perform a more precise cut-out, based on a grouping by a previous relationship to shrink a reconstructed surface to a corresponding raw structure - from now on called ExactMeshCutter (see Figure 3.30)
5. Swap Z with Y in order to come back to the initial CRS

Those steps assure that the processing time should be minimized as the process of 3D cutting is transformed to 2.5D problem (see Figure 3.28) similarly to horizontal-like objects mentioned before. The first cut (GeneralMeshCutter) reduces faces which are relatively far away from the wall area which results in a reduction of processing polygons. Then, the ExactMeshCutter clip out each wall individually thanks to introduced relation between the raw and the reconstructed object.

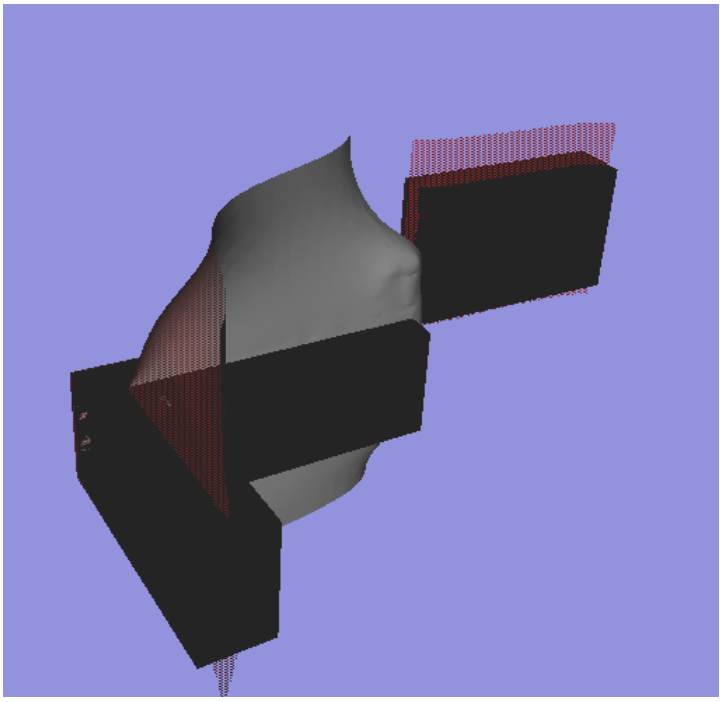

Figure 3.27.: Maximal ranges (dark grey) and reconstructed surfaces (light grey) in the standard CRS

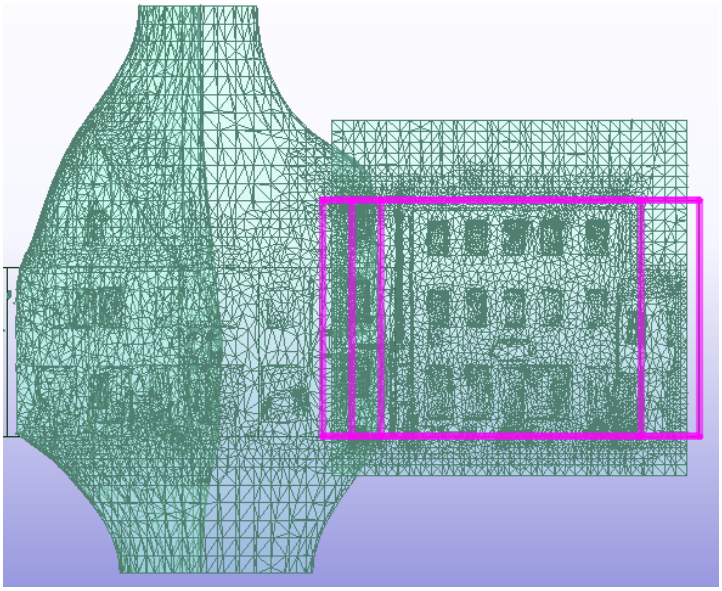

Figure 3.28.: Maximal ranges (one highlighted by pink colour) and reconstructed surfaces (light green) in 2D view, swapped CRS

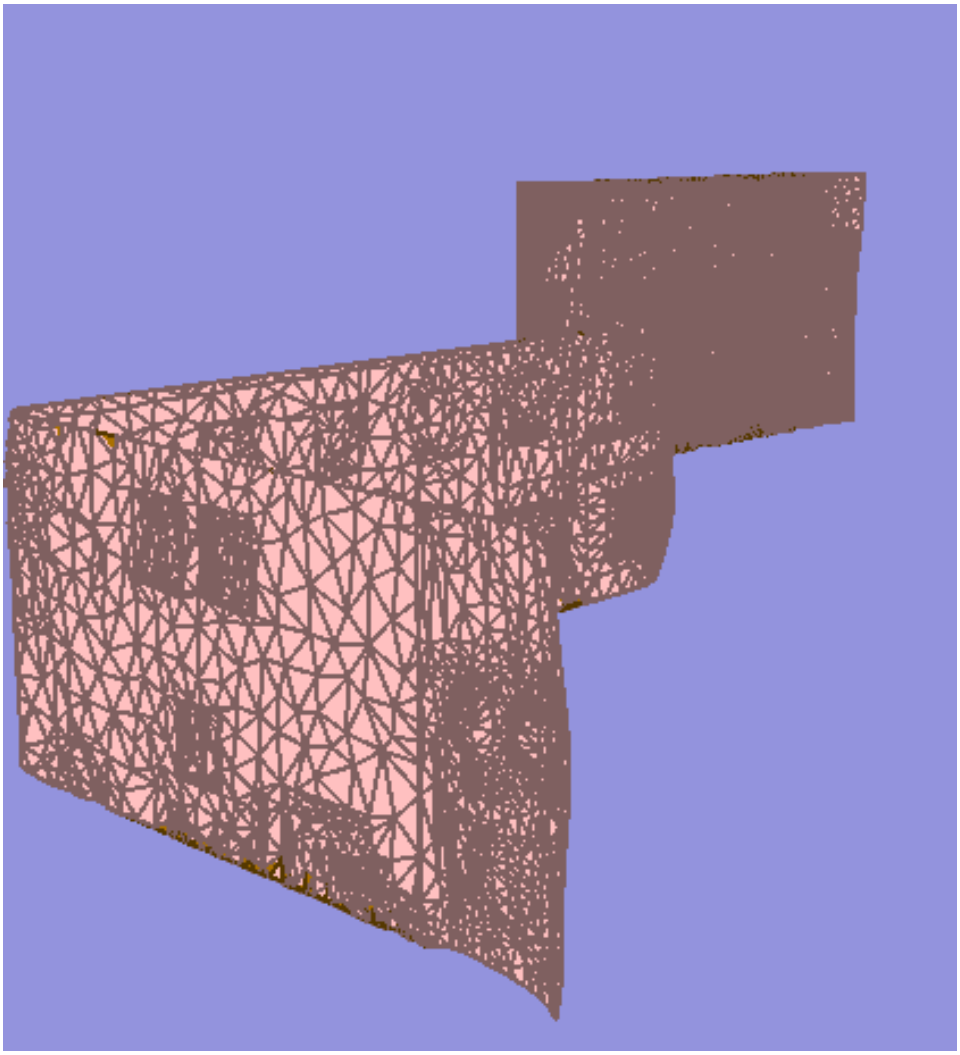

Figure 3.29.: Reconstructed surfaces after GeneralMeshCutter, Z swapped with Y - back to the standard CRS

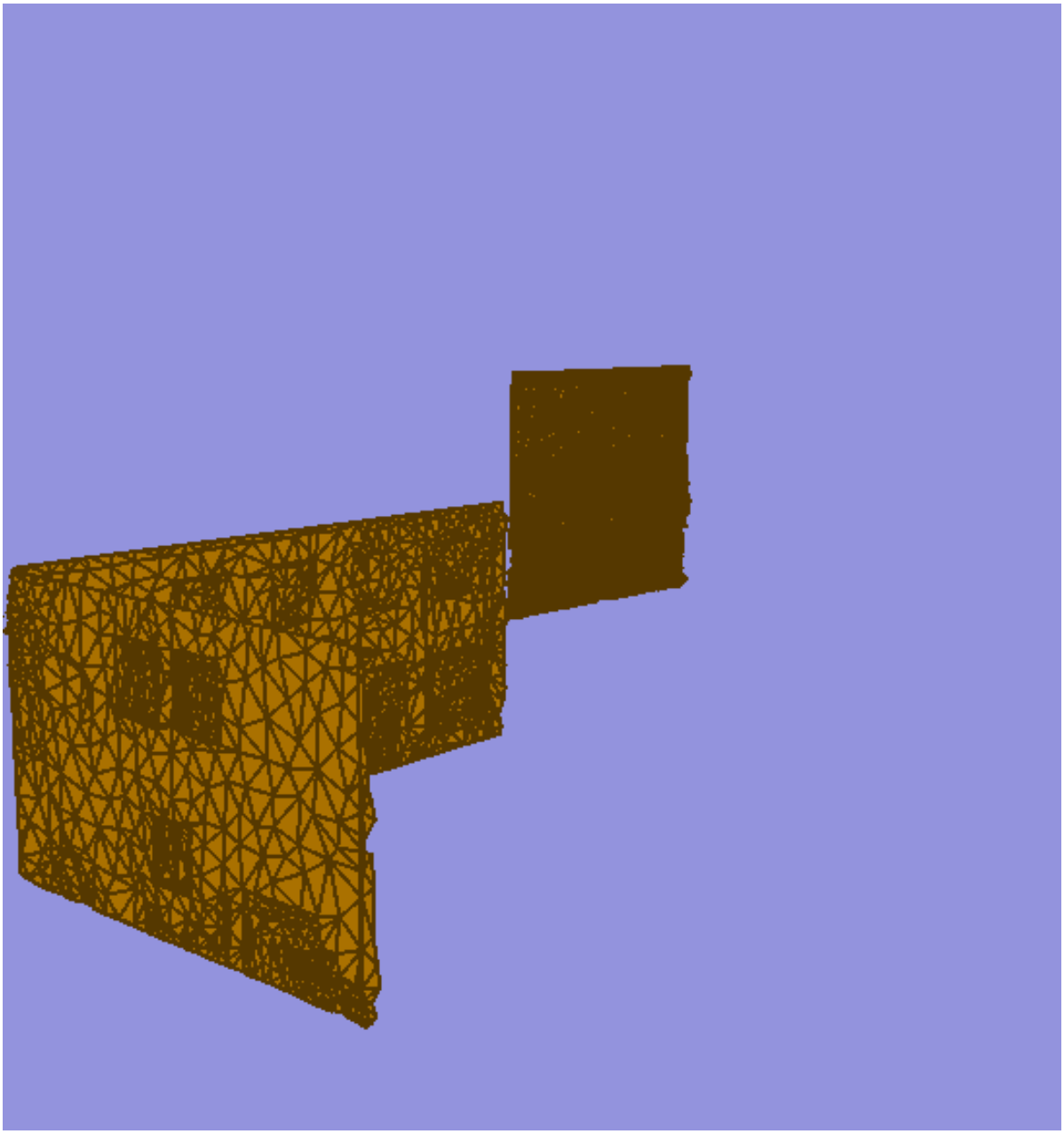

Figure 3.30.: Reconstructed surfaces after ExactMeshCutter, Z swapped with Y - back to the standard CRS

3.4.4. Assignment of semantics to reconstructed surfaces

In the case of both horizontal- and vertical-like objects, the cut assigns semantics from a raw model in an unchanged manner. This assures compatibility with the rest of the not refined city model. However, in order to distinguish newly created objects, generic attributes are

introduced. *Timestamp* indicates the date of refinement of a geometry following the UTC format. This attribute allows for easier managing of recreated surfaces and comparison to the not refined ones. Also, *FeatureNr* enhances the easiness of producing desired output as it allows to count how many polygons construct recreated surfaces per each building. However, the first of attributes to query output city models is *HasGeoRefined* which has either *True* or *False* value when a geometry of an object is reconstructed or not respectively.

3.4.5. Adding refined geometries to the city model

Depending on a type of a city model object's class reconstructed objects can swap a raw geometry or be added as an additional geometry. Allowed GML geometries suitable for storing such refined object are saved as *MultiSurface*. This design is not rigid as the *CompositeSurface* which does not allow for overlap. For example, to create a compliant CityGML 2.0 building representation it is possible to use class *Building* and *BuildingPart* which refer to one building object. In each of those classes, a geometry can be stored. The *Building* can hold generic and raw attributes of an object as well as raw geometry while the *BuildingPart* can be used to store refined geometry. This solution is introduced in this workflow.

In case of structures like roads, the CityGML *Road* class does not have such an augmentation. Therefore, a raw geometrical representation is swapped with a refined one and raw attributes are assigned to a new geometry together with generic attributes mentioned before.

3.5. Semantic Enrichment

Not only the geometry itself can be refined by means of point clouds and existing maps but also automatically detected additional features can be added. Moreover, new city models (like CityGML 3.0) standards arise. This enables a new design of semantic objects - more suitable for the needs of novel applications.

Researchers have already acknowledged the advantage of having intensity values recorded by a scanner (see Figure 3.31). For example, intensity values suggest the material of a depicted object. However, the intensity itself cannot be a definite factor which distinguishes between materials as the intensity value varies depending on instrumental effects, acquisition geometry, environmental effects [Kashani, Olsen, Parrish, & Wilson, 2015].

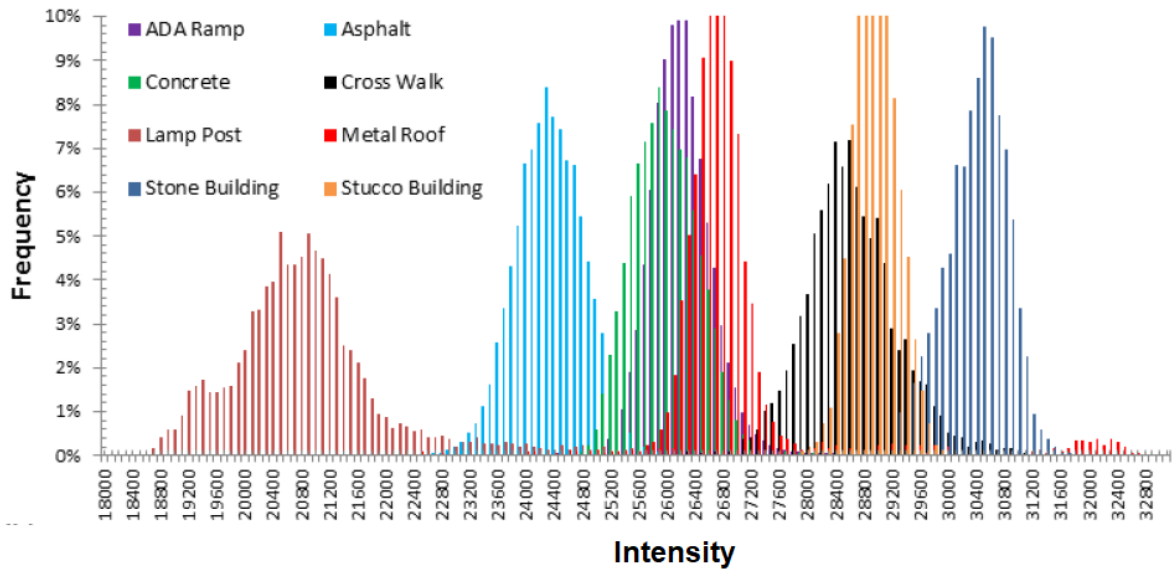

Figure 3.31.: Histograms of intensity values acquired on different surfaces. Courtesy of Kashani et al. [Kashani, Olsen, Parrish, & Wilson, 2015]

In this section, a workflow enabling manholes detection and mapping are presented. It utilises knowledge of refined geometries and intensity values of MLS point clouds presented in previous chapters. The approach introduces a CityGML 3.0 Transportation module as an output format that has specific classes *Hole* and *HoleSurface* created for modelling of manholes.

3.5.1. Manholes

The manhole is a structure which can be perceived as a distinctive feature of a road surface. Those differences are visible in a structure, a material, and a shape. There could be several types of manholes covers depending on the utilisation like water, sewers, gas etc. Within this study case, the focus is on the water manhole cover. This type of cover follows a certain pattern in terms of a structure, a material, and a shape. The study area for this section is set in Germany and thus a certain common manhole's cover type to describe a water manhole can be applied (see Figure 3.32 and Figure 3.33). However, it is believed that this approach works for other manhole's cover types and countries after simple adjustments.

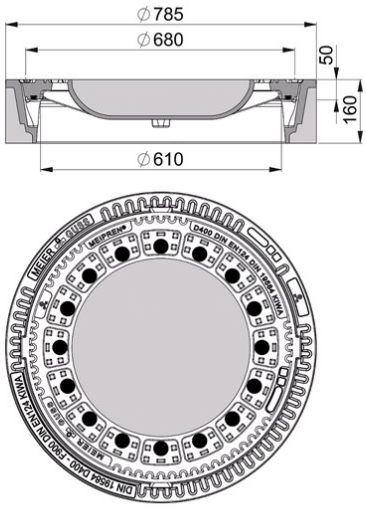

Figure 3.32.: The size of a manhole cover typical for German roads [Kemmler Baustoffe GmbH, 2020]

Figure 3.33.: An example of a road segment in Munich, Bavaria, Germany with a visible manhole

3.5.2. Selection of a point cloud within road segment

In the first step, an area of investigations has to be shrink to the area of interest of a specific road segment. Since the workflow should detect manholes located on a road surface only segments designed for vehicles are taken into account. To shrink the area, boundaries of road segments are used to cut a portion of a point cloud (see Figure 3.34). The important input from the previous workflow is Ground Points Filtering. This enables the filtering of road-irrelevant points in a point cloud and allows for better detection results.

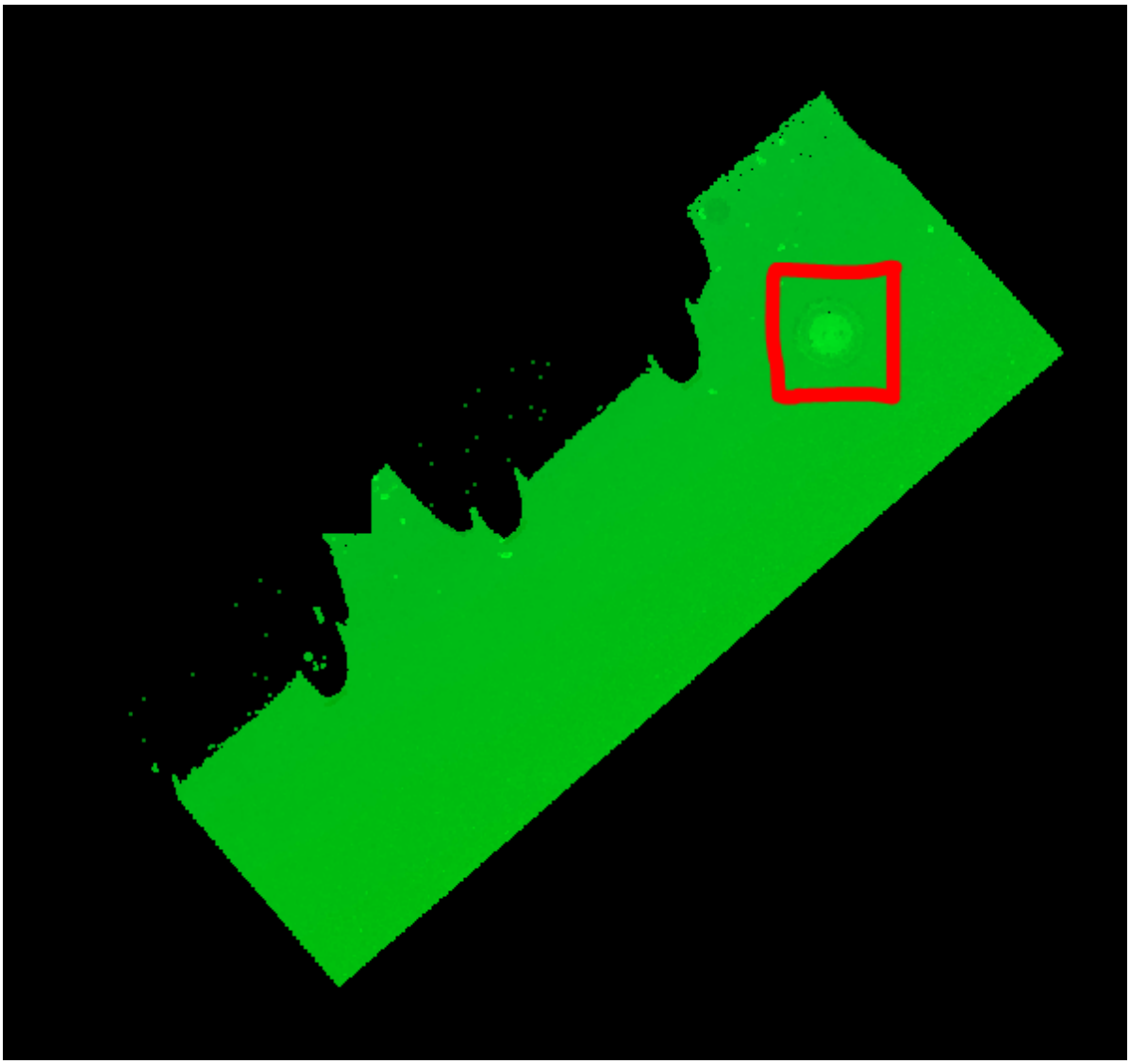

Figure 3.34.: An example of a point cloud depicting a road segment with a visible manhole, area marked by the red rectangle. The Ground Points Filtering method applied to filter out irrelevant points

3.5.3. Rescaling of an input point cloud

An input point cloud can vary in terms of an intensity scale depending on the manufacturer of the scanner and/or post-processing methods. Therefore, in order to match the target intensity scale presented by Kashani et al. the rescaling of the input point cloud is applied [Kashani, Olsen, Parrish, & Wilson, 2015]. The min-max normalization can be applied to scale values to the target range of [18000 - 32800].

3.5.4. Threshold to separate manhole's distinctive parts

The most distinctive part of a manhole is its concrete filling. However, the intensity values of testing dataset suggest that the filling corresponds to the *stucco building* class from Figure 3.31 of Kashani et al. [Kashani, Olsen, Parrish, & Wilson, 2015]. Thorough visual investigations of water manholes coverings proved that indeed coverings are made of concrete but a rough one - at least in the testing area of Ingolstadt, Bavaria, Germany. This results in different intensity values, more similar to the stucco material. Therefore, a threshold in range 28400 to 29200 is applied. It has to be emphasised that as mentioned before, this threshold cannot be perceived as a definite segmentation factor (see Figure 3.35).

