Thesis in geographical information technics nr 45

Parametric Roof Modelling Using a Machine Learning Approach

Iris Åberg & Thelma Åsvärd

Program in surveying and land management
Faculty of Engineering

Department of Earth and Environmental Sciences
Lund University

Lund University
Faculty of Engineering

Parametric Roof Modelling Using a Machine Learning Approach

EXTM05 Master thesis, 30 ECTS
Program in surveying and land management

Iris Åberg & Thelma Åsvärd

Supervisor
Lars Harrie
Department of Earth and Environmental Sciences

May 29, 2026

Opponent: Nils Pedersen
Examiner: Per-Ola Olsson

Department of Earth and Environmental Sciences
Lund University
Sölvegatan 12
223 62 Lund

Telephone: 046-222 30 30
Fax: 046-222 03 21
Web: http://www.nateko.lu.se

Examensarbete i geografisk informationsteknik nr 45
Printed by E-tryck, E-huset, 2026

Preface

This report presents the results of a master's thesis conducted within the field of Geographical Information Science, carried out as part of the Master of Science in Engineering, Surveying and Land Management program at Lund University. The thesis comprises 30 ECTS credits and was carried out during the spring of 2026 in collaboration with Sweco.

First and foremost, we would like to express our sincere gratitude to our academic supervisor at LTH, Lars Harrie, whose guidance, expertise, and continuous support have been invaluable throughout the project. The insightful feedback and encouragement provided during the entire process greatly contributed to the development of this work.

We would also like to thank our supervisors at Sweco, Anna Larsson and Ulf Månsson, for their engagement, support, and many valuable ideas throughout the study. Their practical knowledge, constructive discussions, and continuous encouragement have been of great importance for the completion of this thesis.

Special thanks are directed to Göteborgs Stad for providing the data used in this study, as well as for valuable discussions and input during the early stages of the project.

In addition, we would like to thank everyone at Sweco who contributed with support, discussions, and a welcoming work environment during the project.

Finally, we would like to thank our classmates, friends, and families for their encouragement and support throughout our years of study.

Iris Åberg & Thelma Åsvärd

Abstract

Three-dimensional building models are becoming increasingly important in areas such as urban planning, simulations, and geographical analyses. At the same time, detailed information about roof structures is often missing from current geospatial datasets, which complicates the automated reconstruction of coherent and functional 3D building models. Point cloud data acquired through laser scanning enables detailed geometric representation of buildings, but classification of roof structures from point clouds remains a complex and resource-demanding task. In addition, the application of machine learning methods is limited by the lack of annotated training data.

This thesis investigates the possibility of classifying real-world roof structures based on point cloud data using a machine learning model trained on synthetically generated data. The study also examines how classified roof types can be represented parametrically and used for reconstruction of simplified 3D building models.

The methodology consists of four main stages:

- Generation of parametric 3D building models
- Creation of synthetic point clouds
- Training and evaluation of a machine learning model
- Reconstruction of 3D buildings based on classified roof types.

Parametric building models are generated in CityEngine and FME using CGA rules, where parameters such as roof type, building height, and roof height are varied. The generated models are converted into synthetic point clouds, rasterized into image representations, and used to train a ResNet-based deep learning model. Real-world LiDAR data from Gothenburg were used for testing.

The results show that the model achieves high classification accuracy when both training and testing are performed on synthetic data. However, the accuracy decreases significantly when the model is applied to real-world LiDAR data, indicating that generalization between synthetic and real data remains a major challenge. The study also demonstrates that certain roof types are difficult to distinguish due to similar geometric characteristics, variations in real building structures, and differences between synthetic and real data. The classified roof types were subsequently linked to parametric roof descriptions and used to reconstruct simplified 3D building models.

In conclusion, the study demonstrates that synthetically generated training data combined with parametric modeling and machine learning has potential for automated roof classification and parametric representation of buildings from point cloud data. At the same time, improved generalization towards real-world data and more representative training datasets are identified as important areas for future development.

Sammanfattning

Tredimensionella byggnadsmodeller blir allt viktigare inom områden såsom stadsplanering, simuleringar och geografiska analyser. Samtidigt saknas ofta detaljerad information om byggnaders takstrukturer i befintliga geodata, vilket försvårar automatiserad rekonstruktion av sammanhängande och användbara 3D-modeller. Punktmolnsdata från laserskanning möjliggör detaljerad geometrisk representation av byggnader, men klassificering av takstrukturer från punktmoln är fortfarande en komplex och resurskrävande process. Dessutom begränsas användningen av maskininlärning inom området av bristen på annoterade träningsdata.

Detta examensarbete syftar till att undersöka möjligheten att klassificera verkliga takstrukturer baserat på punktmolnsdata genom att använda en maskininlärningsmodell tränad på syntetiskt genererade data. Arbetet undersöker även hur taktyper kan beskrivas parametriskt och användas för rekonstruktion av förenklade 3D-byggnadsmodeller.