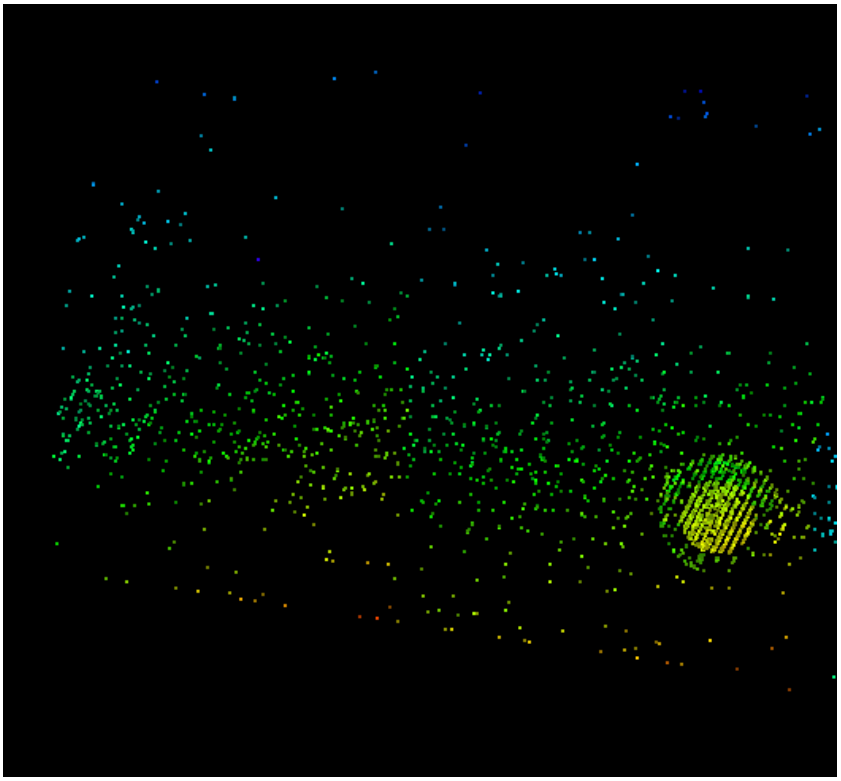

Figure 3.35.: The point cloud representing a road segment with intensities only in range 28400 to 29200. A manhole visible on the right

3.5.5. Finding a centre of a manhole

As it is presented in Figure 3.35, the density of a point cloud is higher in the area of a manhole comparing to other sectors. Therefore, it should be possible to filter out noise using the density of a point cloud. In order to do that the point cloud is transformed into an image storing number of points as a band value. The image is designed to have pixels of size 0.1 m x 0.1 m. They serve now as patches of point cloud representing the corresponding density. For the sake of further processing, pixels are coerced to points located in pixels centres. Points hold an attribute representing a total number of points within a patch (see Figure 3.36).

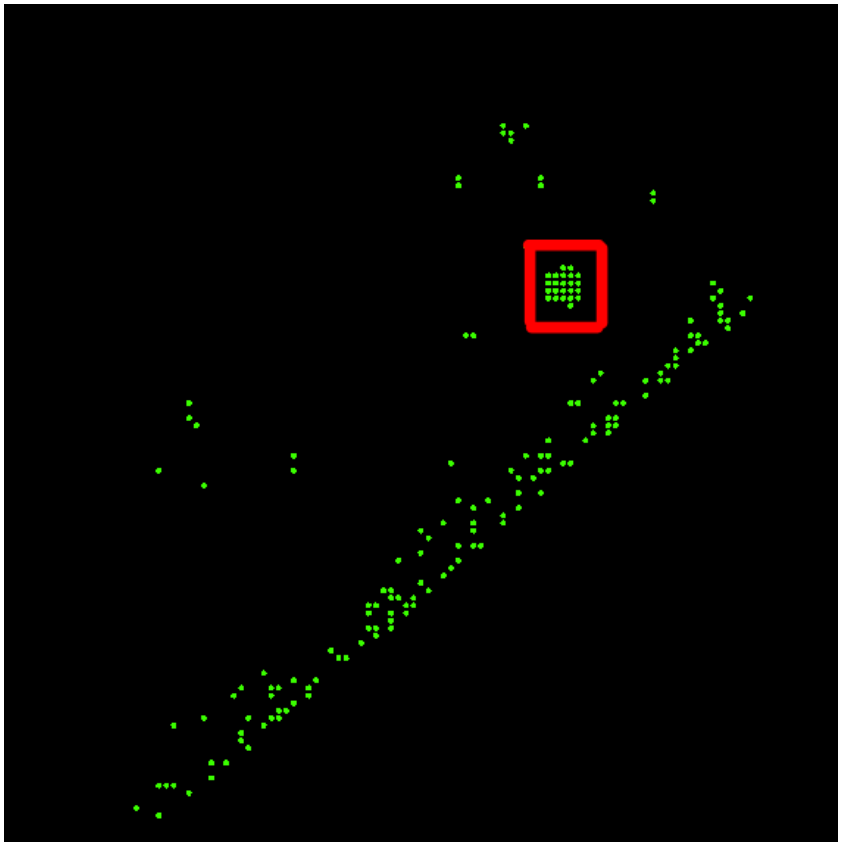

Figure 3.36.: An example of a road segment with a visible high density area (marked by red rectangle) where a manhole is located

Afterwards, the ten most dense regions (represented now as points) are selected for further investigation. Due to that, sparse regions are rejected and the most probable guess is assured. To finally assess whether this region contains a manhole a test to check an overlap is performed. Taking into consideration the previous patch size (0.1 m x 0.1 m) a buffer of a respective size has to be introduced. To check the density of a ten patches region a buffer of 0.2 m x 0.2 m is introduced (see Figure 3.37). This allows inspecting whether the ten most dense buffered patches overlap at least five times to filter out unrelated patches and decide

whether a manhole is detected. If there is such an overlap of regions found within a road segment, then the most overlapping polygon part is picked as the centre of a manhole.

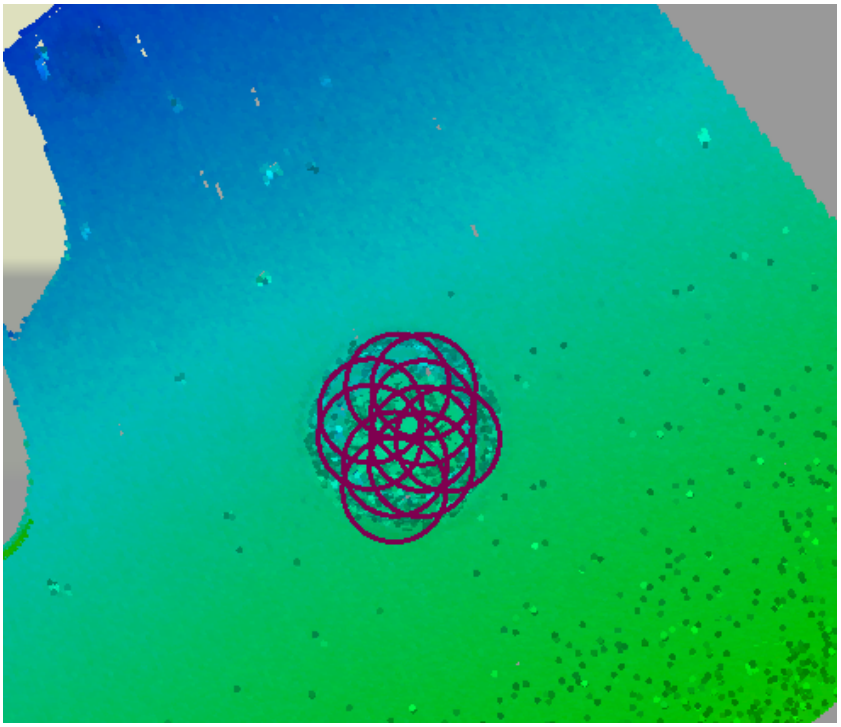

Figure 3.37.: An example of a point cloud depicting road segment with visible buffers around the most dense points presented in Figure 3.36

However, to retrieve the centre of a polygon the so-called gravity centre is extracted. This point in space, however, can be shifted out of the centre of a manhole by some centimetres due to the applied method.

Hence, the precise location of the centre of a manhole has to be further specified. The search allows a buffer around the point of interest having a following diameter: *Diameter of the manhole + diameter of the stucco part + pixel size as a possible deviations*. Patches within this area serve as the ultimate calculation of the centre point. Before this is conducted, the patches with less than 10 point per patch are filtered out as noise. The final centre is calculated based on the centre of gravity of patches (see Figure 3.38).

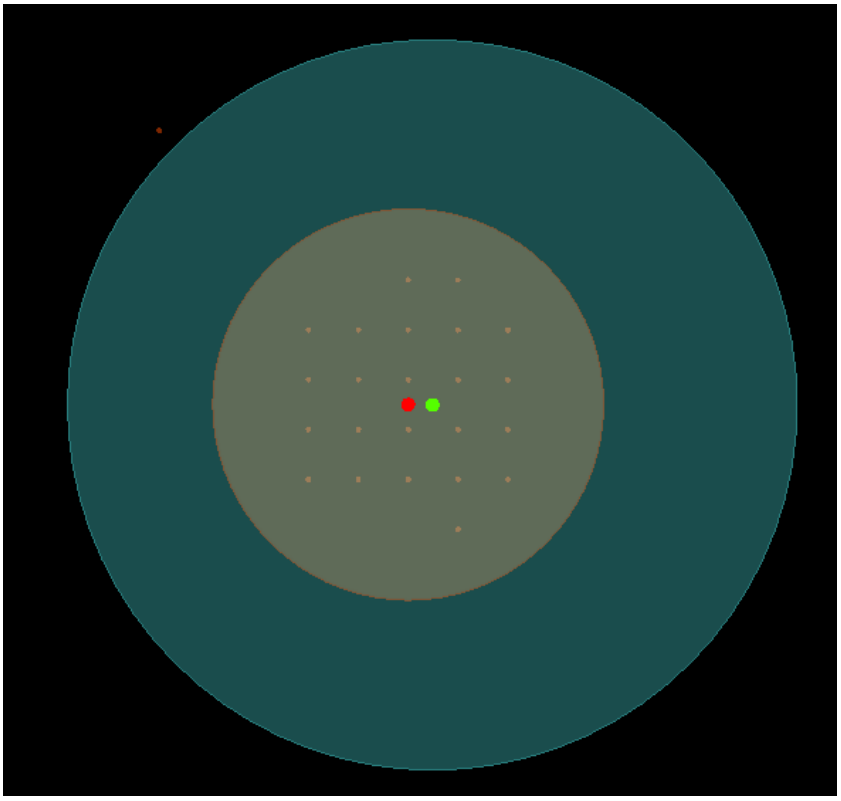

Figure 3.38.: A searching area of a manhole's exact centre. The first point of interest is marked by the green point, first buffer is marked with blue colour. The final centre point (red colour) is a gravity centre of patches within the second buffer (orange colour)

3.5.6. Creating a manhole

The manhole is created taking a buffer with a diameter of 0.785 m

The manhole is created based on a buffer with a diameter based on the respective manhole class (*Klasse D 400*) which is 0.785 m. Then, the respective buffer clips out the geometry from a refined road segment (see Figure 3.39).

As the output format for manholes is CityGML 3.0 the surface of a manhole is independent of a road segment. As a consequence, the road segment does not have missing faces in the area of a hole.

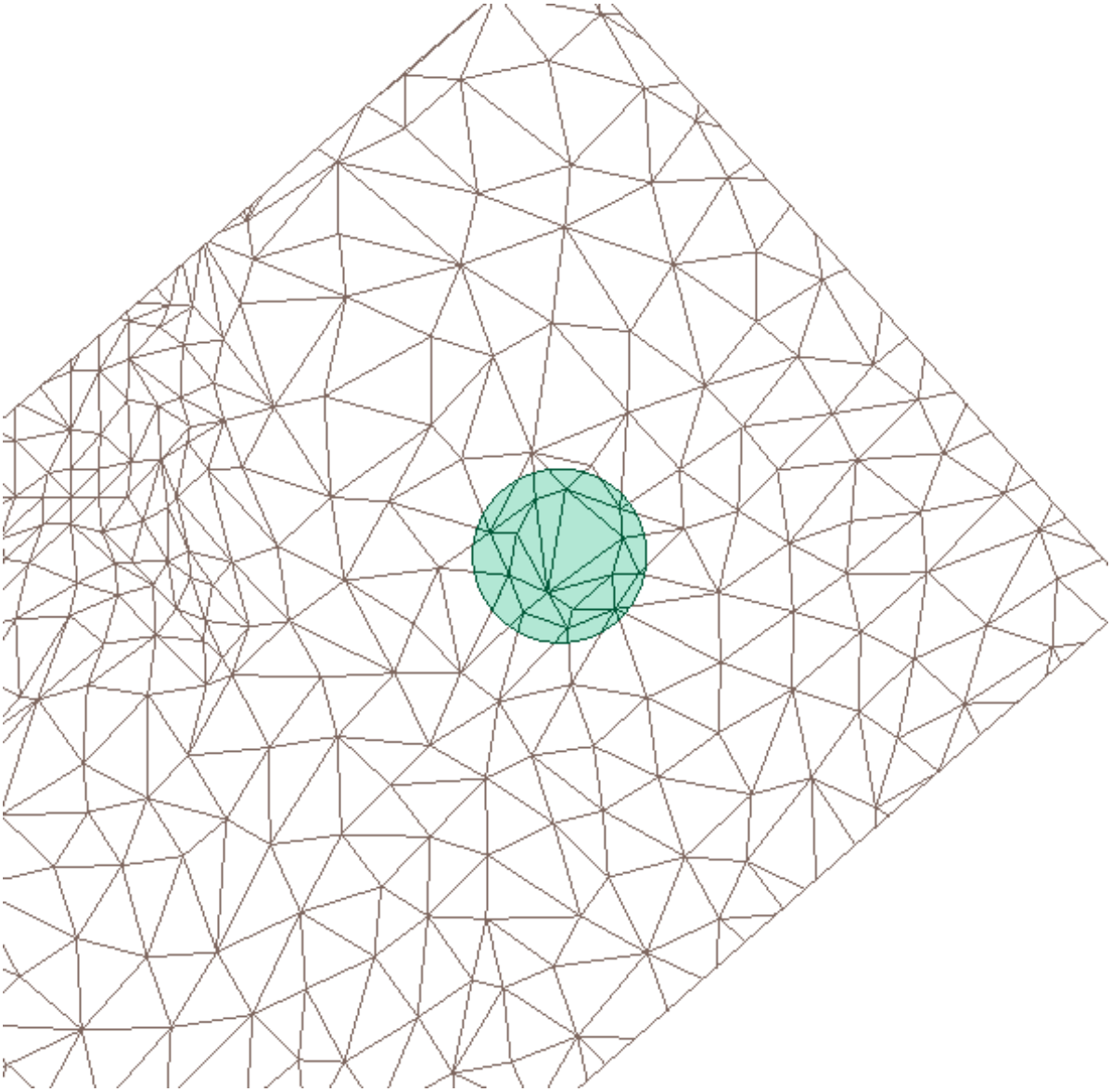

Figure 3.39.: A refined geometry of a road segment (white polygons) and a cut out geometry of a manhole (green polygons)

3.5.7. Manholes as CityGML 3.0

As mentioned previously, CityGML 3.0 is a novel approach which has a tailored module for transportation features modelling. The revised CityGML standard allows to explicitly represent manholes as a class *Hole* (holding semantics) and HoleSurface which is designed to represent the geometry of the manhole. This approach is introduced within the scope of this work (see an example in Figure 3.40).

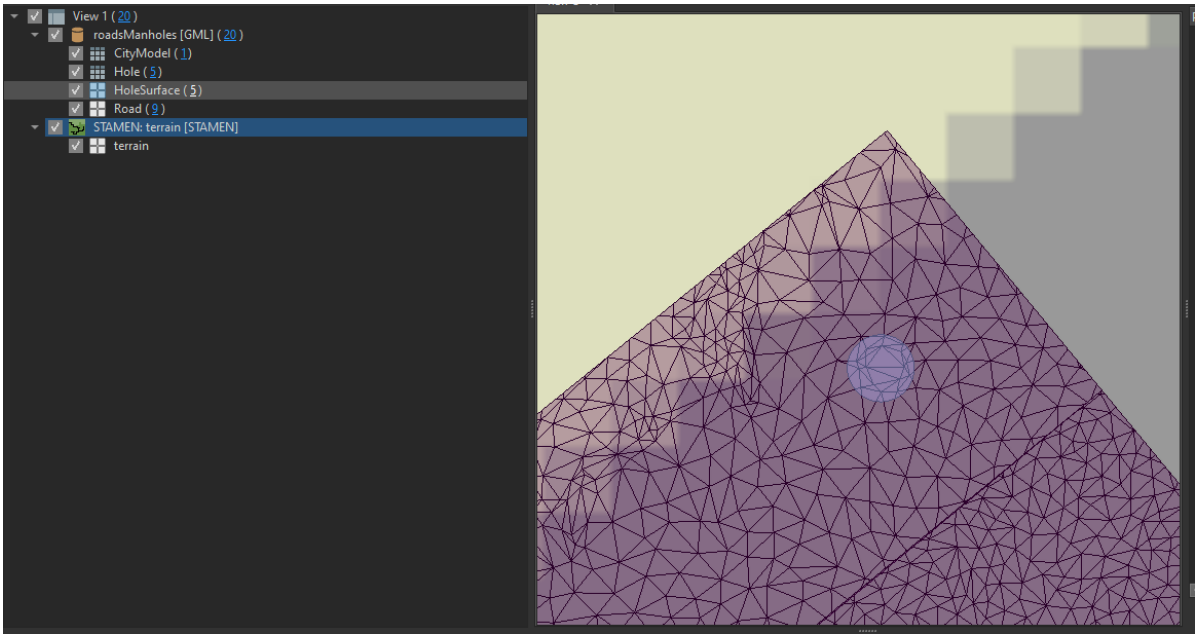

Figure 3.40.: *Road*, *Hole*, and *HoleSurface* as one CityGML 3.0 model

3.6. The visualisation of results

The ultimate results visualisation of created datasets can be performed in various tools. This is achieved through standardisation of city models' refined geometries to the open CityGML standard. Within the scope of this workflow, the Unreal Engine and 3DCityDB-Web-Map-Client are used as visualisation tools. The first is a game engine used as a backbone of state-of-the-art simulators like CARLA while the latter is a web-based tool tailored to visualise 3D city models.

One of the possible workflows is presented in the next chapter which also stands as a validation process of the methodology.

4. Evaluation & Performance

4.1. Visual inspection & performance assessment

The qualitative visual inspection is performed within the FME Inspector tool. This allows for a preliminary assessment of the obtained results. The main focus of the preliminary assessment is to check whether refinement is applied to all expected objects' segments of objects in the area of interests. This could be checked not only visually but also by querying attributes like *HasGeoRefined*. Moreover, it is possible to check whether the LoD of refined objects satisfies expectations. Furthermore, already in FME Inspector, one can explore the output data w.r.t. to the performance speed. If FME Inspector tends to lag, it can give the first impression that the data in game engines like Unreal Engine or GIS web applications like 3DCityDB-Web-Map-Client can be hard to explore because of the speed of performance.

Besides validation of output files itself, it is also possible to compare less detailed objects with higher LoD representing the physical object. This can be easily done with the aid of FME Inspector. Thus, raw vector objects can be utilised as a reference dataset. However, it has to be underlined that each of datasets has its pros and cons mentioned before in Chapter 2. Therefore, during the validation one has to keep in mind that vector objects inherit inaccuracies and errors. For example, although LoD3 is a reference vector dataset with the highest available level of details, it can have random errors resulting from a manual process of creating those models.

Furthermore, the performance assessment can be carried out not only w.r.t. to output geometries but also w.r.t. to the processing time of a workflow with different settings like fidelity of the output file, number of processing objects etc.

For the purpose of this work, all tests were performed at simplification set to 0.01 % as this process reduces the number of polygons but does not alter the shape significantly. It has to be emphasised that the parameter of an octree depth was tested at levels 8, 10, 12 as suggested by creators of Screened-Poisson algorithm [Kazhdan, Bolitho, & Hoppe, 2006].

As a result of those tests default parameters has been chosen and applied to the workflow. Nevertheless, final parameter values can vary depending on an application and user-specific requirements. Therefore, default values presented in the following sections should serve as a reference for a future researcher.

A high achievable fidelity value obtained with this workflow exposes a promising direction for future, possible improvements to detection algorithms. Structures visible to a naked eye of an operator are available for detection and consequently an extraction. An example of this process is investigated within the scope of this work.

Tests were performed on the processor Intel Core i7-8750H CPU @ 2.20GHz 2.21 GHz and with working memory (RAM) 16.0 GB, on Windows 10 operating system.

4.1.1. Horizontal-like objects

Generally, 94 road segments in the city centre of Ingolstadt were refined within the scope of this test scenario. Fine-tuned parameters were set to 10th level of the octree depth, 0.01 % simplification and manholes corrections were enforced.

The workflow presented in this thesis enforces the borders of segments to save a topological relation between adjacent segments. This assures that refined objects do not overlap which is an important factor in the creation of city models. However, this puts a limitation on a final result as it does not allow to perform geometrical corrections in terms of the segment's extent. If an HD Map is used this does not stand as a problem since HD Maps do not allow for gaps in the road segments. This produces a problem with incorporating other datasets like buffered roads of OpenStreetMap where a confidence level of a correct extent is low.

The combination of ALS and MLS point clouds as complementary datasets for a refinement process is an important part of this workflow. A scanner mounted on a car puts limitations w.r.t. a later point cloud coverage. A different acquisition geometry of ALS stands as ideal support of a MLS dataset. However, it has to be emphasised that in this test scenario (and in most cases) a ALS dataset was sparse in comparison to MLS data. Merging of those two datasets allows reconstructing full geometry of a road segment. In places where a vehicle is mapped a ALS can fill in a resulting hole in the MLS dataset. On the other hand, a hole is only filled when it is large enough to be covered by ALS (having one and more meters radius). However, those smaller holes in a dataset are filled in by scene reconstruction algorithm which interpolates those values. Reconstructed surfaces in places where the ALS filling is introduced result in a lower density of recreated surfaces (see Figure 4.1).

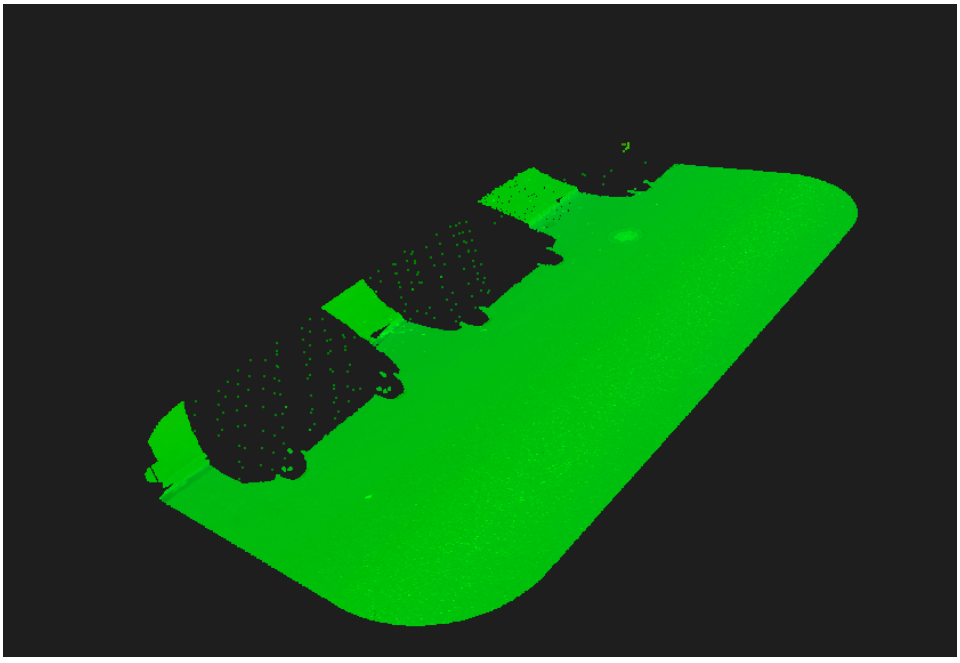

Figure 4.1.: Missing parts of an MLS dataset filled in with a sparse ALS point cloud

Accuracy assessment vs. raw data

In case of horizontal-like objects and specifically roads only raw geometry of the HD Map was available to be compared to. One of the important inputs of refined geometries is a superelevation of a road that is introduced (see Figure 4.2). Generally, OpenDRIVE has an option to model a superelevation but provided input raw geometries did not have it. Thus, it is believed that the presented solution can depict superelevation when it is not present in an OpenDRIVE dataset. A further application can be a verification of modelled superelevation within OpenDRIVE dataset.

Moreover, the presented methodology can depict small, irregular, laterally changes in a road area like potholes, ruts etc.

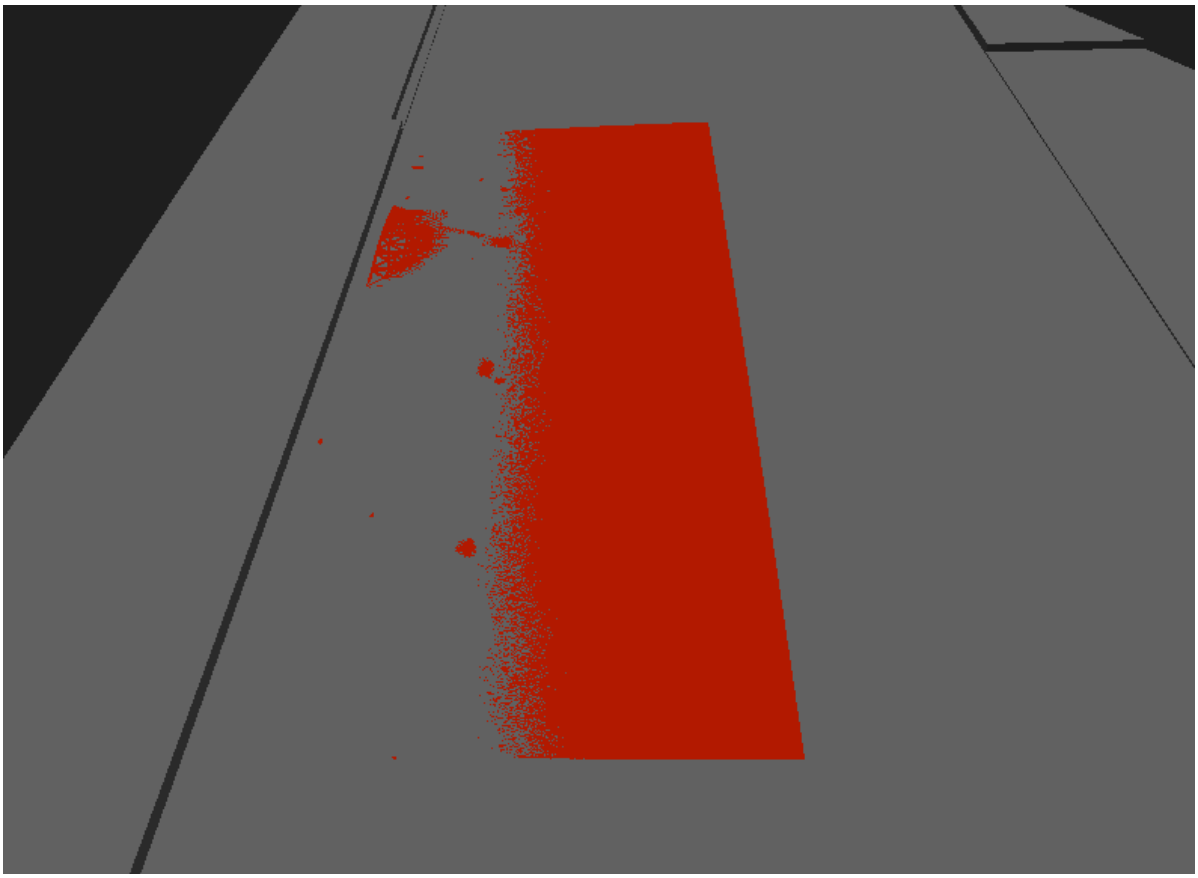

Figure 4.2.: A refined road surface (red-coloured) compared to a raw HD Map geometry. The superelevation of the road is well depicted while this information is missing in a provided HD Map test dataset

Evaluation of geometric fidelity

In order to understand the importance of octree depth levels for horizontal-like structures, several tests were performed. This parameter plays the most important role in a final fidelity

of a reconstructed geometry. The choice of which octree depth should be used depends highly on the refined geometries requirements. Based on tests performed, an octree level 10 was chosen as a fine-tuned parameter. This level meets the requirements of enhanced road shape representation and is easy to maintain and explore. While level 12' biggest advantage is its high degree of details, one has to take into account its large memory requirements - 120 MB for 94 segments, whereas the level 10 requires in such a case only 25 MB.

However, in case of very close investigation of a single object level 12 of octree depth is recommended as it is more detailed and in case of a single object easy to maintain and explore. All levels are shown in Figures 4.3 and 4.4 and 4.5.

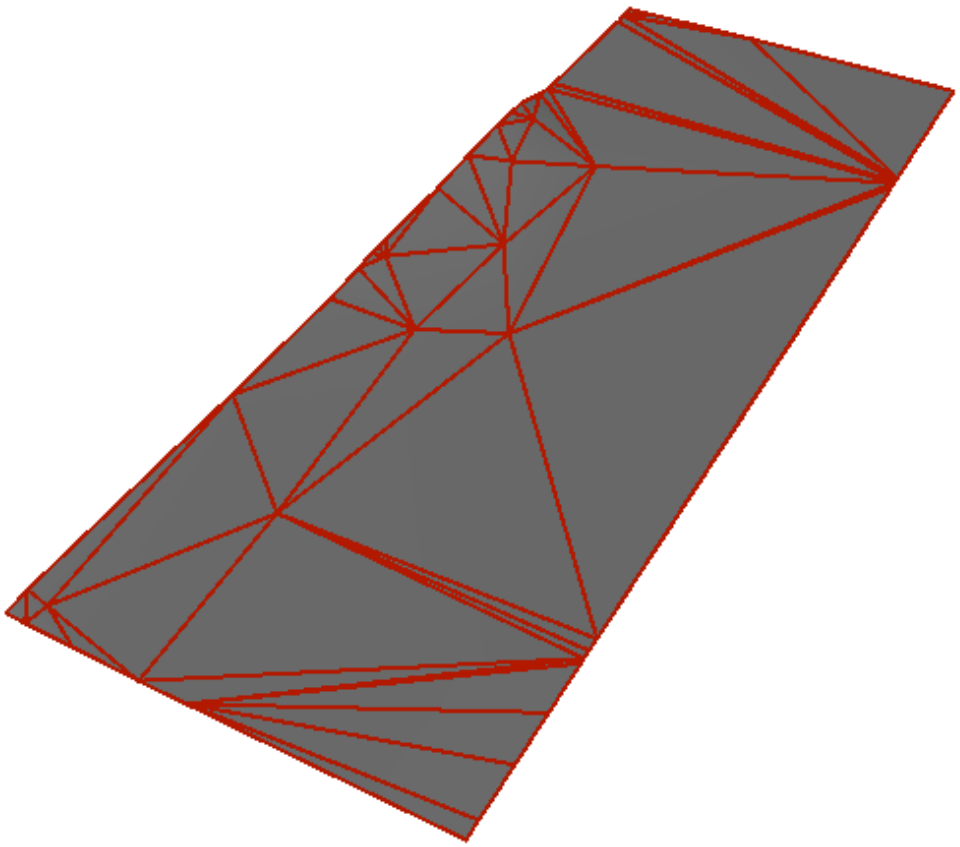

Figure 4.3.: A reconstructed road at octree level 8. Reconstructed surface consist of 36 faces

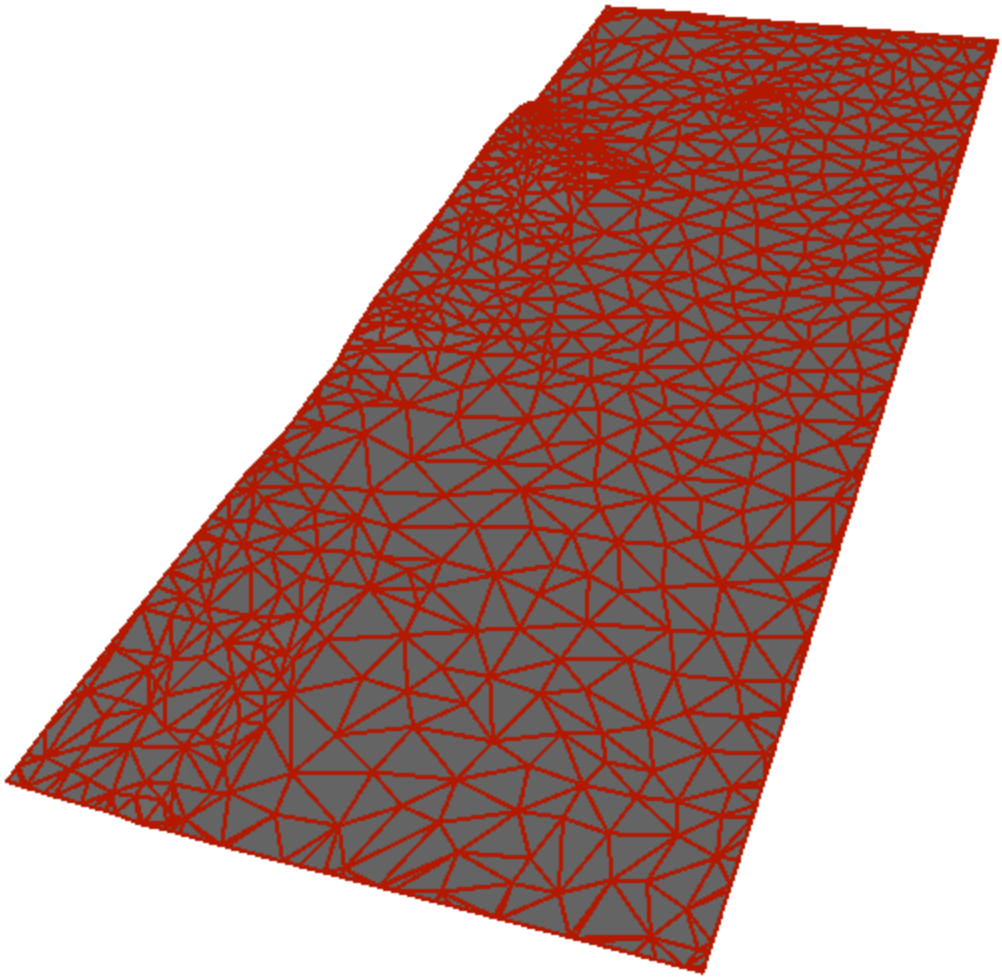

Figure 4.4.: A reconstructed road at octree level 10. Reconstructed surface consist of 1262 faces

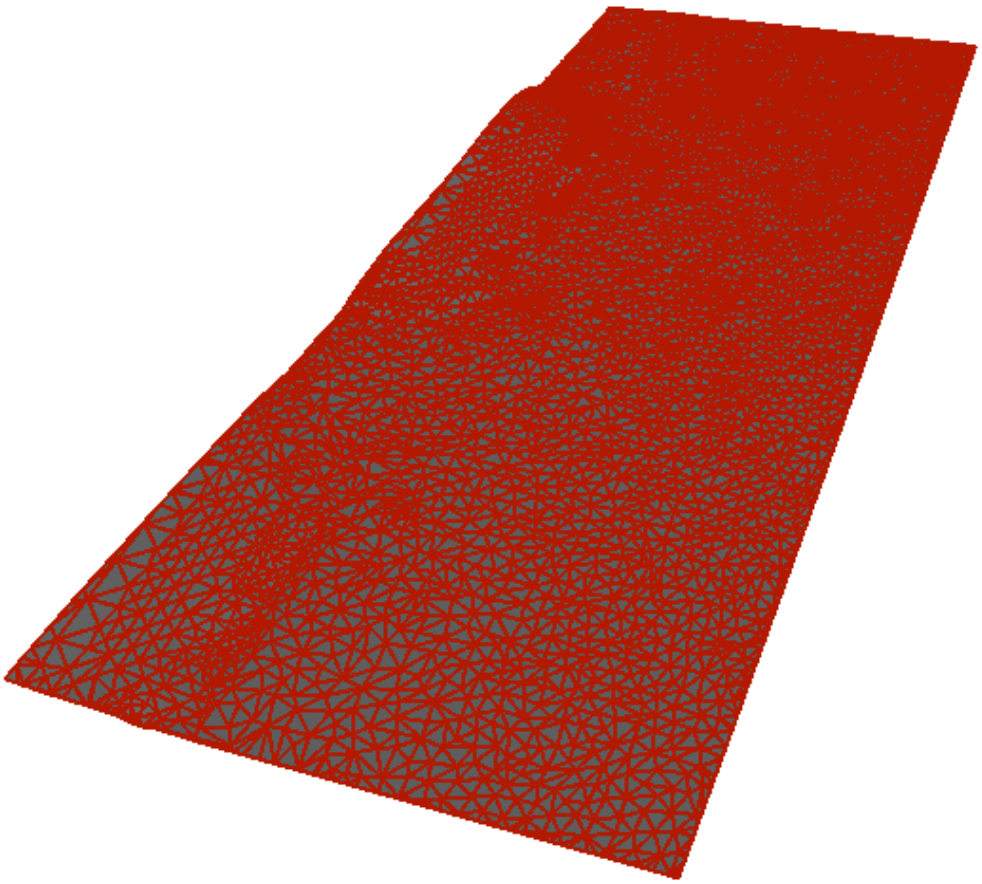

Figure 4.5.: A reconstructed road at octree level 12. Reconstructed surface consist of 6476 faces

Processing time assessment

The octree depth is also a parameter which vastly influences the processing time of the workflow. Thus, this factor has to be taken into account when using the tool. The processing time increases linearly together with a higher octree level (see Figure 4.6). The rule of thumb here is that two more levels result in twice and a half as much processing time (see Figure 4.6).

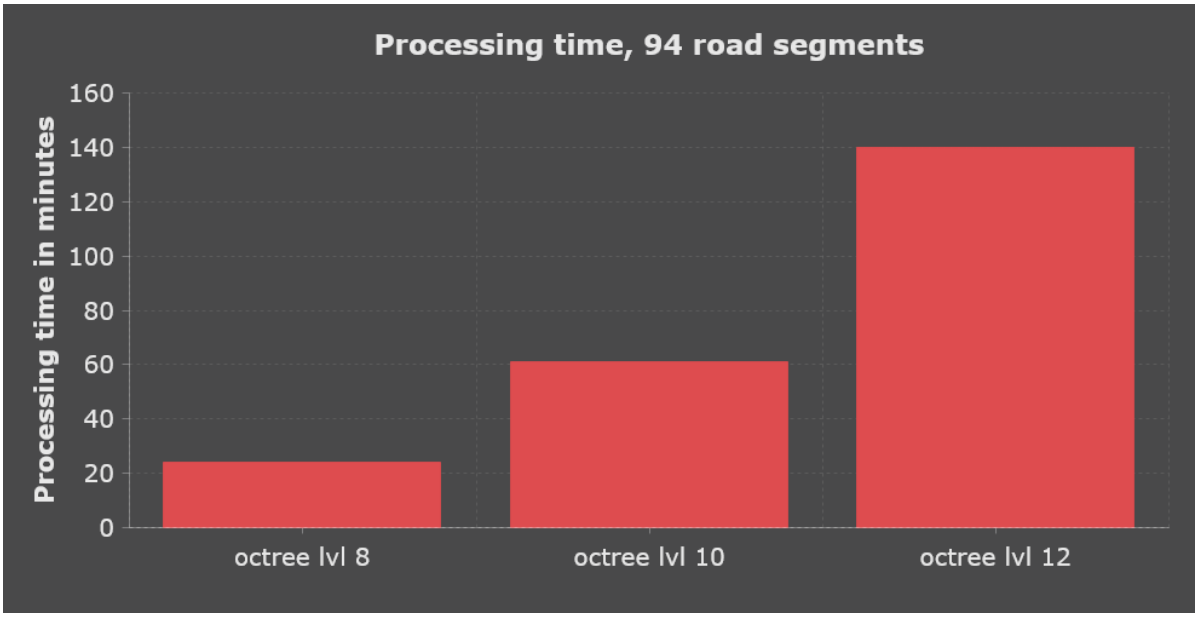

Figure 4.6.: A processing time at different octree levels for roads. 94 road segments refined within a city model

The processing time can be also assessed for a smaller scale. Two random road segments were chosen for this test (see Figure 4.7).

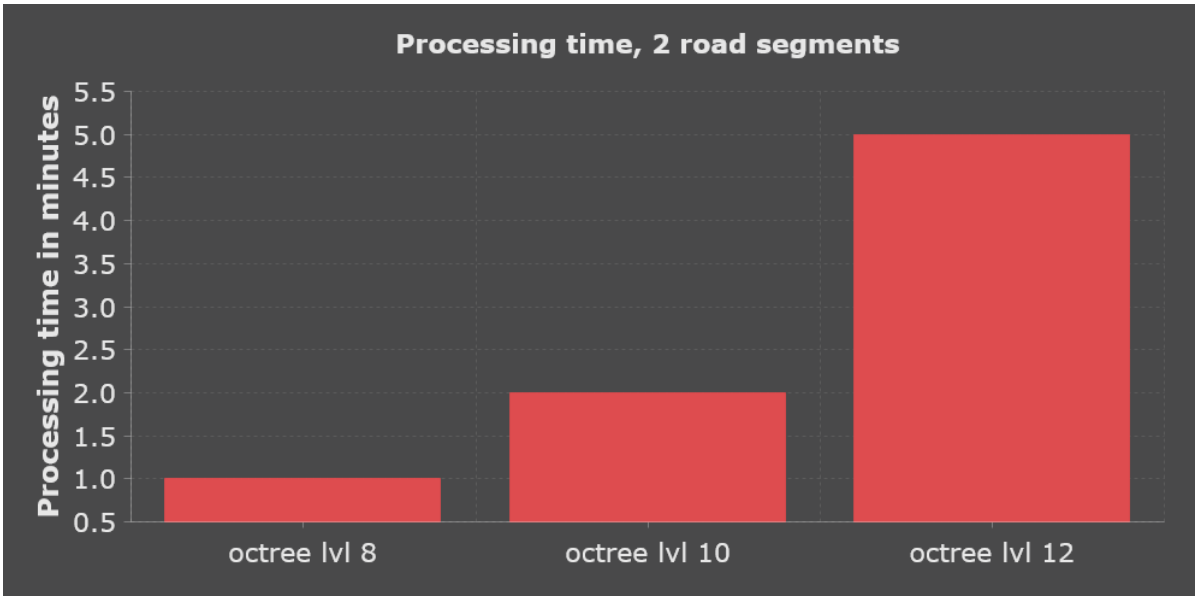

Figure 4.7.: A processing time at different octree levels for 2 road segments

4.1.2. Vertical-like objects

Generally, 69 buildings of total 87 in the city centre of Ingolstadt were refined within the scope of this test scenario. The fine-tuned parameters were set to level 10 of the octree depth and a 0.01 % simplification.

Similarly to the horizontal-like objects, borders of refined objects are enforced so that they do not augment the raw extent of objects. This puts on the same limitations and advantages as mentioned in the horizontal-like objects section. What has to be mentioned in this section is that this step also extents the processing time of the workflow since all buildings are 3D objects and in order to enforce rigid borders the cut has to be made in three dimensions.

Vertical-like objects do not consume ALS data for the refinement since the focus is put on walls' surfaces. Point clouds obtained during ALS campaigns present no significance for a detailed wall reconstruction due to their acquisition geometry.

The selection of only reconstruction-relevant walls is a step which reduces processing time but more importantly decides which walls can be reconstructed. Since this process is based on a fixed threshold there is a space for improvement. However, the fine-tuned threshold proposed in this thesis provided positive results in this testing field since all manually modelled buildings (besides 2 at the outskirts) not marked with *notSufficient* value of an attribute *DataAvailable* were reconstructed. This value of an attribute means that operator modelling LoD3 buildings recognised them as partially or not at all covered by a point cloud. What should be mentioned as an advantage of an automatic algorithm is that of one of the buildings labelled as not sufficient to creation for LoD3 was correctly reconstructed.

An alternative idea to select walls which are depicted by a mobile scanner at the street level would be to use vectors representing road's features (e.g. from OpenDRIVE or OpenStreetMap) and select walls which are facing roads [Wysocki & Albrecht, 2019]. However, this can result in false results as not all walls facing main roads are fully visible by other objects. Additionally, some backyards could be in a scanner range if there is no covering object.

The choice of reconstruction of only walls can be discussed. Since vertical structures like buildings consist of roofs, installations at roofs etc. However, the most important features for testing automated driving functions are objects that are within the range of sensors. Since MLS mounted on a car has gathered almost no spatial information about roof structures it can be assumed that this will not play a major role in future simulations and therefore could be discarded within the scope of this project. However, this is still unclear and here is a space for improvements for future investigators.

During the testing certain limitation of the implementation was encountered. The relatively big objects (550 m^2 and larger) sufficiently (see Chapter 3) covered by point clouds can be stopped by FME in batch processing. This is presumably due to high memory usage. Within the testing scenario for octree level 12 only 1 building resulted in such problem. The solution to this problem is to process selected objects separately. It is also believed that the higher computational power of a computer will be a solution to the issue. The template to marge separately reconstructed objects is provided.

Accuracy assessment vs. raw data

The comparison of LoD3 buildings with reconstructed walls is subjectively assessed as very good. However, expected limitations (mentioned before) could be observed in Figures 4.8, 4.10 and 4.12.

Wall's surface is cut to the extent of an LoD1 input object and thus does not cover V-shaped walls modelled in LoD2 datasets (see Figure 4.8). Higher-order details in terms of windows and walls depicted on a refined structure in comparison to an LoD2 structure is observed.