Metoden är uppbyggd kring fyra huvudsakliga steg:

- Generering av parametriska 3D-byggnadsmodeller
- Skapande av syntetiska punktmoln
- Träning och utvärdering av en maskininlärningsmodell
- Rekonstruktion av 3D-byggnader baserat på klassificerade taktyper.

Parametriska byggnadsmodeller genererades i CityEngine och FME med hjälp av CGA-regler där parametrar såsom taktyp, byggnadshöjd och takhöjd varierades. De genererade modellerna omvandlades därefter till syntetiska punktmoln som rasteriserades till bilder och användes för träning av en ResNet-baserad djupinlärningsmodell. För testning användes riktig LiDAR-data från Göteborg.

Resultaten visar att modellen uppnår hög klassificeringsnoggrannhet när både träning och testning utförs på syntetiska data. Däremot minskar noggrannheten tydligt när modellen tillämpas på riktig LiDAR-data, vilket visar att generalisering mellan syntetiska och verkliga data fortfarande är en stor utmaning. Studien visar även att vissa taktyper är svåra att skilja åt på grund av liknande geometriska egenskaper, variationer i verkliga byggnadsstrukturer och skillnader mellan syntetiska och verkliga data. De klassificerade taktyperna kunde därefter kopplas till parametriska takbeskrivningar och användas för att rekonstruera förenklade 3D-byggnadsmodeller.

Sammanfattningsvis visar studien att syntetiskt genererade träningsdata i kombination med parametrisk modellering och maskininlärning har potential för automatiserad klassificering och parametrisk representation av byggnader från punktmolnsdata. Samtidigt framgår att förbättrad generalisering mot verkliga data och mer representativa träningsdata är centrala områden för fortsatt utveckling.

Table of contents

List of abbreviations

2D	-	Two-dimensional
3CIM	-	3D-City Information Model
3D	-	Three-dimensional
AI	-	Artificial Intelligence
ALS	-	Airborne Laser Scanning
CAD	-	Computer-Aided Design
CGA	-	Computer Generated Architecture
CityGML	-	City Geography Markup Language
CNN	-	Convolutional Neural Network
CSV	-	Comma Separated Values
DEM	-	Digital Elevation Model
DINO	-	DIstillation with NO labels
DL	-	Deep Learning
DNN	-	Deep Neural Network
DSM	-	Digital Surface Model
EO	-	Earth Observation
ESRI	-	Environmental Systems Research Institute
ETL	-	Extract, Transform, Load
EU	-	European Union
F1-score	-	Harmonic mean of precision and recall
FME	-	Feature Manipulation Engine
GIS	-	Geographic Information Systems
GML	-	Geography Markup Language
GNSS	-	Global Navigation Satellite System
IBM	-	International Business Machines Corporation
IMU	-	Inertial Measurement Unit
INS	-	Inertial Navigation System
Inspire	-	Infrastructure for Spatial Information in Europe
JSON	-	JavaScript Object Notation
LAS	-	LASer file format
LiDAR	-	Light Detection and Ranging
LIR	-	Largest Inscribed Rectangle
LOD	-	Level of Detail
ML	-	Machine Learning
NGP	-	Nationell Geodataplattform (Swedish National Geodata Platform)
NH	-	Nationell Höjdmodell (Swedish National Elevation Model)
NPR	-	Neural Procedural Reconstruction
NS	-	Nationell Specifikation (Swedish National Specification)
OGC	-	Open Geospatial Consortium

PM	-	Procedural modeling
RANSAC	-	Random Sample Consensus
ResNet	-	Residual Network
ReLU	-	Rectified Linear Unit
RGB	-	Red, Green, Blue
RMSE	-	Root Mean Square Error
RPK	-	Rule Package
TLS	-	Terrestrial Laser Scanning
YOLO	-	You Only Look Once

List of figures

Figures Chapter 1

Figures Chapter 2

Figures Chapter 3

Figures Chapter 4

Figures Chapter 5

List of tables

Tables Chapter 2

Tables Chapter 3

Tables Chapter 4

Part I

Introduction

1. Introduction

1.1 Background

Three-dimensional (3D) building models have been proved to be very useful for urban planning, simulations, analysis and visualization. Such models are essential for decision-making in planning, design and infrastructure management. A key component in creating accurate and semantically rich 3D building models is correct identification of roof types and façade elements. In this report, focus is on reconstruction of building models from point clouds. The conventional 3D reconstruction methods rely heavily on the quality of the data source and manual work (Buyukdemircioglu, 2022). Point cloud data is well-suited for generating these models because it captures geometry accurately, however transforming point clouds into a coherent building model can be challenging, particularly when integrating elements such as walls and roofs into a consistent representation (Ding et al. 2025; Sajadian & Arefi, 2014).