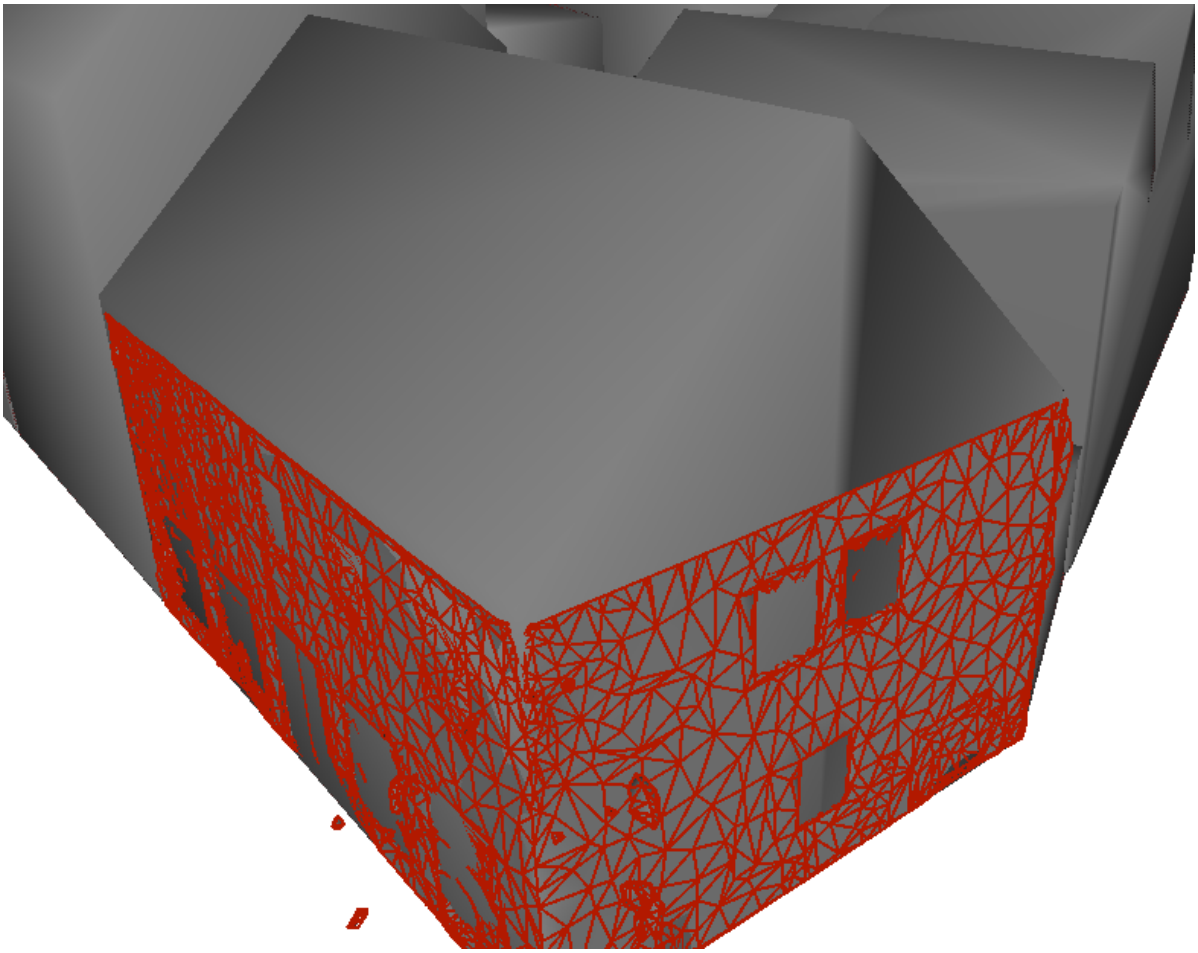

Figure 4.8.: A comparison of a reconstructed wall (red-coloured) with an LoD2 building model

The comparison of LoD3 buildings with reconstructed walls is subjectively assessed as very good keeping in mind the aforementioned limitations. It is possible to observe that a refined surface is more accurate since it reflects even small deviations in a wall surface. The recreated wall also represents windows and doors by a higher density of polygons and a concave shape. However, borders of windows and doors are ambiguous in the reconstructed surface and they are not in manually modelled LoD3 buildings (see Figure 4.9 and 4.10).

Nevertheless, it is believed that there exist automatic algorithms which can utilise the information contained in such a refined model [Swiss Federal Institute of Technology in Zurich, 2017].

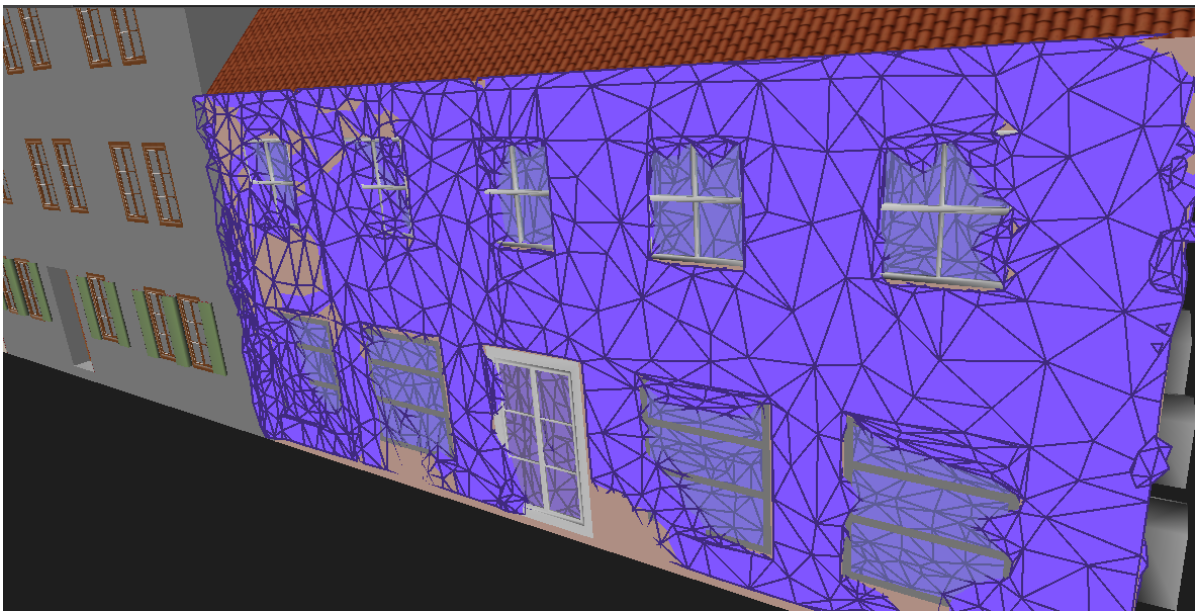

Figure 4.9.: A comparison of a reconstructed wall (blue-coloured) with features of an LoD3 building model

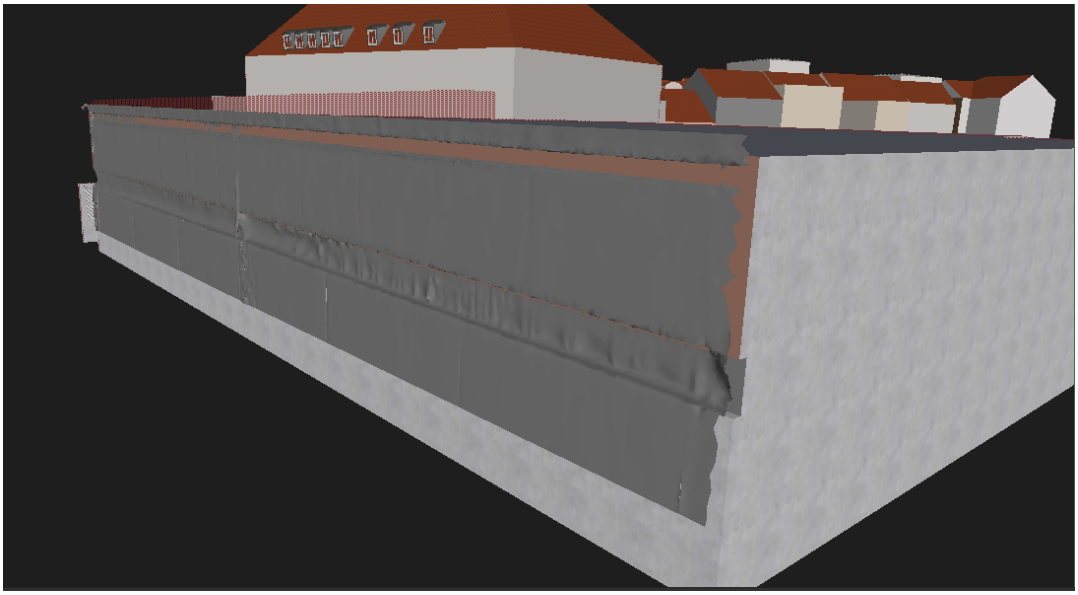

Figure 4.10.: A comparison of a reconstructed wall (dark grey colour) with an LoD3 building model

The LoD3 objects can also have certain additional building's features like balconies, stairs, or ornaments (see Figure 4.11). This could not be properly reconstructed since the RANSAC algorithm allows only for small deviations in a fitting plane. However, those gaps do not result in an empty space as they are interpolated by a scene reconstruction algorithm (see Figure 4.12).

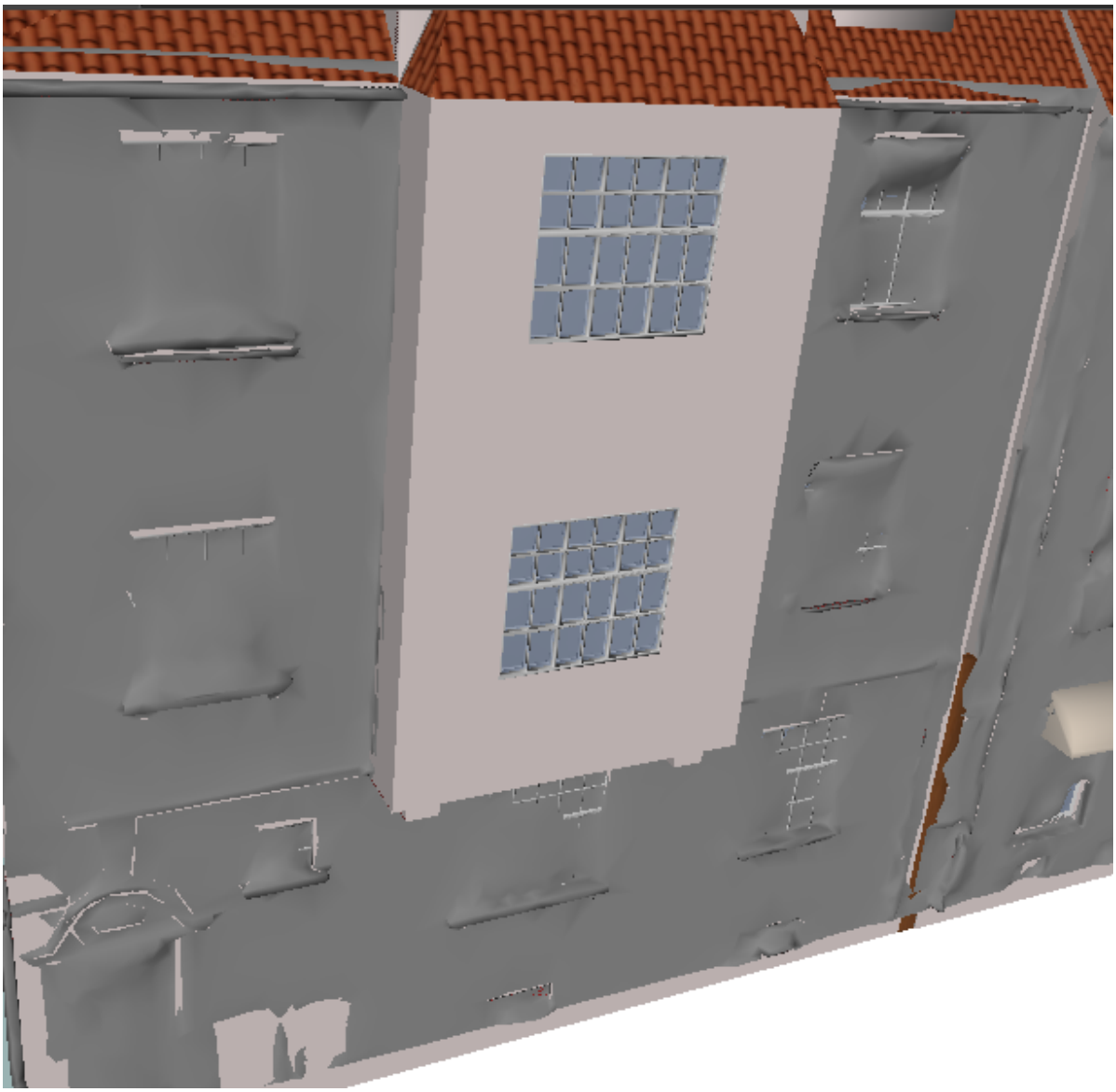

Figure 4.11.: A comparison of a reconstructed wall (dark grey colour) with an LoD3 building model (white, pink, blue colours and textures). The extruded wall modelled in an LoD3 by an operator is visible

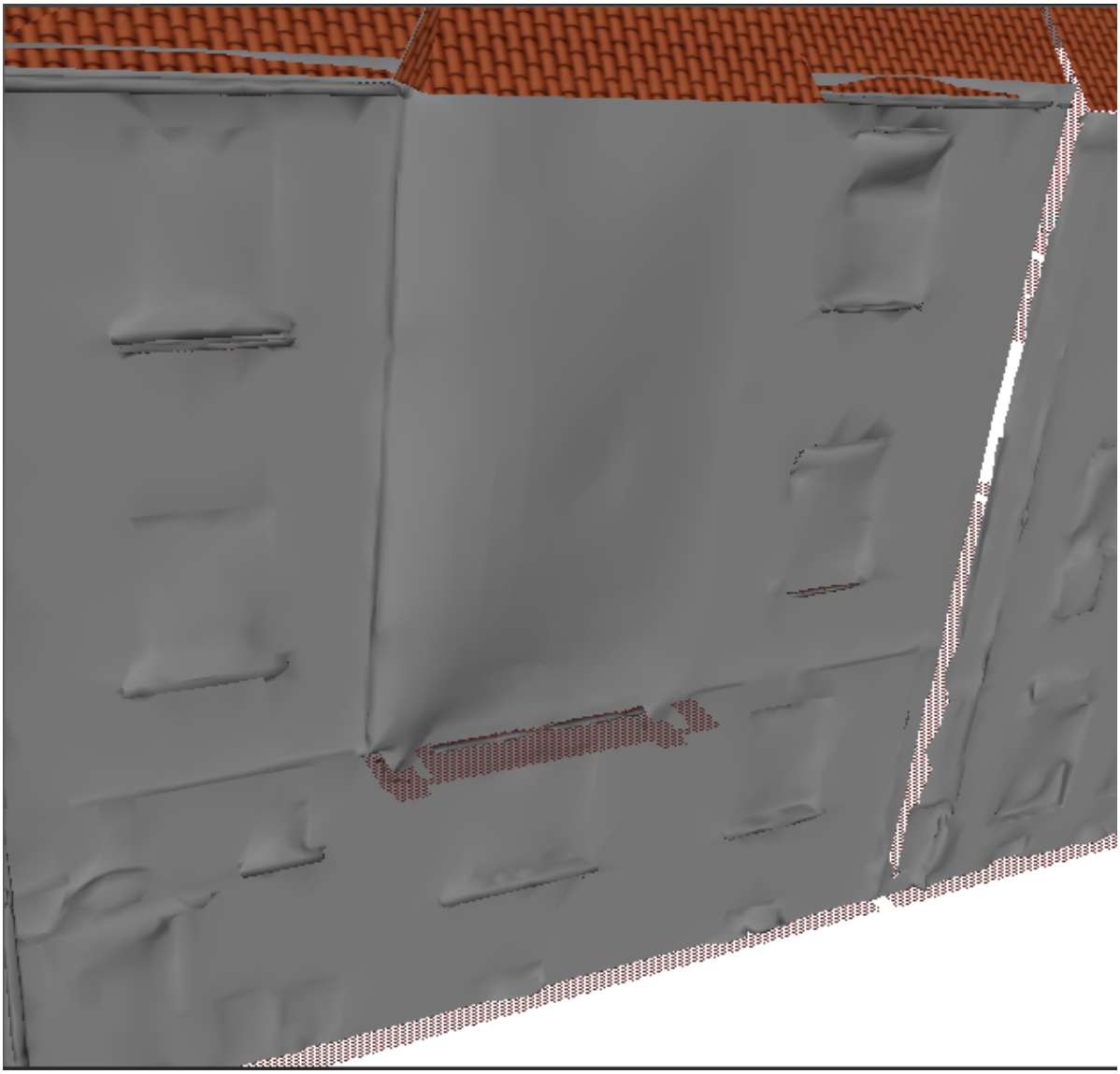

Figure 4.12.: A reconstructed wall's extrusion within a wall (dark grey colour) and a roof of LoD3 building model (textures). A visible slight wall's extension in the place of the extrusion

Some buildings in the dataset are located closely or even adjacently to branches of trees. Furthermore, traffic signs and other city furniture can be located adjacently to a building. This type of location can result in falsely segmented point cloud subsets. In other words, those objects can be classified as parts of a building structure. However, only if objects are adjacent to an object or the object is very close (within an accuracy range) and lies in line with one of the wall surfaces' plane margin (due to RANSAC fitting plane algorithm). This situation is shown in Figure 4.13.

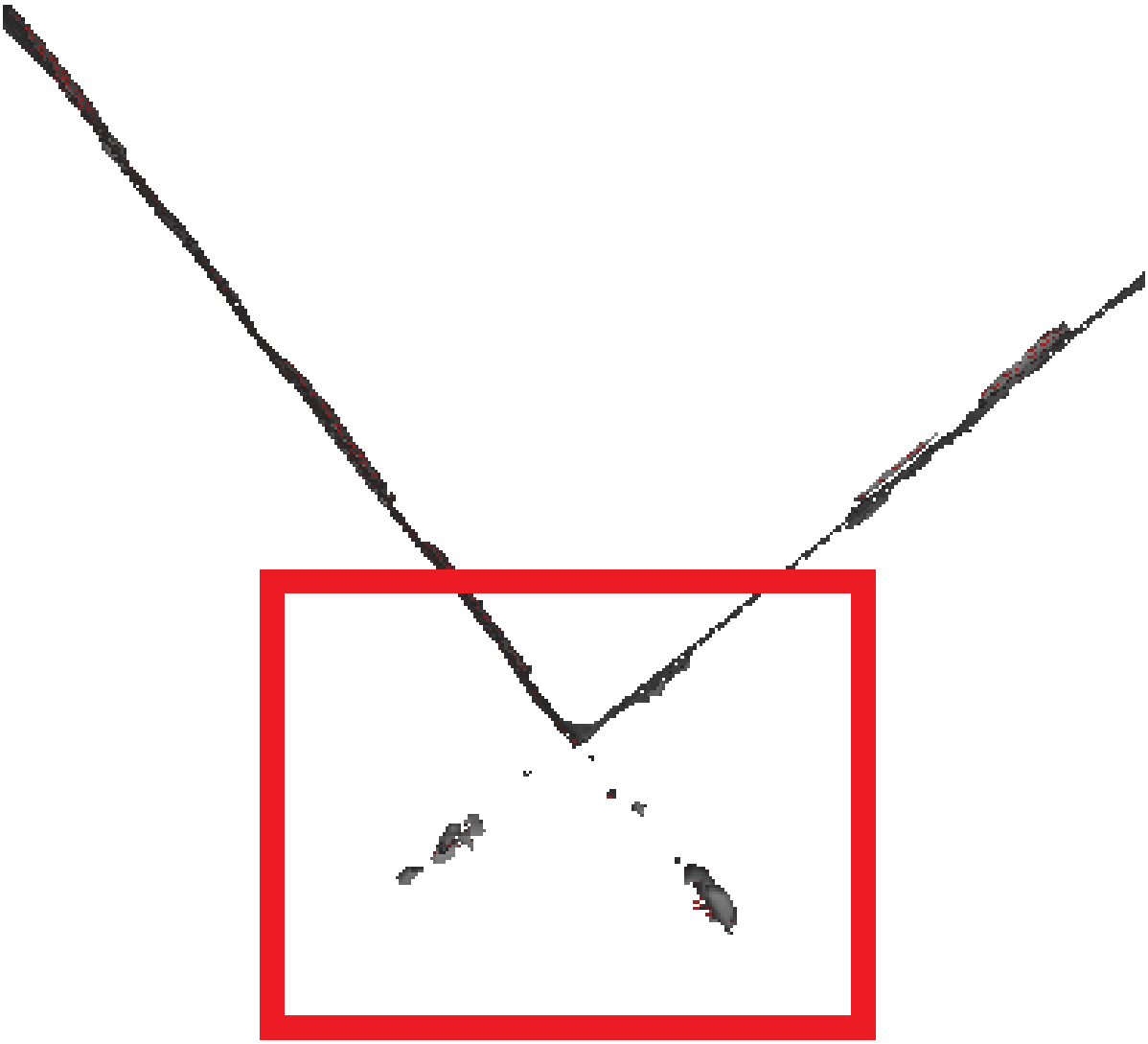

Figure 4.13.: A 2D view of reconstructed building's walls. Visible small parts reconstructed as buildings parts (within red rectangle) which in the reality describe tree branches

Evaluation of geometric fidelity

In order to understand the importance of octree depths levels for vertical-like structures, several tests were performed. The octree depth parameter plays the most important role in a final fidelity of reconstructed geometry. The choice of which octree depth should be used depends highly on requirements of refined geometries. Based on performed tests, the octree level 10 was chosen as a fine-tuned parameter. This level meets the requirements of an

enhanced building shape representation and is easy to maintain, explore. The level 12 stands as a more detailed representation however it is relatively large in terms of disk space (437 MB for 69 refined buildings, whereas the level 10 requires only 138 MB in such a scenario). Despite a large memory consumption, in case of very close investigation of a single object level 12 of octree depth is recommended as it is more detailed and in case of a single object easy to maintain and explore. All octree levels reconstructions are presented in Figures 4.14 and 4.15 and 4.16.

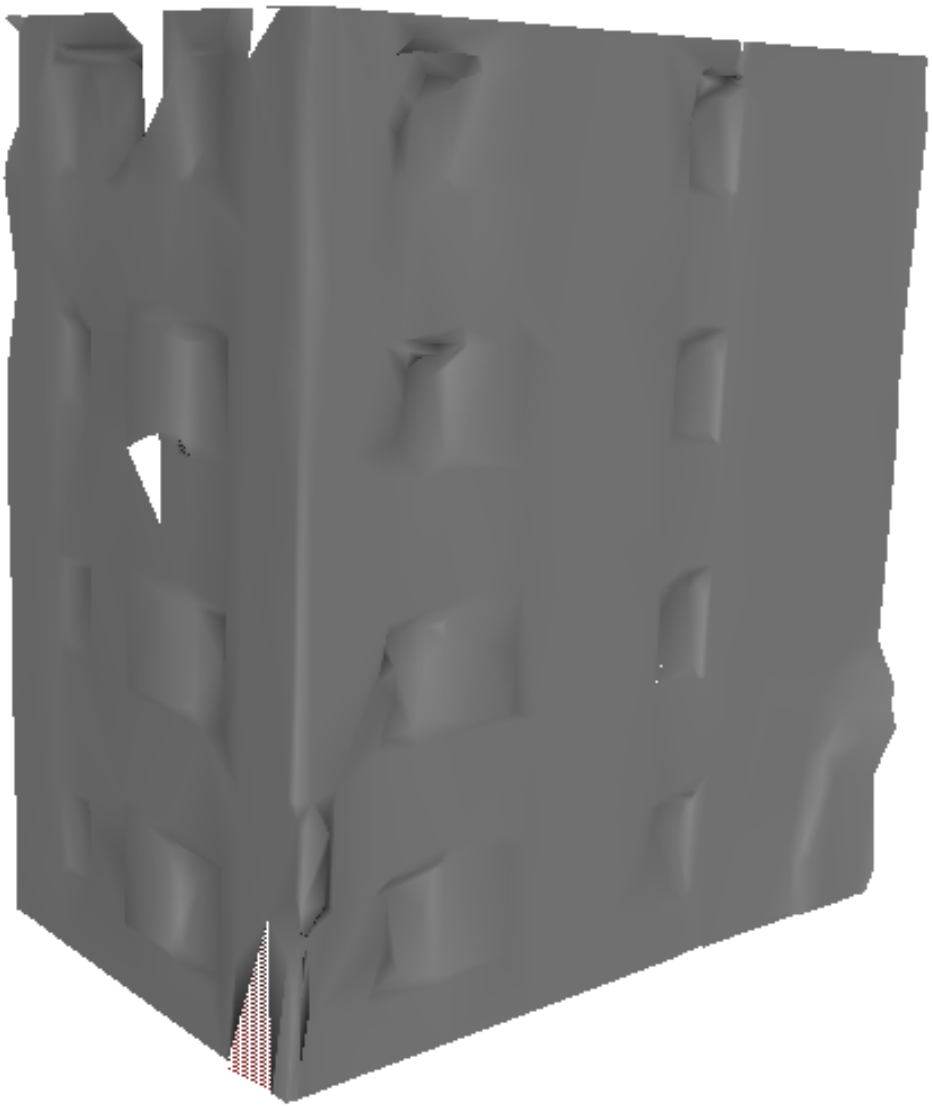

Figure 4.14.: A reconstructed building at octree level 8, 968 faces

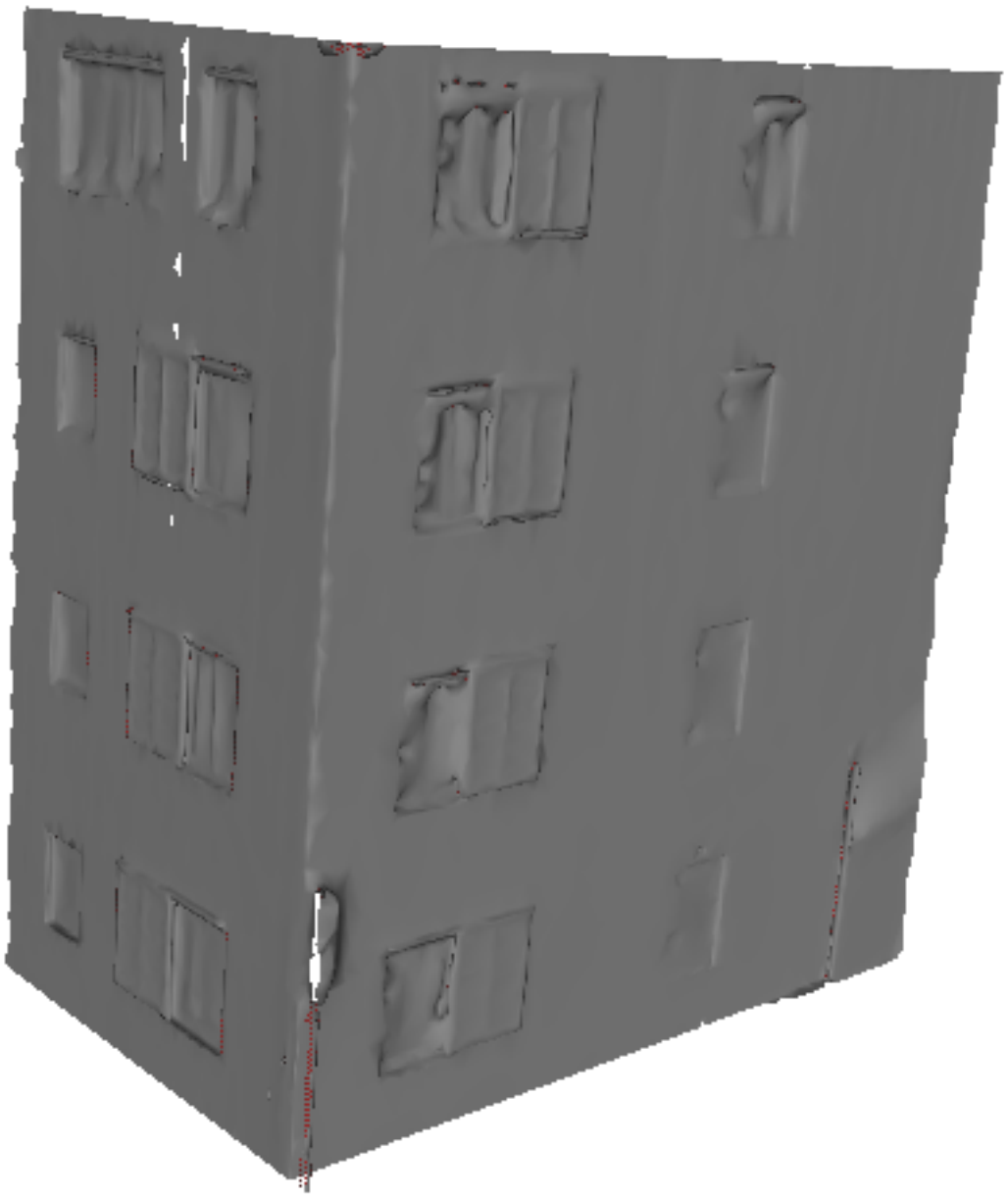

Figure 4.15.: A reconstructed building at octree level 10, 16 980 faces

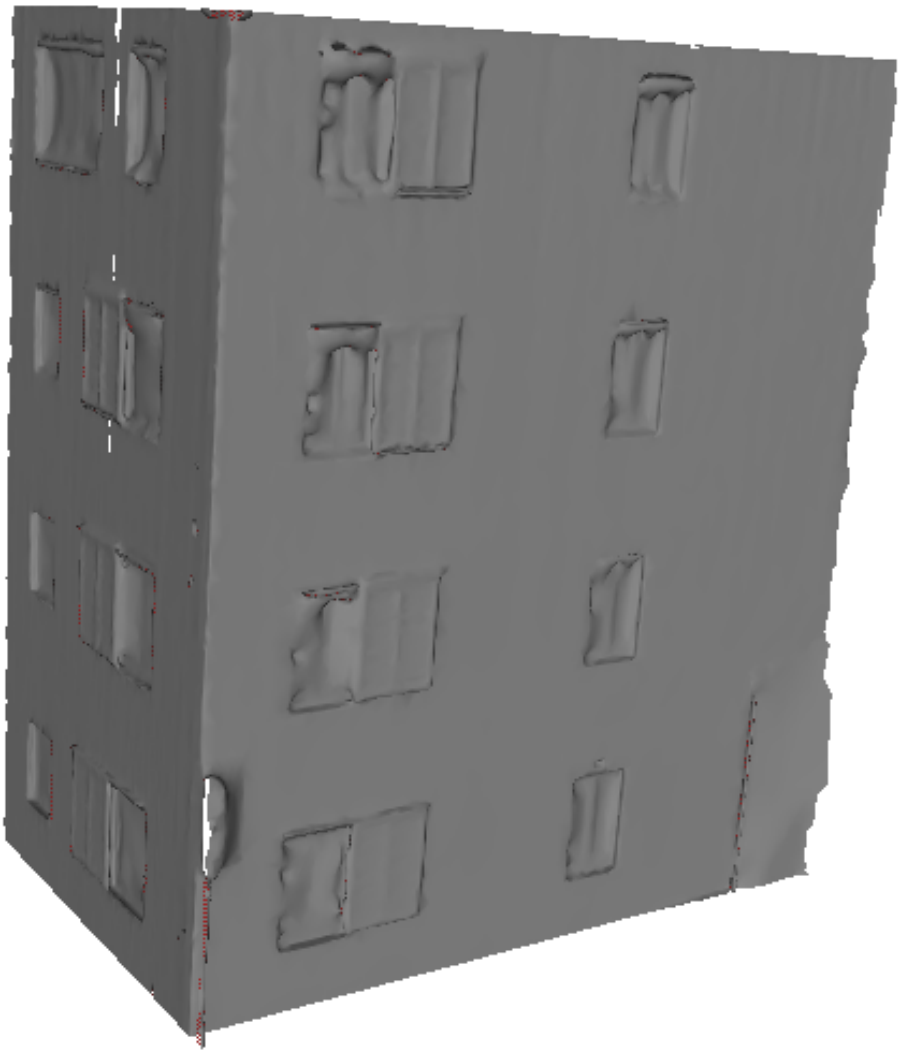

Figure 4.16.: A reconstructed building at octree level 12, 28 837 faces

Processing time assessment

A tested building was also analysed w.r.t. to processing time (see Figure 4.17). It has been concluded that the high complexity of a building and a point cloud in this area lead to long processing time. The processing speed tests were further performed on some randomly selected buildings at one octree level (see Figure 4.18) and on all tested buildings at different octree levels (see Figure 4.19).

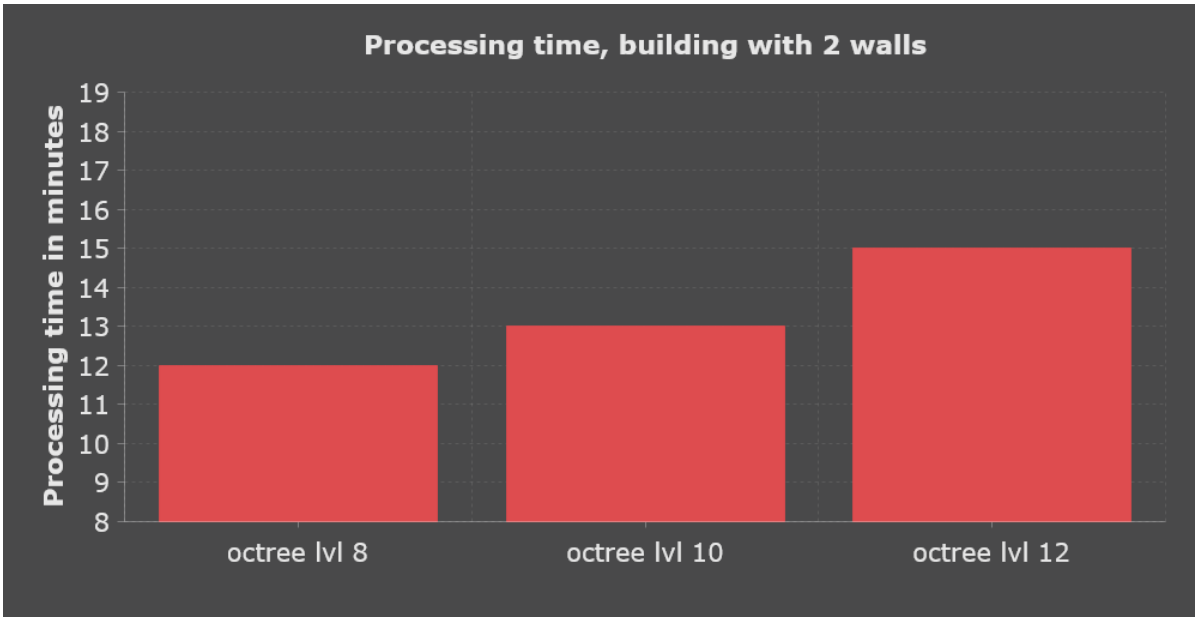

Figure 4.17.: Processing time at different octree levels for buildings

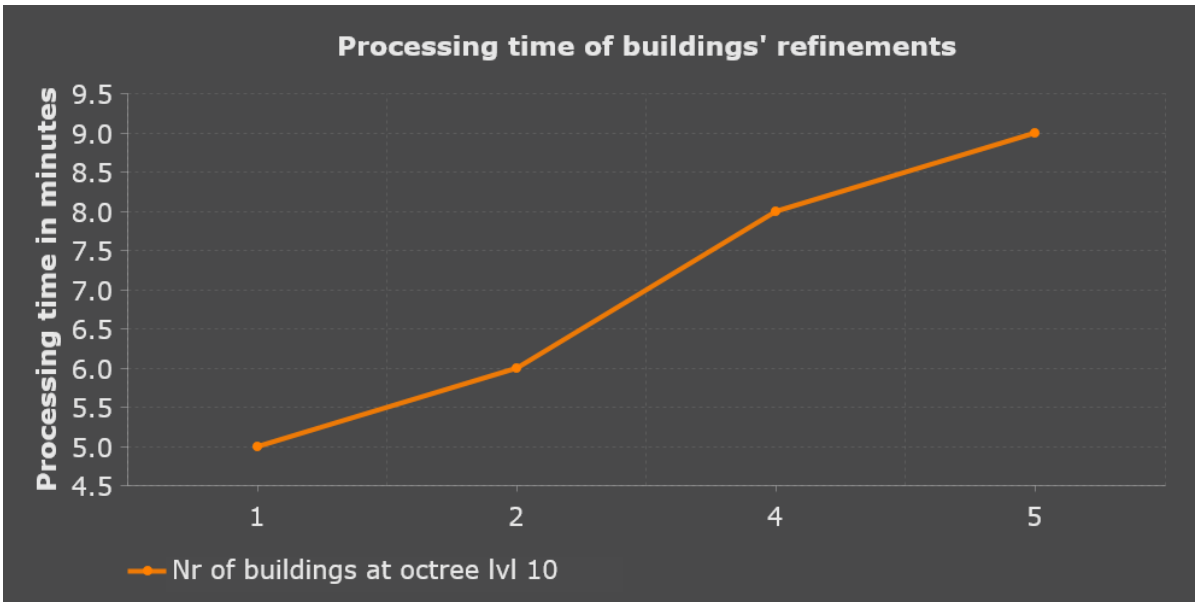

Figure 4.18.: Processing time at one octree level with a varying number of buildings

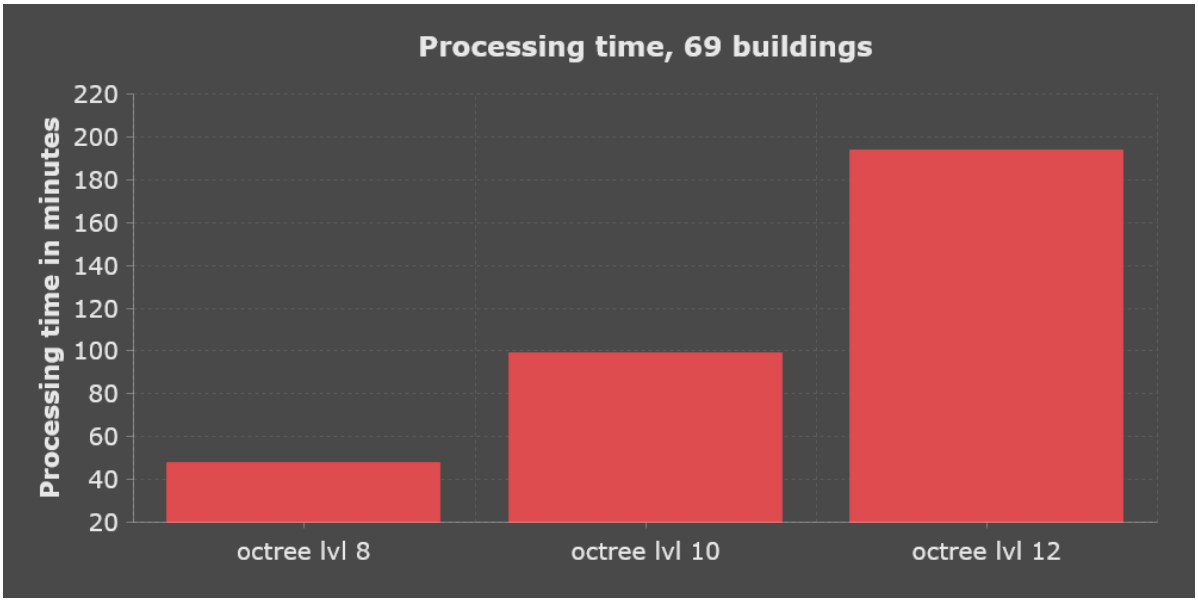

Figure 4.19.: Processing time at different octree levels for buildings. 69 Buildings refined with total 87 within a city model

4.1.3. Processing time comparison of vertical-like vs. horizontal-like objects

Processing time differences (see Figure 4.20) reflect variations between a workflow suitable for horizontal-like structures and the one for vertical-like. The group does not perform a fitting plane as well as the final cut to the raw geometry extent is performed in a 2.5D space. In contrast, the latter group performs a fitting plane search using RANSAC algorithm and performs a quasi 2.5 D space cut by inversions of coordinates. Furthermore, the complexity of a raw point cloud impacts a computational effort of the workflow. For example, in the case of roads, there is almost no point cloud depicting such complex structures like trees or windows.

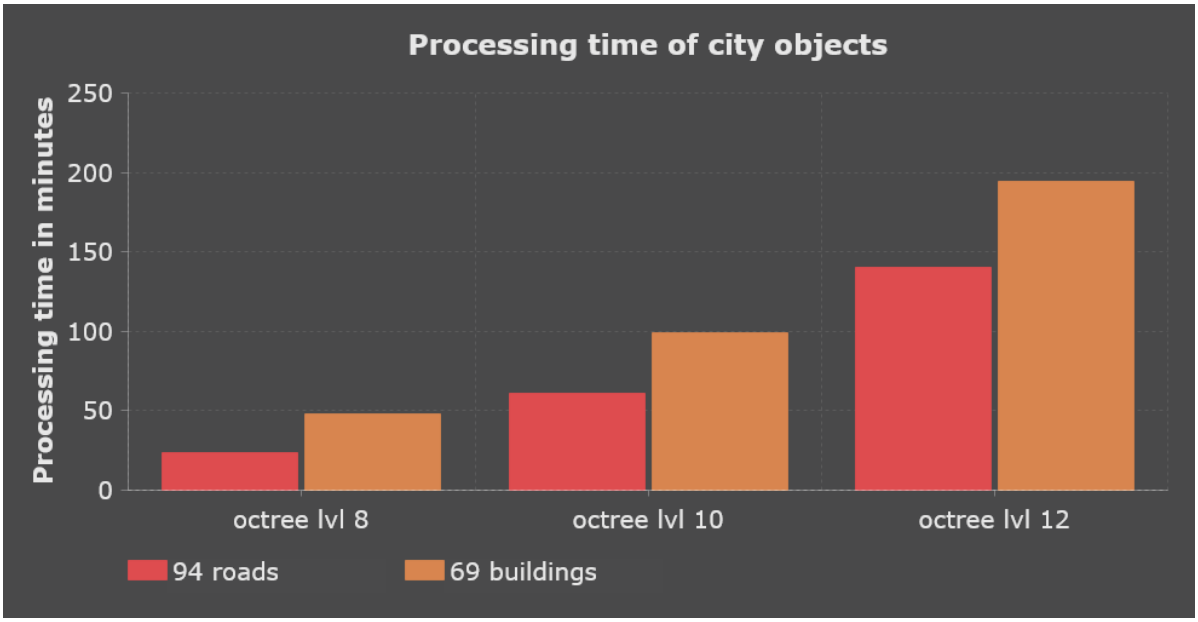

Figure 4.20.: Processing time at different octree levels. A comparison of roads and buildings

4.1.4. Syntax validation of city models

There are several possibilities to validate the syntax of city models. The FME software used in this work has an option to validate CityGML models while saving w.r.t. CityGML XML Schemas syntax. Furthermore, within this project an FME validation process and an Importer/Exporter function were used. This was performed within presented testing scenarios. Furthermore, besides the validation process also the function was used within this project. Based on this two-step verification the XML syntax was proved valid.

The KML and Epic Games Unreal Datasmith formats were checked automatically by FME while saving models to appropriate formats by FME. The final conformation of KML and Datasmith format were proved by a successful upload of created models to 3DCityDB-Web-Map-Client and Unreal Engine respectively.

The geometrical verification was conducted through the deployment of output models in different dataset formats and for applications mentioned before. Aforementioned validation was successful. However, this does not determine the final validity of created models - small nonconformities can be present.

4.1.5. Performance speed assessment & validation through the exploration of results

As stated before, FME Inspector can serve as a first-gist tool to understand the complexity of refined models in terms of an exploration of results. This, however, can be only measured subjectively by an operator. Level 8 and 10 of the octree depth allowed for a seamless exploration of results in FME Inspector.

3D City Database suite

For a second stage assessment, a 3D City Database (3DCityDB) suite was chosen. The database enables managing of city models in the CityGML standard. Furthermore, seamless database managing using an Importer/Exporter tool and exploration of results in a web browser are possible. The check was performed with the following assumption: if an import to the database is (relatively) fast and export is possible the created city model is stable and can be utilised in numerous applications.

Refined buildings were imported to the 3DCityDB with a high pace (12 s) and without errors (see Figure 4.21). The same operation was performed for horizontal-like objects and was successful too (see Figure 4.22).

```
[11:29:02 INFO] Resolving XLink references.
[11:29:02 INFO] Cleaning temporary cache.
[11:29:02 INFO] Imported city objects:
[11:29:02 INFO] bldg:Building: 87
[11:29:02 INFO] bldg:BuildingPart: 69
[11:29:02 INFO] Processed geometry objects: 726131
[11:29:02 INFO] Total import time: 12 s.
[11:29:02 INFO] Database import successfully finished.
```

Figure 4.21.: A summary of imported vertical-like objects to the 3DCityDB using an Importer/Exporter tool

```
[11:53:46 INFO] Resolving XLink references.
[11:53:46 INFO] Cleaning temporary cache.
[11:53:46 INFO] Imported city objects:
[11:53:46 INFO] tran:Road: 94
[11:53:46 INFO] Processed geometry objects: 135850
[11:53:46 INFO] Total import time: 02 s.
[11:53:46 INFO] Database import successfully finished.
```

Figure 4.22.: A summary of imported horizontal-like objects to the 3DCityDB using Importer/-Exporter tool

The export from the database backwards to a CityGML file was performed to check the conformation of the city models. The Importer/Exporter tool enables managing of the export through filtering by selected attributes. Here, the test was performed using a *gml:id* attribute for a randomly selected building (see Figure 4.23 and Figure 4.24).

This successful test confirmed that it is possible to utilise attributes introduced within a workflow. Thus, attributes like *HasGeoRefined* can be also used for better managing of 3D

models. For example, to query only refined buildings.

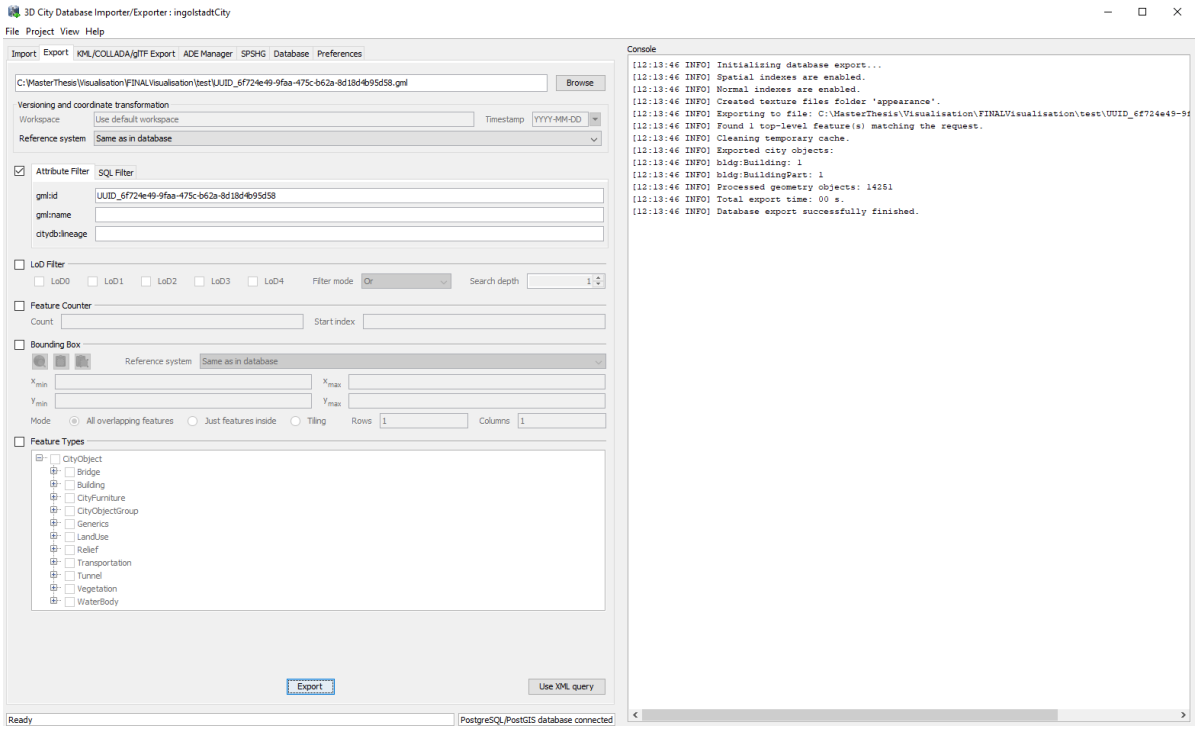

Figure 4.23.: A summary of exported object to the local disk in CityGML standard using the Importer/Exporter tool

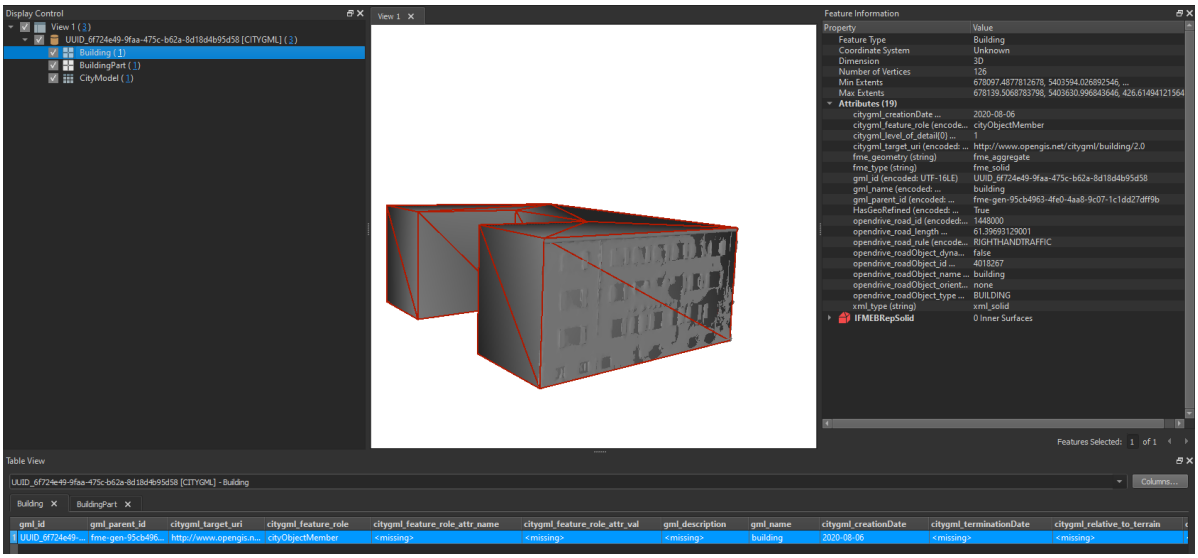

Figure 4.24.: An exported building in the FME Inspector tool. All raw semantic information kept and geometry preserved. The refined wall visible in the foreground

The geodata stored in the 3D City Database can be seamlessly exported to efficient formats supporting geo visualisations like KML, COLLADA and glTF. Those can be utilised in web browsers and tools like Cesium WebGL Virtual Globe or Google Earth Pro.

This export was performed within this test scenario to prove the interoperability of the created city models w.r.t. to geometry. Then visualisation in Google Earth as well as in 3DCityDB-Web-Map-Client was created.

The export to KML/COLLADA/glTF has taken 65 s for buildings and 12 s for roads. The export was successful, fast (especially for roads), and without errors. As the area of the Ingolstadt centre is flat and there might be elevation discrepancies w.r.t. 3D models, the objects were clamped to ground. Within the Importer/Exporter tool it is possible to add an appearance to exported models. In this case, buildings and roads have the same transparency of 200, whereas the first were assigned an orange and the latter a dark grey hue. The *Highlight when onMouseOver* option was enabled, that allows to highlight objects in Google Earth Pro (see Figure 4.25 and 4.26).

The pace of visualisation in Google Earth Pro was sufficient to explore more demanding models of buildings without and with toggled on-road objects. However, small lags in the exploration performance were noticeable. Not all roads were visualised instantly. Only hovering at one road at the time as possible in order to show the geometrical structure of a road object.

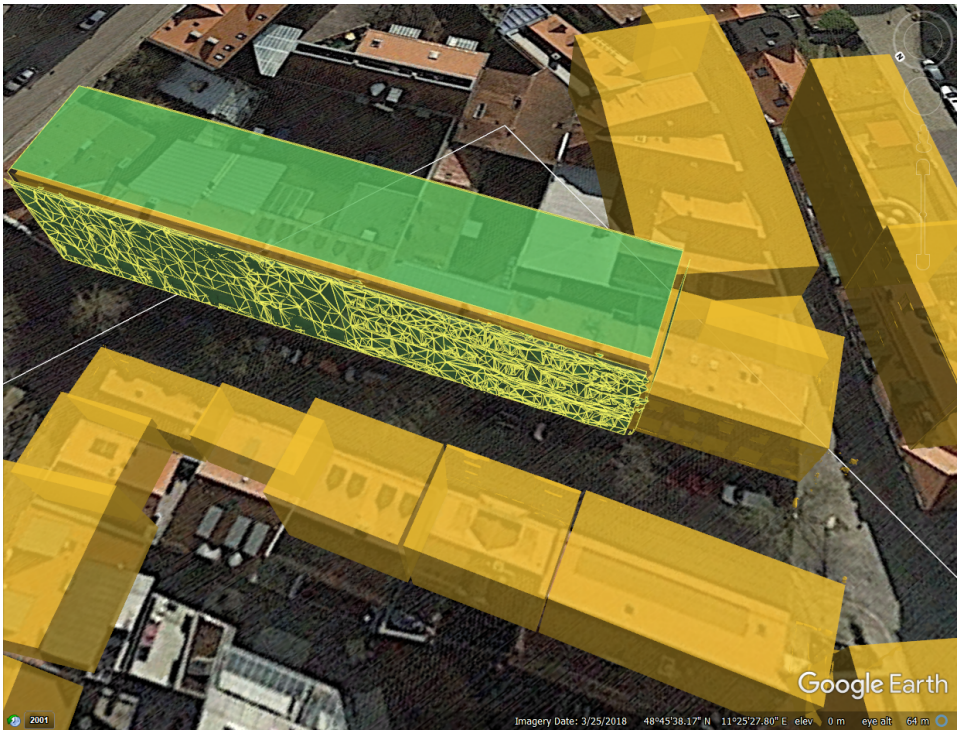

Figure 4.25.: The exported buildings in Google Earth Pro. The green highlight shows a picked object and its refined structure (yellow edges)

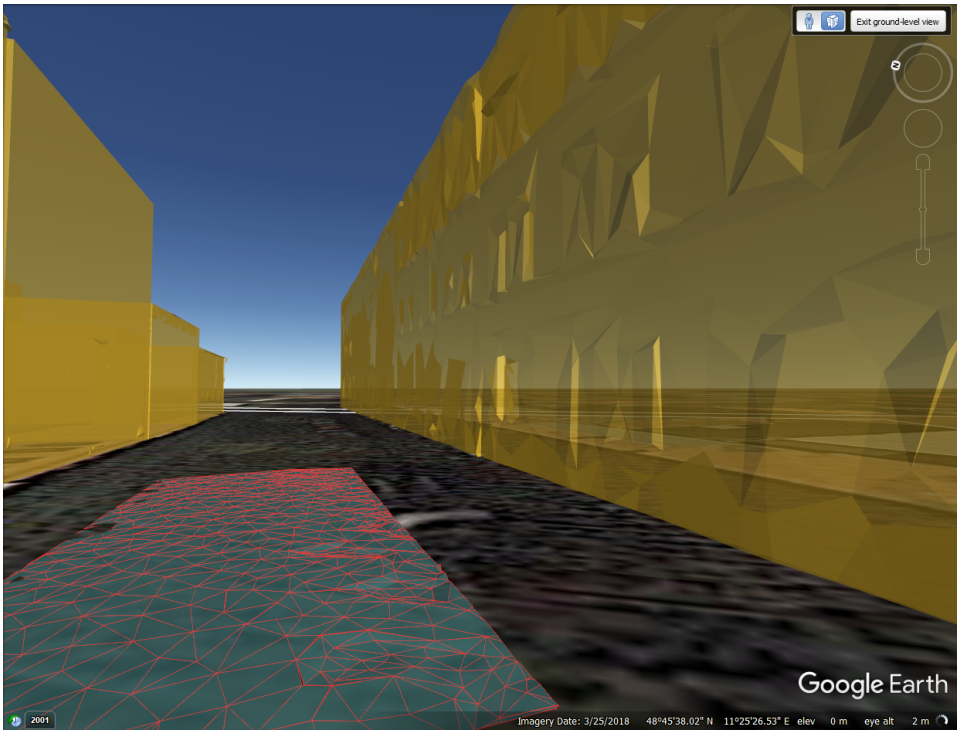

Figure 4.26.: The exported buildings and roads in Google Earth Pro

Those problems were caused by switched on setting in the Importer/Exporter tool allowing to highlight objects while hovering in Google Earth Pro (*Highlight when onMouseOver*). If this option had been disabled, export time would have been significantly shorter - 9 s for buildings and 2 s for roads. Moreover, no lags were noticed and all features were visible instantly. A downside of the solution is that objects cannot be highlighted and thus refined structures were barely visible. The remaining settings were not changed except roads' colour - from dark grey to red (see Figure 4.27 and 4.28).

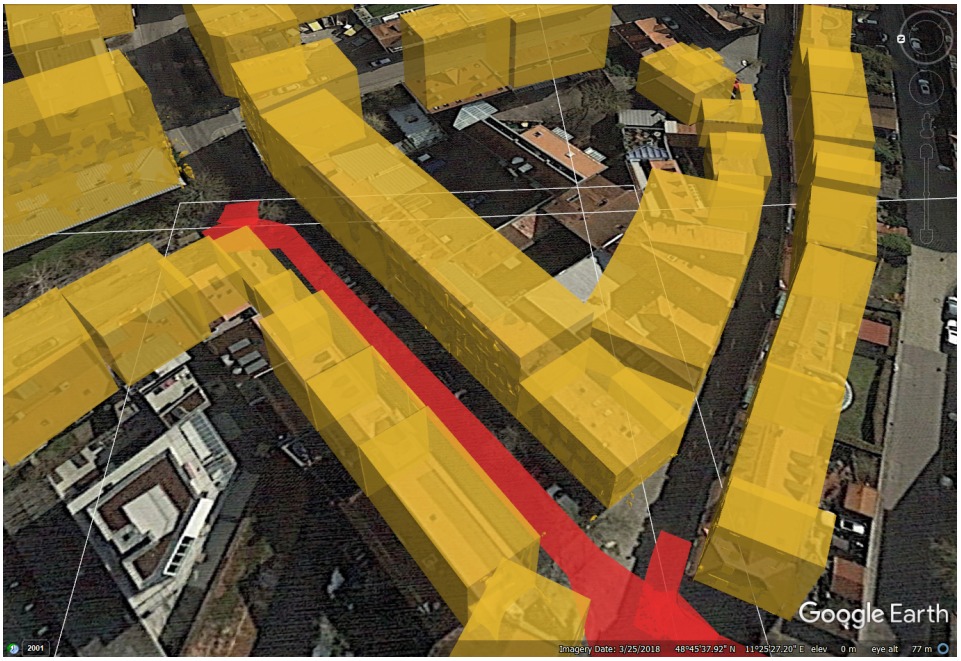

Figure 4.27.: The exported buildings and roads in Google Earth Pro with a disabled highlight while hovering

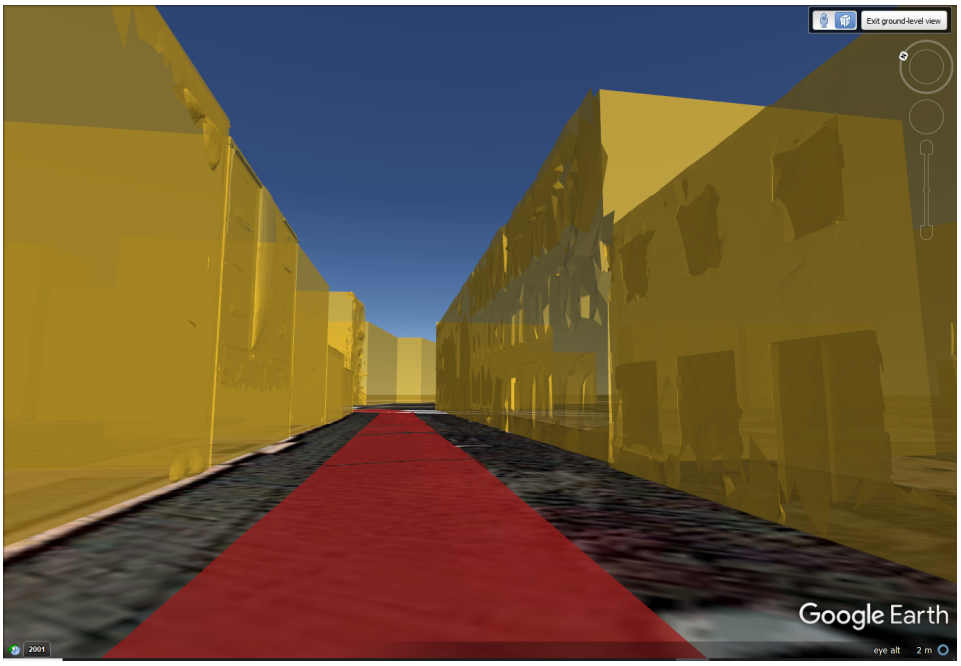

Figure 4.28.: The exported buildings and roads in Google Earth Pro with a disabled highlight while hovering. View from the street level

Google Earth Pro is not the only tool that allows for seamless data exploration. In the course of this work, a 3D web client proved to be suitable for visualization of city models. In order to test whether exported city models could be deployed on 3DCityDB-Web-Map-Client, a local server and Cesium Virtual Globe framework were set up. Under the requirement that geometries of roads and buildings were deployed on the web browser without errors, the test was successful.

In order to create layers, additional glTF objects were created. This step significantly increases model exploration performance in the web browser. Though, the export time from Importer/Exporter tool increased to 12 s and 3 s for buildings and roads respectively. This still assured swift export and deployment of models.

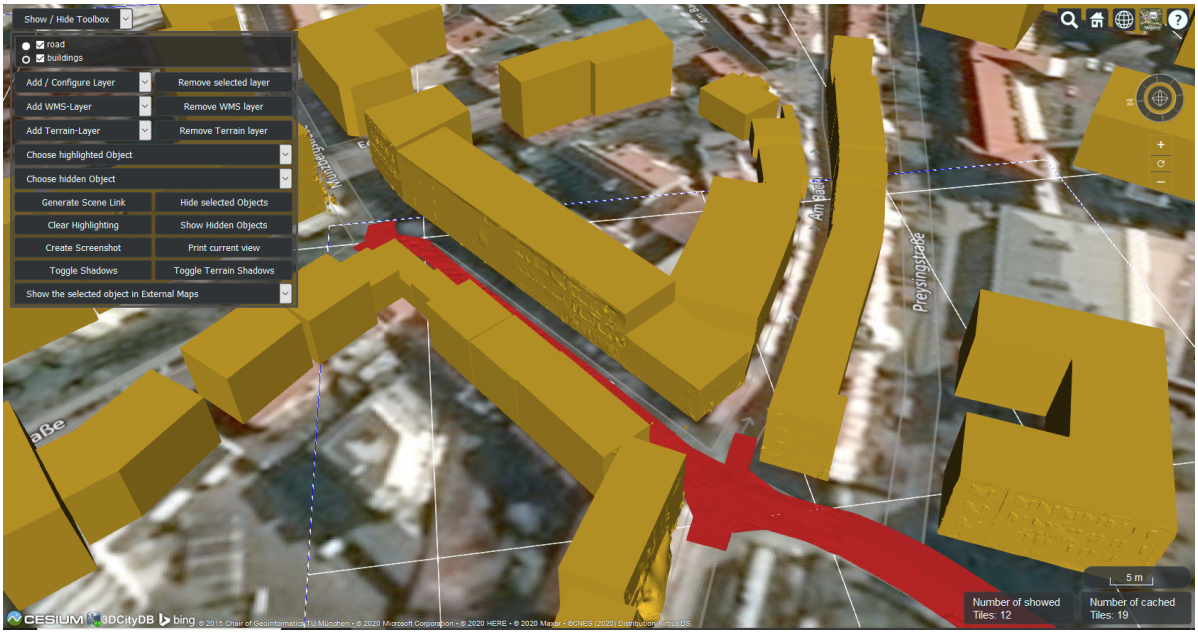

Figure 4.29.: The exported buildings and roads in 3DCityDB-Web-Map-Client

The main advantages of 3DCityDB-Web-Map-Client in comparison to Google Earth Pro is a possibility to change a base map to one of the shipped ones or to add a base map of user's choice via e.g. Web Map Service (WMS). Moreover, the *DualMaps* functionality enables exploration of selected objects in third party services like Google Street View or Google Maps with oblique images (see Figure 4.30. In the case of a tested area, the Google Street View service was not available due to a lack of Google's data for this specific region.

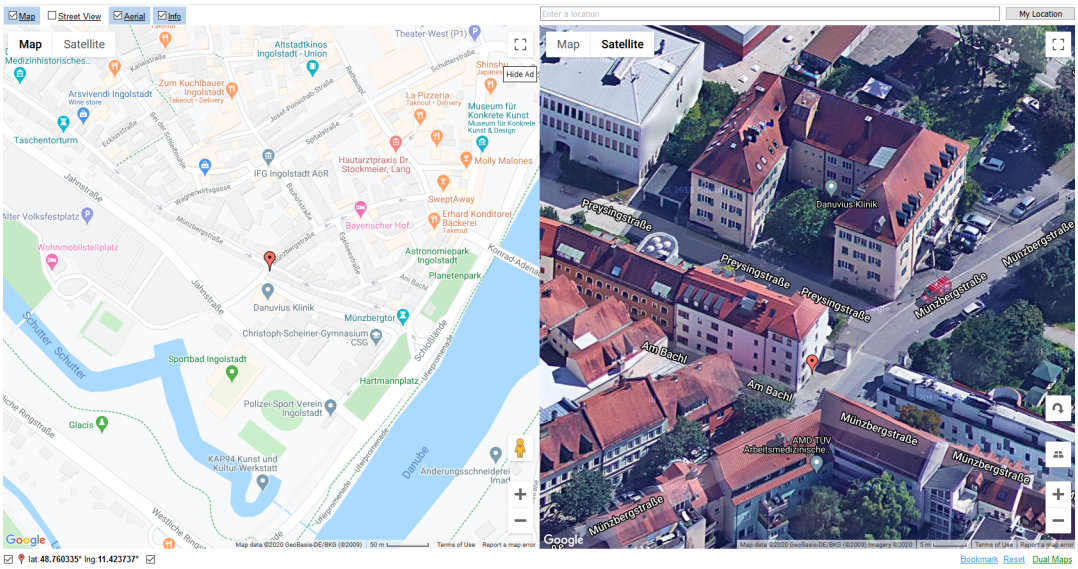

Figure 4.30.: A view of one of selected building using the *DualMaps* functionality. The same refined building model present in Figure 3.3 and in Figure 4.31

The main downside of the 3DCityDB-Web-Map-Client tool is limited exploration of refined wall structures if they overlap the raw building model which is possible in Google Earth Pro through highlighting reconstructed structures (see Figure 4.31). However, roads structures were properly visualised as there had been no raw structures shipped.

Figure 4.31.: Limited comparison possibilities of refined wall structures (green) with a raw model (yellow)

All of the configuration files to the Importer/Exporter tool and a configuration URL (Uniform Resource Locator) was added to this thesis in Appendices.

Unreal Engine

The Unreal Engine plays a pivotal role in modern automated driving simulators like CARLA. As the topic of this thesis is focused on the delivery of detailed testing grounds for simulators the proof of concept had to be performed on Unreal Engine. The test scenario was constructed to test whether models could be deployed on Unreal Engine. Additionally, the reconstructed and raw objects had to be distinguishable and possible to interact with. The check was performed with the following assumption: if objects' import to the editorial mode and export of a project to an interactive game *IngolstadtCitizen* is possible the models can be used in automated driving simulators like CARLA.

The Unreal Engine tool was not primarily designed to work with spatial data. Thus, the software does not have a possibility to directly import CityGML models to a working environment. Therefore, the models were transformed into the Datasmith format using a dedicated writer within FME Workbench. The Datasmith importer can directly incorporate models into the working environment of Unreal Engine. In order to provide easy to manage, tagged models within the working environment of Unreal Engine the attribute *UnrealEngineName* was created. Each value of the attribute represents *Building* (raw buildings models), *BuildingPart* (refined structures of a building) and *Road* (refined road structures) - according to CityGML's terminology standard.

The scenario proved that all of the reconstructed models could be consumed by Unreal Engine. Moreover, interaction with models was possible. A dedicated GUI was created that enables toggling on and off raw buildings models and reconstructed buildings respectively. This has proved that semantics of city models can be incorporated into Unreal Engine. However, only one attribute can be added to *UnrealEngineName*. The exploration pace and an overall details depiction were rated as very high.

Models within the scope of reconstruction were created without textures. However, it is possible to add textures for models in Unreal Engine which can enhance exploration results. This was performed within this test scenario for a randomly selected building and its reconstructed wall (see Figure 4.32 and 4.33 and 4.34).

Figure 4.32.: Raw buildings (grey-coloured), refined wall structures (purple colour) and roads (grey, tiled pattern) in the exploratory game *IngolstadtCitizen* using Unreal Engine

Figure 4.33.: Inspection of results in the exploratory game *IngolstadtCitizen* using interactive GUI. Refined walls are invisible

Figure 4.34.: Inspection of results in the exploratory game *IngolstadtCitizen* using interactive GUI. Only refined walls visible

Additional objects like trees, bushes, fireplaces and pedestrians were added. This test proved that reconstructed models can be augmented by extra objects.

The main disadvantage of Unreal Engine is a lack of support for CRS. Therefore, all georeferenced objects had to be transformed into a local, Cartesian reference system. This limitation creates a problem for all models except for small-scale ones.

4.1.6. Manholes detector

A small testing dataset can be compared with a ground truth acquired by visual inspection from photos gathered during the MLS campaign (see Figure 4.35).

SegmentID	expected	predicted
seg1	Manhole	Manhole
seg2	Manhole	Manhole
seg3	Manhole	Manhole
seg4	Manhole	Manhole
seg5	Manhole	noManhole
seg6	noManhole	noManhole
seg7	noManhole	noManhole
seg8	noManhole	noManhole
seg9	noManhole	noManhole
seg10	noManhole	noManhole

Figure 4.35.: 10 tested road segments with respective detector scores

Given Figure 4.35 accuracy of 90% has been determined. However, what has to be emphasised here, is that the number of testing samples has no statistical significance and thus this figure should serve more as an outlook for the future work.

In order to better assess results, it is possible to introduce a confusion matrix (see Figure 4.36). However, still, the problem of a small sample group exists and this figure just serves as a hint for future researchers.

	manhole	noManhole
manhole	4	0
noManhole	1	5

Figure 4.36.: Confusion matrix

The detector is also biased towards types of specific manholes that are common in Germany. This limits the global extension of the solution. Furthermore, there are other types of manholes that could be falsely detected on a road surface. The solution assumes that there is a maximum of one manhole on the selected road segment. Therefore, in the case of more manholes present the detector will likely return inaccurate results. Moreover, the centre of a detected manhole can be shifted towards a more dense representation of the manhole by a point cloud in a specific intensity range. However, the results are oscillating in the range of manholes by maximal of 30 cm.

The general aim of this detector is to give a gist how one can utilise refinements the geometries in order to semantically enrich city models. Also, it utilises a novel approach

of city modelling introduced by CityGML 3.0 which opens new possibilities for the 3D modelling. The example of a detected manhole is presented in Figure 4.37.

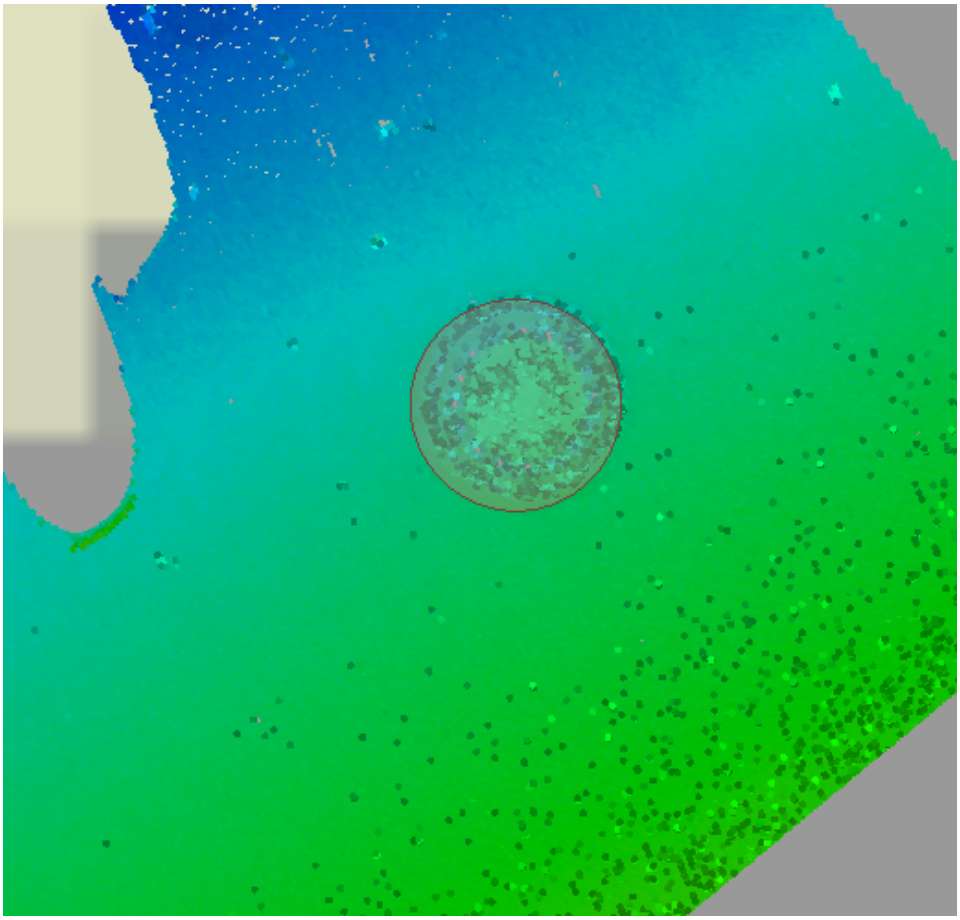

Figure 4.37.: One of the detected manholes

4.2. Summary of results of conducted tests and workflow implementation:

Workflows and scripts:

- An FME Workspace file for buildings reconstruction (*BuildingsRefinement.fmw*)
- An FME Workspace file for roads reconstruction (*RoadsRefinement.fmw*)
- An Python script for RANSAC algorithm, integrated within the FME Workspace file (*BuildingsRefinement.fmw*) and as a standalone script (*RANSAC.py*)
- An MLX template script for MeshLabServer *SurfaceReconstructionMeshLabTemplate.mlx*

- A Python script for XML parsing, integrated within A FME Workspace files (*BuildingsRefinement.fmw* and *RoadsRefinement.fmw*) and as a standalone script (*XMLparser.py*)
- An FME Workspace file for detection of manholes (*ManholeDetector.fmw*)
- An FME Workspace file for transformation of CityGML models to the Datasmith format (*CityGML2UnrealEngine.fmw*)
- An FME Workspace file for CityGML models merging to the one city model (*CityGMLMerger.fmw*)

Datasets:

- Refined geometries of buildings within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*building8lvl.7z* (total: 87 buildings of which 69 refined))
- Refined geometries of buildings within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*building10lvl.7z* (total: 87 buildings of which 69 refined))
- Refined geometries of buildings within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*building12lvl.7z* (total: 87 buildings of which 69 refined))
- Refined geometries of roads within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*road8lvl.7z* (total: 94 roads of which 94 refined))
- Refined geometries of roads within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*road10lvl.7z* (total: 94 roads of which 94 refined))
- Refined geometries of roads within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*road12lvl.7z* (total: 94 roads of which 94 refined))
- Refined geometries of roads and manholes within city centre of Ingolstadt, Bavaria, Germany as CityGML 3.0 objects (*roadManhole.7z* (total: 10))

Visualisations:

- An exploratory game consisting of refined city objects of Ingolstadt, Bavaria, Germany made in Unreal Engine (*IngolstadtCitizen.7z*)
- A 3DCityDB-Web-Map-Client configuration of the visualisation of refined city models (*RefinedIngolstadtScene.7z*)
- A configuration file to the Importer/Exporter tool with enabled highlight option in Google Eart Pro (*ingolstadtRefineSettHighlightON.xml*)
- A configuration file to the Importer/Exporter tool with disabled highlight option in Google Eart Pro (*ingolstadtRefineSettHighlightOFF.xml*)
- A configuration file to the Importer/Exporter tool with tailored to Cesium and the 3DCityDB-Web-Map-Client visualisation style (*ingolstadtRefineSettCesium.xml*)

5. Discussion

Within this chapter, a discussion of research questions and hypotheses is provided. Those are implicitly discussed within Chapter 4.

How to implement a reproducible, automatic surface reconstruction workflow for non-professionals?

In order to create a workflow which is reproducible and easy to maintain even by a non-professional user, one should implement an end-to-end method. This means that by setting up desired parameters and providing necessary input datasets a user should be able to recreate the implemented methodology. However, the main challenge of a real scene reconstruction is that it needs efficient GIS and Photogrammetry libraries. Therefore, an integration of those libraries is crucial to create such a workflow. This can be achieved with tools like the FME. The main advantage of this software is that it comprises state-of-the-art GIS-oriented methods. Furthermore, it has a function to integrate programming scripts and can communicate with operating system functions. A created workflow can be saved as a template and used by other operators easily. If adjustable parameters are introduced, any potential user can provide suitable values via a user-friendly GUI that additionally provides detailed documentation of those parameters. As a consequence, the user does not have to concentrate on a thorough understanding of the method in order to efficiently apply it to a task at hand. Neither it is required to change the implementation of a method that has been proved to work for a given class of problems. An additional advantage of FME is that a user does not to be proficient at programming to be able to execute the workflow.

As a process which runs in the background of this workflow is complex, one has to consider a cost of longer processing time of the whole workflow due to integration of many tools (i.e libraries, scripting languages, external functions). Nevertheless, a manual recreation of all the steps automated by the workflow would require more time and is error-prone. Therefore there is a trade-off between automated processing time and the level of automation. However, even in the case of working with a pre-defined workflow, a few manual steps have to be completed. For vertical-like objects, refinement implementation a correct Python global or a local environment has to be set up. Albeit FME has an option to import libraries and select desired Python version, in practice libraries often are incompatible with FME's native packages. That is why an external Python interpreter has to be chosen. This issue can cause a problem for non-professionals however there are tutorials and step-by-step explanations available. One of this kind of sources was created within the scope of the project as an issue request to the FME community forum [oloocki, 2020].

***A priori* road space information (e.g. obtained from OpenDRIVE datasets) can enhance the results of point cloud segmentation algorithms**

Scene reconstruction methods applied in order to create surfaces are often point cloud-based. Though, they do not take into consideration geo-contextual information. The results of this thesis demonstrate that it is possible to introduce such information into a scene pipeline. Moreover, this additional step enhances segmentation algorithms and thus a final reconstruction (see Chapter 4). As a result, one can focus on concrete city parts or even single objects. This means also that irrelevant point cloud parts can be rejected at the beginning of processing and thus minimise a computational effort. Taking into consideration those aspects the hypotheses that coarse 3D models can improve results of point cloud segmentation algorithms is correct.

However, one has to remember that solutions based on a *a priori* information are biased towards provided input datasets. This can generate positive results mentioned previously but in case of incomplete, not harmonised, or erroneous datasets can result in poor segmentation as well as scene reconstruction results. Therefore, it is important to provide only valid source datasets. Moreover, the methodology assumes that for vertical-like objects only one wall (with small possible deviations) exists per one input wall from a provided dataset. Though, vertical-like features might consist of several wall's parts like balconies, stair etc. Nevertheless, those features are segmented out of a point cloud portion for a reconstruction algorithm.

Moreover, methods from Machine Learning and specifically Deep Learning are gaining attraction. In recent years many training datasets and efficient segmentation algorithms have been published. They mostly operate on small benchmark datasets (like 1 m x 1 m with around 4000 points) where they proved to be highly efficient (even up to 94 %) [Guo, Wang, Hu, et al., 2020]. However, for now, those algorithms struggle to generalise and work on a large-scale datasets (like the one used for the purpose of this thesis) [Guo, Wang, Hu, et al., 2020]. For example, a network trained on an input dataset coming from one street in China will not properly segment a point cloud for a whole city in Europe.

On the other hand, this thesis has proven that incorporation of operator's knowledge and prior information from datasets together with efficient Computer Vision algorithms assures robust segmentation of point clouds' at a city's district scale.

To sum up, the direction of further research in this field is still unclear but it is believed that this thesis has the potential to answer questions of researchers.

How can Mobile Laser Scanning point clouds having intensity values be utilised in a surface reconstruction of semantic vector objects?

The MLS point clouds which have only 3D spatial information and intensity values can be segmented and then used for surface reconstruction of vector objects. However, coverage of MLS point cloud is limited as it is acquired at the street level. In case of scanners mounted on the top of a car, it reduces the field of view to only objects visible from a road. Thus, not all parts of objects and not all objects can be covered by this type of point cloud. Also, the

obstruction of the field of view caused by buildings, vehicles, trees and smaller objects has to be taken into account. It is hard to determine which objects can or cannot be reconstructed by MLS type of point cloud as it is depending on a local situation. Therefore, the incorporation of an operator's knowledge about a dataset is important to achieve satisfying results. The exception of this rule, are features located at the top of an object directly occluded by lower parts of the same object e.g. flat roofs of buildings. Those cannot be reconstructed using MLS point clouds.

In order to achieve better results of reconstruction, one can check the coverage of a point cloud per selected part of an object. Then, it is possible to reject poorly covered parts which will result in a low-resolution reconstruction. The other approach to enhance reconstruction results is to utilise supportive point cloud datasets like from ALS which can fill in gaps resulted in not covered parts (e.g. roads occluded by cars). Additionally, those approaches assume incorporation of information saved in raw 3D models. The main focus is on a geometrical reconstruction. In order to use those methods, the intensity information within MLS point clouds datasets does not have to be utilised.

However, points' intensity values can be used to add additional features to vector objects. Thus, the semantic enrichment of existing city models is possible. The tested scenario (see Chapter 4) proved that hypothesis. For example, manholes can be detected by using refined geometries, MLS measurements, and its intensity values. Novel standards of city models like CityGML 3.0 open new possibilities for modelling of such detailed structures of objects (like manholes).

However, the testing was performed on a statistically insignificant sample of 10 objects. It is believed that this approach has to be investigated in more detail.

To what extent and how can the workflow be parametrised by a user?

The reproducible workflow can be steered by a set of parameters. Those parameters are responsible for the fidelity of the surface reconstruction. However, higher fidelity results in longer processing time. Thus, through parameters, an operator can explicitly set the desired fidelity of reconstruction but it implicitly impacts processing time too.

There are numerous possible parameters that can be chosen to control the flow of the geometry refinement. Four of them, with the highest impact on the final fidelity level and processing time, can be classified as crucial and thus should be customisable for the user. The rest of the possible parameters can be treated as fixed as they are fine-tuned for city models reconstruction. Though those can be also changed by stepping into a concrete function. The parametrisation is implemented through *User Parameters* in an FME Desktop Workspace file. The selected or typed-in parameters are inserted into an XML file which however steers MeshLabServer processing steps. The program's execution depends on chosen parameter values.

Moreover, the usage of different datasets steers the reconstruction differently. Thus, they can be perceived as input parameters. Working with LoD1 buildings instead of LoD2 will result in a larger point cloud to process as the absolute accuracy is lower. In terms of workflow

execution, a very important aspect of an object is its type (i.e. horizontal- or vertical-like) because of different characteristics of those types. Generally, horizontal-like objects might be processed more than twice as much faster as vertical-like objects (see Chapter 4). This is mainly due to the simpler complexity of horizontal-like features (e.g. roads) than vertical-like objects (e.g. buildings).

The introduced parameters should serve for most common surface reconstruction tasks. However, not all future users' needs have been discovered yet. This project should evolve based on feedback from the community interested in the topic. Thus, an open-source repository on GitHub has been established (the address to the repository is within Appendices).

6. Conclusion and outlook

6.1. Conclusion

Over the past years, two trends in the spatial information field have arisen. The 3D GIS branch has already made an impact among others in the field of smart cities. Now, researches and vendors are seeking new applications of city models. In the field of simulations, data play a crucial role. Thus, the geodata in simulations of automated driving functions is essential to test the behaviour of sensors. City models are mostly created using ALS and cadastral data which have mostly a low temporal resolution and a high spatial accuracy at the national or international scale.

Mobile mapping solutions gain popularity thanks to vast investments in fields like automated and autonomous driving. The data from MLS is getting more accessible, measuring devices are getting cheaper and the community, both academic and industrial, in this field is growing. The MLS data should have a high temporal and spatial resolution at the street level scale.

Those two trends are interconnected and overlap each other. One field can be complementary to the other. However, the link between geospatial and automotive branch is still under development. City models lack automatic reconstruction solutions and data at the street level space. The automotive branch, however, lacks a high fidelity of vector geodata to perform robust simulations. The problem is also beyond the data itself. As it is an interdisciplinary issue there is a lack of professionals with such a wide field of expertise. Taking into consideration those issues, one of the missing links connecting those two branches has been developed within the scope of this thesis.

The main requirement to this work was to create a solution which will allow for automatic reconstruction of road space objects with a user-desired fidelity level. The requirements that a user has non-professional knowledge and that the tool should be possible to customise were also declared at the beginning of the project. Moreover, the *a priori* geo-context knowledge had to be taken into account to minimise a computational complexity of the tool.

In order to overcome those challenges, certain substeps had to be taken. First of all, the solution should serve as the end-to-end method. This was achieved by using FME software which integrated LASTools and Python libraries enabling segmentation of point cloud data and a scene reconstruction as well as saving to formats like CityGML. The control over the workflow was enabled through parameters (published parameters in FME) which are fine-tuned by default with a possibility to change them by a user in a user-friendly manner via GUI. The corresponding explanations to the meaning of parameters were added. Therefore, the user can decide at the beginning what fidelity of the output data he expects as well as estimate processing time based on conducted tests documented in this thesis.

To introduce semantics from coarse road space geometries certain measures were taken. This allowed to minimise the computational complexity and enhance final reconstruction results. Furthermore, it was proved that not only explicit semantics from models could be utilised but also country-specific road standards for a specific country. This information first decreases the area of interest for each of the introduced models, allows filtering noise in the input point cloud and simplify segmentation of the point cloud to the 2D space. Moreover, the geo-context knowledge with a combination of intensity values inherited from MLS data allows also to enrich semantically refined objects.

However, one has to remember that semantic-based solutions can also generate false results. This can be a case when input data have errors and/or lack harmonisation. Therefore, one has to remember that the method based on *a priori* semantics is biased towards raw models.

Moreover, testing scenarios and visualisations proved that the methodology can yield expected input to the state-of-the-art simulation engines and 3D GIS tools.

The research questions mentioned at the beginning of this thesis were addressed. The solution allows seamless integration of different datasets, segmentation methods, and GIS analysis in order to automatically reconstruct object surfaces, in addition, provide with options for user-friendly customisation.

6.2. Outlook

Obviously, in the case of research projects, there is always room for improvements. This project is not an exception to this rule.

First of all, it is believed that the processing time of the workflow can be reduced. The FME software does not have a dedicated function to cut mesh objects by 2D and 3D geometries. Therefore, each mesh has to be split to simpler geometries, then cut by the corresponding extent, and aggregated again - this is a time-expensive task. A possible solution for that could be an integration of external software with a suitable function or a release of updated FME version - there is a feature request opened by the community to introduce such possibility [Safe Software, 2019].

The segmentation of point cloud, however successful, still results in some falsely classified points, specifically in a point cloud processing for the vertical-like objects. One of the possible solutions to overcome the problem would be to reject not adjacent, low density and remote parts of surfaces. Presumably, it should be possible before the surface is reconstructed.

The promising part of the methodology is also Semantic Enrichment. The so far tested road segments allow believing that this method can be developed and applied for more road objects other than manholes e.g. ruts or potholes. Furthermore, it is believed that the method can be applied to detect objects e.g. walls or windows on vertical structures. Those encroachments together with novel approaches and concepts of city modelling like CityGML 3.0 have been tested within this project and provided promising results and tailored 3D models for automated driving functions testing (see Figure 6.1). Those findings should be investigated in more detail in the future.

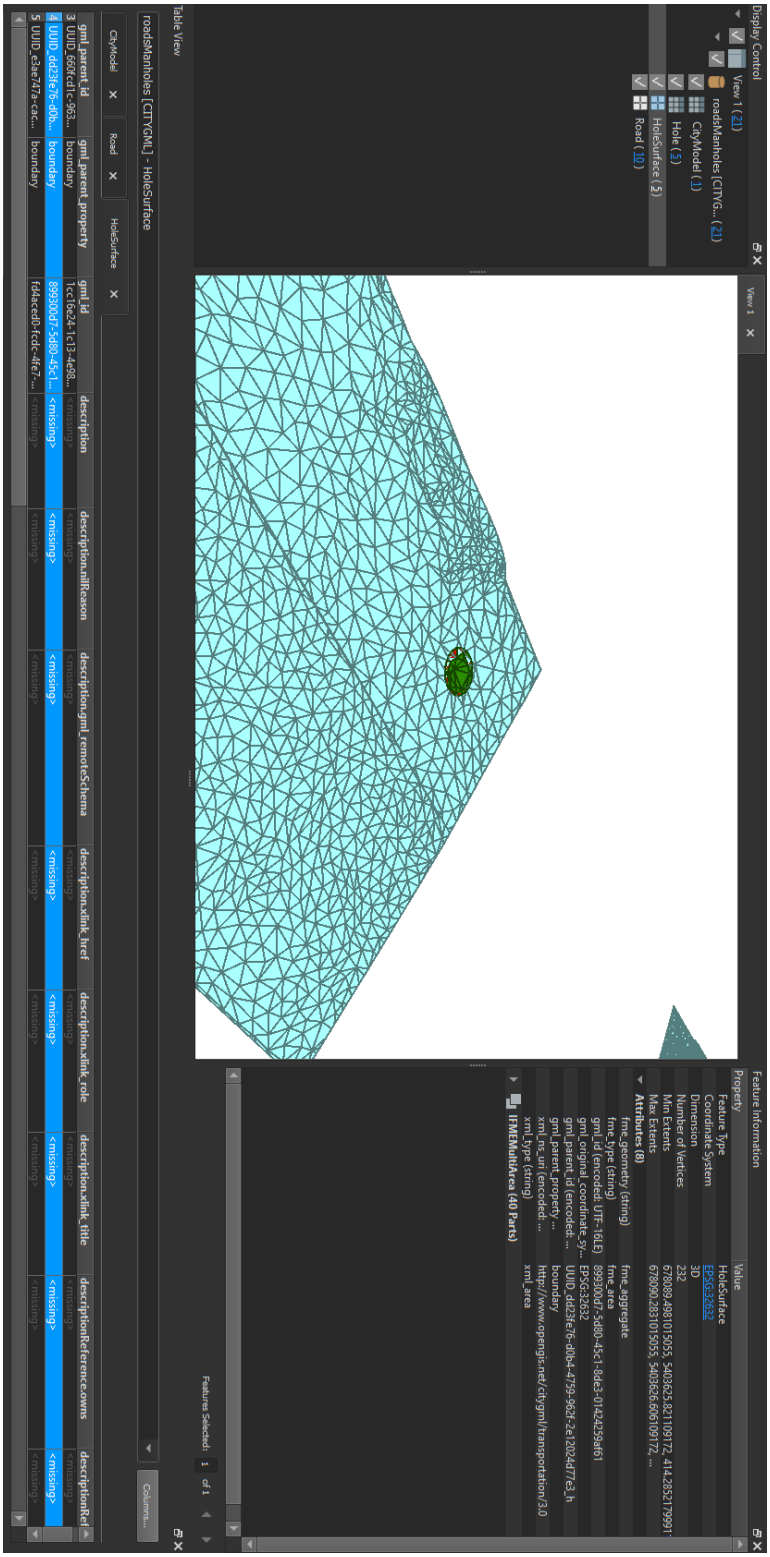

Figure 6.1.: One of the detected manholes saved in the CityGML 3.0 standard. This shows promising results for the future research

A. Appendices

A.1. Appendix A: Implementation of the methodology

The implementation of the methodology part (FME Workspaces, Python scripts, MLX templates for MeshLabServer):

- The FME Workspace file for buildings reconstruction (*BuildingsRefinement.fmw*)
- The FME Workspace file for roads reconstruction (*RoadsRefinement.fmw*)
- The Python script for RANSAC algorithm, integrated within the FME Workspace file (*BuildingsRefinement.fmw*) and as a standalone script (*RANSAC.py*)
- The MLX template script for MeshLabServer *SurfaceReconstructionMeshLabTemplate.mlx*
- The Python script for XML parsing, integrated within the FME Workspace files (*BuildingsRefinement.fmw* and *RoadsRefinement.fmw*) and as a standalone script (*XMLparser.py*)
- The FME Workspace file for detection of manholes (*ManholeDetector.fmw*)
- The FME Workspace file for transformation of CityGML models to the Datasmith format (*CityGML2UnrealEngine.fmw*)
- The FME Workspace file for CityGML models merging to the one city model (*CityGMLMerger.fmw*)

This implementation is available at the following address:
https://github.com/OloOcki/CityModelsRefinement

A.2. Appendix B: Output datasets

Results of testing scenarios at different octree levels:

- Refined geometries of buildings within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*building8lvl.7z* (total: 87 buildings of which 69 refined))
- Refined geometries of buildings within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*building10lvl.7z* (total: 87 buildings of which 69 refined))
- Refined geometries of buildings within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*building12lvl.7z* (total: 87 buildings of which 69 refined))

- Refined geometries of roads within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*road8lvl.7z* (total: 94 roads of which 94 refined))
- Refined geometries of roads within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*road10lvl.7z* (total: 94 roads of which 94 refined))
- Refined geometries of roads within city centre of Ingolstadt, Bavaria, Germany as CityGML 2.0 LoD1 objects (*road12lvl.7z* (total: 94 roads of which 94 refined))
- Refined geometries of roads and manholes within city centre of Ingolstadt, Bavaria, Germany as CityGML 3.0 objects (*roadManhole.7z* (total: 10))

Those datasets are available at the following address:
https://github.com/OloOcki/CityModelsRefinement

A.3. Appendix C: Visualisation of results

Also, visualisation results (3D City Database configuration files, the Unreal Engine game):

- Exploratory game consisting of refined city objects of Ingolstadt, Bavaria, Germany made in Unreal Engine (*IngolstadtCitizen.7z*)
- 3DCityDB-Web-Map-Client configuration of the visualisation of refined city models (*RefinedIngolstadtScene.7z*)
- Configuration file to the Importer/Exporter tool with enabled highlight option in Google Eart Pro (*ingolstadtRefineSettHighlightON.xml*)
- Configuration file to the Importer/Exporter tool with disabled highlight option in Google Eart Pro (*ingolstadtRefineSettHighlightOFF.xml*)
- Configuration file to the Importer/Exporter tool with tailored to Cesium and the 3DCityDB-Web-Map-Client visualisation style (*ingolstadtRefineSettCesium.xml*)

Those visualisations are available at the following address:
https://github.com/OloOcki/CityModelsRefinement

List of Figures

Acronyms

3DCityDB 3D City Database. 90

ALS Airborne Laser Scanning. 1, 2, 13, 21, 26, 28, 29, 31, 36–38, 40, 72, 78, 105, 107

ASAM Association for Advancement of international Standardization of Automation and Measuring Systems e.V. 10

CGAL The Computational Geometry Algorithms Library. 20, 21

CRS Coordinate Reference System. 10, 29, 50, 57–60, 99

EPSG European Petroleum Survey Group. 29

FME Feature Manipulation Engine. v, 22–24, 31, 33, 36, 47, 50, 53, 71, 89, 91, 97, 101, 102, 107, 108, 110

GIS Geographic Information Systems. 1, 10, 12, 13, 23, 26, 31, 33, 71, 103, 107, 108

GML Geography Markup Language. 15, 61

GML3 Geography Markup Language version 3. 12

GUI Graphical User Interface. 23, 24, 53, 97–99, 103, 107

HD Map High Definition Map. 2, 10, 26, 31, 72, 73

Importer/Exporter 3D City Database Importer/Exporter. 25, 31, 89–93, 95, 97, 102, 111

LDBV Bavarian State Office for Survey and Geoinformation (ger. Landesamt für Digitalisierung, Breitband und Vermessung). 27, 28

LiDAR Light Detection And Ranging. 2

LoD0 Level of Detail 0. 12, 13

LoD1 Level of Detail 1. 13, 25, 26, 31, 33, 34, 45, 79, 102, 105, 110, 111

LoD2 Level of Detail 2. 1, 2, 13, 21, 22, 27, 31, 33, 79, 105

LoD3 Level of Detail 3. 1, 2, 13, 14, 21, 22, 27, 28, 31, 71, 78–82, 114

LoD4 Level of Detail 4. 12

LRS Linear Referencing System. 10

MLS Mobile Laser Scanning. 1, 2, 11, 21, 22, 26–31, 36–38, 40–42, 62, 72, 78, 99, 104, 105, 107, 108

OGC Open Geospatial Consortium. 12, 14, 15

RANSAC RANdom SAmple Consensus. 18, 19, 24, 31, 46–49, 81, 88, 101, 110, 113

TLS Terrestrial Laser Scanning. 21, 22

UAV Unmanned Aerial Vehicle. 13

UML Unified Modeling Language. 15

VTD Virtual Test Drive. 11

WMS Web Map Service. 95

XML eXtensible Markup Language. v, 10–12, 15, 22, 24, 50, 53, 89, 102, 110

Bibliography

Albrecht, C., Kraus, S., Zimmermann, A., & Stilla, U. (2019). A Concept For An Automated Approach Of Public Transport Vehicles To A Bus Stop. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-2/W16*, 13–20.

alemuntoni. (2020). Meshlab Server. Retrieved June 10, 2020, from https://github.com/cnr-isti-vclab/meshlab

Althoff, M., Urban, S., & Koschi, M. (2018). Automatic Conversion of Road Networks from OpenDRIVE to Lanelets, In *2018 IEEE International Conference on Service Operations and Logistics, and Informatics (SOLI)*, Singapore, IEEE.

Angermann, L., Donaubauer, A., Graw, K., Holtkamp, J., Huber, T., Nguyen, S. H., Schwab, B., & Wysocki, O. (2019). Trendanalyse INTERGEO 2019. *Runder Tisch GIS e.V.*, 13.

Audi AG. (2017). The new Audi A8 – conditional automated at level 3. Retrieved June 10, 2020, from https://www.audi-mediacenter.com:443/en/on-autopilot-into-the-future-the-audi-vision-of-autonomous-driving-9305/the-new-audi-a8-conditional-automated-at-level-3-9307

Audi Electronics Venture. (2020). Driving Dataset; Audi Electronics Venture. Retrieved June 10, 2020, from https://www.audi-electronics-venture.de/aev/web/de/driving-dataset.html

Autodesk. (2020). Roads & Highways Design Solutions | Road Infrastructure | Autodesk. Retrieved June 10, 2020, from https://www.autodesk.com/solutions/architecture-engineering-construction/roads-highways

Beil, C., & Kolbe, T. H. (2017). CityGML And The Streets Of New York -A Proposal For Detailed Street Space Modelling. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *IV-4/W5*, 9–16.

Beil, C., & Kolbe, T. H. (2018). Detaillierte Repräsentation des Straßenraums in 3D Stadtmodellen. *38. Wissenschaftlich-Technische Jahrestagung der DGPF und PFGK18 Tagung in München*, 12.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., & Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, *4*, 2842–2889.

Bodis-Szomoru, A., Riemenschneider, H., & Van Gool, L. (2016). Efficient volumetric fusion of airborne and street-side data for urban reconstruction, In *2016 23rd International Conference on Pattern Recognition (ICPR)*, Cancun, IEEE.

Bodis-Szomoru, A., Riemenschneider, H., & Van Gool, L. (2015). Superpixel meshes for fast edge-preserving surface reconstruction, In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, IEEE.

Cambridge Dictionary. (2020). Refinement | meaning in the Cambridge English Dictionary. Retrieved June 10, 2020, from https://dictionary.cambridge.org/dictionary/english/refinement

Castro, D. d. l. I. (2020). Welcome to pyntcloud! — pyntcloud 0.1.2 documentation. Retrieved June 10, 2020, from https://pyntcloud.readthedocs.io/en/latest/

CGAL. (2020a). CGAL 5.0.2 - Poisson Surface Reconstruction: User Manual. Retrieved June 10, 2020, from https://doc.cgal.org/latest/Poisson%5C_surface%5C_reconstruction%5C_3/index.html

CGAL. (2020b). CGAL 5.0.2 released. Retrieved June 10, 2020, from https://www.cgal.org/2020/02/25/cgal502/

Chair of Geoinformatics, Technical University of Munich. (2020). 3DCityDB Database – Homepage. Retrieved June 10, 2020, from https://www.3dcitydb.org/3dcitydb/

Chellapilla, K. (2018). Rethinking Maps for Self-Driving. Retrieved June 10, 2020, from https://medium.com/lyftlevel5/https-medium-com-lyftlevel5-rethinking-maps-for-self-driving-a147c24758d6

Coduro, T. (2018). Straßenraummodellierung mittels Mobile Mapping in OpenDRIVE und CityGML sowie Entwicklung geeigneter Visualisierungsmethoden. *(Master's Thesis) Technical University of Munich, Munich, Germany*. Retrieved March 10, 2020, from https://www.lrg.tum.de/en/gis/publications/student-theses/

Davies, A. (2018). Americans Can't Have Audi's Super Capable Self-Driving System. *Wired*. Retrieved April 4, 2020, from https://www.wired.com/story/audi-self-driving-traffic-jam-pilot-a8-2019-availablility/

de Prez, M. (2018). Legislation puts brakes on Audi's Level 3 autonomous technology. *Fleet News*. Retrieved April 4, 2020, from https://www.fleetnews.co.uk/news/manufacturer-news/2018/02/15/legislation-puts-brakes-on-audi-s-level-3-autonomous-technology

Dosovitskiy, A., Ros, G., Codevilla, F., López, A., & Koltun, V. (2017). CARLA: An Open Urban Driving Simulator. *Proceedings of Machine Learning Research*, *78*, 1–16.

DPCcars. (2018). Safe Automated Driving From Bosch CES 2019. Retrieved March 7, 2020, from https://www.youtube.com/watch?v=7H4HS6irkDQ&list=PL6Ht4QVO7jN4NInHmim1Gp2_E4b2w8QpY&index=2&t=0s

Engelmann, F., Kontogianni, T., Hermans, A., & Leibe, B. (2017). Exploring Spatial Context for 3D Semantic Segmentation of Point Clouds, In *2017 IEEE International Conference on Computer Vision Workshops (ICCVW)*, Venice, IEEE.

Epic Games. (2020). Unreal Engine | The most powerful real-time 3D creation platform. Retrieved August 12, 2020, from https://www.unrealengine.com/en-US/

Esri. (2020). What is linear referencing?—Help | ArcGIS for Desktop. Retrieved March 22, 2020, from https://desktop.arcgis.com/en/arcmap/10.3/guide-books/linear-referencing/what-is-linear-referencing.htm

Fiutak, G., Marx, C., Willkomm, P., & Donaubauer, A. (2018). Projekt 3D Digitales Landschaftsmodell (3D-DLM) am Runden Tisch GIS e.V., Abschlussbericht (Demonstrationsphase): Datenvorverarbeitung, Anwendung des 3Dfiers, Abbildung auf CityGML-

Datenmodell, Bereitstellung der Ergebnisdaten & Qualitätsbewertung. Retrieved March 19, 2020, from https://rundertischgis.de/images/5%5C_projekte/3D-DLM-Phase-2---Abschlussbericht.pdf

Frost & Sullivan. (2018). Global Autonomous Driving Outlook, 2018. Retrieved March 17, 2020, from https://info.microsoft.com/rs/157-GQE-382/images/K24A-2018%5C%20Frost%5C%20%5C%26%5C%20Sullivan%5C%20-%5C%20Global%5C%20Autonomous%5C%20Driving%5C%20Outlook.pdf

Geopoz, SHH, & virtualcitySYSTEMS. (2020). Model 3D Poznania. Retrieved March 22, 2020, from http://sip.poznan.pl/model3d/%5C#/legend

Griffiths, D., & Boehm, J. (2019). A Review on Deep Learning Techniques for 3D Sensed Data Classification. *Remote Sensing*, *11*, 1499.

Grilli, E., Menna, F., & Remondino, F. (2017). A Review Of Point Clouds Segmentation And Classification Algorithms. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-2/W3*, 339–344.

Gröger, G., Kolbe, T. H., Nagel, C., & Häfele, K.-H. (2012). OGC City Geography Markup Language (CityGML) Encoding Standard. *Open Geospatial Consortium*.

Gruen, A., Schubiger, S., Qin, R., Schrotter, G., Xiong, B., Li, J., Ling, X., Xiao, C., Yao, S., & Nuesch, F. (2019). Semantically Enriched High Resolution Lod 3 Building Model Generation. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-4/W15*, 11–18.

Guo, Y., Wang, H., Hu, Q., Liu, H., Liu, L., & Bennamoun, M. (2020). Deep Learning for 3D Point Clouds: A Survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*.

Hanke, T., van Driesten, C., Hirsenkorn, N., Garcia-Ramos, P., Schiementz, M., Schneider, S., & Biebl, E. (2017). Automotive Veröffentlichungen - Professur für Höchstfrequenztechnik. Retrieved March 4, 2020, from https://www.ei.tum.de/hot/forschung/automotive-veroeffentlichungen/

Hensel, S., Goebbels, S., & Kada, M. (2019). Facade Reconstruction For Textured Lod2 Citygml Models Based On Deep Learning And Mixed Integer Linear Programming. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *IV-2/W5*, 37–44.

HERE. (2020). HD Maps for Autonomous Driving and Driver Assistance. Retrieved March 5, 2020, from https://www.here.com/products/automotive/hd-maps

Hirsenkorn, N., Subkowski, P., Hanke, T., Schaermann, A., Rauch, A., Rasshofer, R., & Biebl, E. (2017). A ray launching approach for modeling an FMCW radar system, In *2017 18th International Radar Symposium (IRS)*.

Huang, R., Xu, Y., Hong, D., Yao, W., Ghamisi, P., & Stilla, U. (2020). Deep point embedding for urban classification using ALS point clouds: A new perspective from local to global. *ISPRS Journal of Photogrammetry and Remote Sensing*, *163*, 62–81.

Isenburg, M. (2020). LAStools. Retrieved March 11, 2020, from http://lastools.org/download/lasground/

Kashani, A., Olsen, M., Parrish, C., & Wilson, N. (2015). A Review of LIDAR Radiometric Processing: From Ad Hoc Intensity Correction to Rigorous Radiometric Calibration. *Sensors*, *15*, 28099–28128.

Kazhdan, M., Bolitho, M., & Hoppe, H. (2006). Poisson surface reconstruction. *Eurographics Symposium on Geometry Processing*.

Kazhdan, M., & Hoppe, H. (2013). Screened poisson surface reconstruction. *ACM Transactions on Graphics*, *32*, 1–13.

Kemmler Baustoffe GmbH. (2020). Beton-Guss Schachtabdeckung Klasse D 400 - rund - mit Ventilation | www.kemmler.de. Retrieved June 27, 2020, from https://www.kemmler.de/sortiment/produkt/beton-guss-schachtabdeckung/-/-/1050100003

Kolbe, T. H. (2009). Representing and Exchanging 3D City Models with CityGML. *Proceedings of the 3rd International Workshop on 3D Geo-Information*.

KPMG International. (2019). 2019 Autonomous Vehicles Readiness Index. *KPMG International*, 56.

Kutzner, T., Chaturvedi, K., & Kolbe, T. H. (2020). CityGML 3.0: New Functions Open Up New Applications. *PFG – Journal of Photogrammetry, Remote Sensing and Geoinformation Science*.

Landesamt für Digitalisierung, Breitband und Vermessung. (2020). Bayerische Vermessungsverwaltung - Produkte - 3D-Produkte - 3D-Gebäudemodell. Retrieved March 26, 2020, from https://www.ldbv.bayern.de/produkte/3dprodukte/3d.html

Lozé, S. (2019). CARLA democratizes autonomous vehicle R&D with free open-source simulator. Retrieved March 28, 2020, from https://www.unrealengine.com/en-US/spotlights/carla-democratizes-autonomous-vehicle-r-d-with-free-open-source-simulator

Martinovic, A., Knopp, J., Riemenschneider, H., & Van Gool, L. (2015). 3D all the way: Semantic segmentation of urban scenes from start to end in 3D, In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Boston, MA, USA, IEEE.

Menzel, T., Bagschik, G., & Maurer, M. (2018). Scenarios for development, test and validation of automated vehicles, In *2018 IEEE Intelligent Vehicles Symposium*.

MeshLab. (2020). MeshLab. Retrieved April 10, 2020, from https://www.meshlab.net/%5C#references

MPA Solutions, P. D., Servizio Gestione Strade, & Ufficio Controllo e Tecnologie Stradali. (2020). LRS Plugin for QGIS — LRS Plugin 0.3.7 documentation. Retrieved April 11, 2020, from https://blazek.github.io/lrs/release/help.0.3.7/index.html

oloocki. (2020). Python, external package installation, problem. Retrieved August 22, 2020, from https://community.safe.com/s/question/0D54Q000080hdLQSAY/python-external-package-installation-problem

Open Geospatial Consortium. (2020a). CityGML | OGC. Retrieved March 8, 2020, from https://www.opengeospatial.org/standards/citygml

Open Geospatial Consortium. (2020b). CityGML 3.0 Conceptional Model. Retrieved July 23, 2020, from https://github.com/opengeospatial/CityGML-3.0CM

Papon, J., Abramov, A., Schoeler, M., & Worgotter, F. (2013). Voxel Cloud Connectivity Segmentation - Supervoxels for Point Clouds, In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, Portland, OR, USA, IEEE.

Point Cloud Library. (2020). Documentation - Point Cloud Library (PCL). Retrieved April 10, 2020, from http://pointclouds.org/documentation/tutorials/region%5C_growing%5C_segmentation.php

Python Software Foundation. (2020). Welcome to Python.org. Retrieved April 1, 2020, from https://www.python.org/

Qi, C. R., Su, H., Mo, K., & Guibas, L. J. (2017). PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). PointNet++: Deep Hierarchical Feature Learning on Point Sets in a Metric Space. *Conference on Neural Information Processing Systems (NIPS) 2017*.

rapidlasso GmbH. (2012). LAStools. Retrieved August 1, 2020, from https://rapidlasso.com/lastools/

Riedmaier, S., Nesensohn, J., Gutenkunst, C., Duser, T., Schick, B., & Abdellatif, H. (2018). Validation of X-in-the-Loop Approaches for Virtual Homologation of Automated Driving Functions. *11th Graz Symposium Virtual Vehicle*.

Safe Software. (2020a). Linear Referencing. Retrieved May 16, 2020, from https://docs.safe.com/fme/html/FME%5C_Desktop%5C_Documentation/FME%5C_Transformers/Categories/linear%5C_referencing.htm

Safe Software. (2019). MeshClipper or 3DClipper or AnyGeometryClipper - FME Community. Retrieved July 16, 2020, from https://knowledge.safe.com/content/idea/86914/meshclipper-or-3dclipper-or-anygeometryclipper.html

Safe Software. (2020b). Safe Software | FME | Data Integration Platform. Retrieved March 17, 2020, from https://www.safe.com/

Safe Software Lab. (2017). LAStools.lasground | FME Hub. Retrieved April 1, 2020, from https://hub.safe.com/publishers/safe-lab/transformers/lastools-lasground

Scholz, M. (2019). Boosting the Development of OpenDRIVE through Integration Into Standardised GIS Frameworks. *ASAM International Conference*, 27.

Schuldt, F. (2017). Ein beitrag für den methodischen test von automatisierten fahrfunktionenmit hilfe von virtuellen umgebungen. *(Dissertation), Technical University of Braunschweig, Braunschweig, Germany*. Retrieved March 21, 2020, from https://publikationsserver.tu-braunschweig.de/receive/dbbs_mods_00064747

Schwab, B., & Kolbe, T. H. (2019). Requirement Analysis Of 3d Road Space Models For Automated Driving. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, *IV-4/W8*, 99–106.

Schwab, B. (2020). Tum-gis/rtron. Retrieved March 19, 2020, from https://github.com/tum-gis/rtron

Schwab, B., Beil, C., & Kolbe, T. H. (2020). Spatio-semantic road space modeling for vehicle–pedestrian simulation to test automated driving systems. *Sustainability*, *12*(9), 3799.

Swiss Federal Institute of Technology in Zurich. (2017). VarCity - The Video - semantic and dynamic city modelling from images. Retrieved March 3, 2020, from https://www.youtube.com/watch?v=6pjEs84DR6Q&t=747s/

Taylor, M. (2017). The Level 3 Audi A8 Will Almost Be The Most Important Car In The World. *Forbes*. Retrieved March 21, 2020, from https://www.forbes.com/sites/michaeltaylor/2017/09/10/tthe-level-3-audi-a8-will-almost-be-the-most-important-car-in-the-world/

Team SmartShuttle. (2018). SmartShuttle Sion 2.0 Projektzwischenbericht 2018 für das Bundesamt für Strassen (ASTRA). *Sion, Switzerland: PostAuto, Project Interim Report*, 19.

TomTom. (2020). HD Map. Retrieved March 16, 2020, from https://www.tomtom.com/products/hd-map/

Ulrich, M. (2019). Close-Range Photogrammetry, Parameter Estimation. *(Lecture notes) Industrial Photogrammetry, 2019/2020 WS, MVTec Software GmbH & Technical University of Munich Photogrammetry and Remote Sensing*.

VIRES Simulationstechnologie GmbH. (2020a). OpenDRIVE - Home. Retrieved March 12, 2020, from http://www.opendrive.org/

VIRES Simulationstechnologie GmbH. (2020b). OpenDRIVE - References. Retrieved March 12, 2020, from http://www.opendrive.org/references.html

VIRES Simulationstechnologie GmbH. (2019). OpenDRIVE FormatSpecRev1.5M. Retrieved March 12, 2020, from http://www.opendrive.org/docs/OpenDRIVEFormatSpecRev1.5M.pdf

VIRES Simulationstechnologie GmbH. (2020c). VTD - VIRES Virtual Test Drive. Retrieved March 5, 2020, from https://vires.mscsoftware.com/vtd-vires-vir-

virtualcitySYSTEMS GmbH. (2018). CityGML-basierte digitale Städte. Retrieved March 10, 2020, from https://www.virtualcitysystems.de

Vo, A.-V., Truong-Hong, L., Laefer, D. F., & Bertolotto, M. (2015). Octree-based region growing for point cloud segmentation. *ISPRS Journal of Photogrammetry and Remote Sensing*, *104*, 88–100.

Wachenfeld, W., Junietz, P., Wenzel, R., & Winner, H. (2016). The worst-time-to-collision metric for situation identification, In *2016 IEEE Intelligent Vehicles Symposium (IV)*, IEEE.

Wang, R., Peethambaran, J., & Dong, C. (2018). LiDAR Point Clouds to 3D Urban Models: A Review. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*.

Wen, X., Xie, H., Liu, H., & Yan, L. (2019). Accurate Reconstruction of the LoD3 Building Model by Integrating Multi-Source Point Clouds and Oblique Remote Sensing Imagery. *ISPRS International Journal of Geo-Information*, *8*, 135.

Willenborg, B., Pültz, M., & Kolbe, T. H. (2018). Integration of Semantic 3D City Models and 3D Mesh Models for Accuracy Improvements of Solar Potential Analyses. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, *XLII-4/W10*, 223–230.

Wysocki, O. (2019). Positioning of autonomous vehicles. *Geodesy & Geoinformation, MSc course 2019 SS, Photogrammetry - Selected Topics (PST), Technical University of Munich*.

Wysocki, O., & Albrecht, C. (2019). Project Photogrammetry and Remote Sensing: Augmentation of CityGML Models with information from HD-Maps. *Geodesy & Geoinformation, MSc course 2019/2020 WS, Photogrammetry and Remote Sensing Project, Chair of Photogrammetry and Remote Sensing TUM Department of Aerospace and Geodesy Technical University of Munich*.

Yang, M. Y., & Forstner, W. (2010). Plane Detection in Point Cloud Data. *(Technical report) Technical Report Nr. 1, University of Bonn, Bonn, Germany*. Retrieved March 16, 2020, from http://www.ipb.uni-bonn.de/technicalreports/

Yao, Z., Nagel, C., Kunde, F., Hudra, G., Willkomm, P., Donaubauer, A., Adolphi, T., & Kolbe, T. H. (2018). 3DCityDB - a 3D geodatabase solution for the management, analysis, and visualization of semantic 3D city models based on CityGML. *Open Geospatial Data, Software and Standards*, *3*, 5.

Zhou, Y., & Tuzel, O. (2017). VoxelNet: End-to-End Learning for Point Cloud Based 3D Object Detection. *2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.