A 3D city model describes an urban area through three-dimensional geometric representations of city elements, where buildings usually are the dominant component. There are several methods to create 3D city models, including photogrammetry, laser scanning, extrusion from 2D footprints, architectural models and drawings (Biljecki et al., 2015). In addition to these methods, parametric modeling can be used to generate 3D city models by applying rule-based systems that define urban structures and building characteristics. In parametric modeling, building geometry is generated from predefined parameters, such as building height, roof height, and roof type. By modifying these parameters, different building configurations can be created automatically, making the method suitable for generating large numbers of building models efficiently (Badwi, Ellaithy & Youssef, 2022; Liu, 2024). Traditionally, 3D city models were primarily used for visualization purposes. However, as technology has developed, their applications have expanded to support urban planning, spatial analyses, environmental simulations, and decision-making processes (Syed Abdul Rahman et al., 2024).

A 3D building model can appear differently depending on how much detail is included in its representation. To describe how much detail is incorporated in a model, the concept Level of Detail (LOD) is used. CityGML is an open international standard for 3D city models that defines a range of LOD, the lowest (LOD 0) being a 2D building footprint and the highest (LOD 4) including façade details and interior building structures (Biljecki et al., 2016). A suitable level of detail for classifying common building attributes, such as roof types and the number of windows, is LOD3, as it includes detailed roof geometry along with exterior façade elements (Figure 1.1). LOD3 building models support a wide range of applications, including urban digital twins, flood scenario simulations, and analyses of solar potential. They are also essential for estimating building energy demand due to their detailed representation of façade geometry (Hanke et al. 2025; Yarroudh et al. 2024).

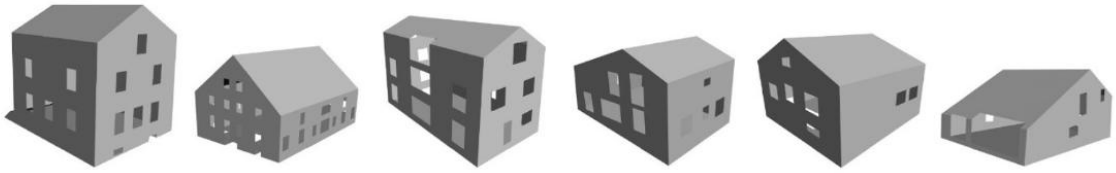


Figure 1.1: Examples of LOD3 building models (Pantoja-Rosero et al., 2022, p. 12)

As previously mentioned, point cloud data can be used for 3D building reconstruction and is commonly acquired using laser scanning techniques. The foundation of laser scanning is that a sensor, airborne or terrestrial, emits a laser beam that is reflected on a surface and then returns to the sensor. By measuring the travel time of the laser pulse and the scanning angles, the distance and direction to each point can be calculated, producing a three-dimensional point cloud (Wehr & Lohr, 1999; Shen et al., 2023).

Despite its advantages, laser scanning-based reconstruction faces several challenges. Point cloud data often require complex segmentation of ground, buildings and vegetation, and large point densities result in datasets that are costly and time-consuming to process. Reconstruction quality is also highly dependent on point density and accuracy, while fine façade elements such as windows remain difficult to extract using purely point cloud-driven approaches (Sajadian & Arefi, 2014; Ding et al., 2025). These limitations highlight the potential benefits of integrating parametric modeling as a complementary approach, as rule-based building generation can help create more consistent building representations despite incomplete or noisy point cloud data.

As aforementioned, there are many possible methods to create a 3D building model. Traditional methods often combine Digital Surface Models (DSM) with building footprints and point clouds, or algorithms such as Random Sample Consensus (RANSAC). Newer methods involve machine learning and deep learning approaches, which have shown potential to automate parts of the 3D reconstruction process and address limitations of conventional techniques. Machine learning methods have been increasingly used for 3D reconstruction and are suitable for this task, as they are able to learn complex spatial patterns that allow for robust reconstruction (Buyukdemircioglu, 2022).

Identifying and categorizing building roof types is an important task within e.g. urban planning and disaster management. However, this can be a complicated task to achieve using machine learning because of the lack of labeled data (Mutreja & Bittner, 2025). Although geographic information systems (GIS) provide detailed information about terrain, infrastructure and building footprints, important attributes such as roof structure are often missing from current datasets. As a result, labeled data for roof types are limited or not available, especially at large scales. Creating such datasets typically requires combining sources like airborne LiDAR with manual annotation, which is time-consuming and resource demanding. The scarcity of annotated data presents a significant challenge for developing machine learning methods for roof classification (Castagno & Atkins, 2018). This study proposes an alternative approach to this problem, by training a machine learning model on synthetically generated point cloud data derived from parametric modelling, based on an idea developed by Sweco.

Figure 1.2 illustrates the general methodology used to achieve this approach. First, a script (explained in Section 2.2.1) is developed to generate different parametric building models. These models are defined based on parameters such as roof type, building height, and roof height. From these parametric models, synthetic point cloud data is generated and used as training data for the machine learning model. For testing, real point clouds are used. During the machine learning process, hyperparameter tuning and model evaluation are performed iteratively to improve performance. Hyperparameter tuning involves adjusting settings that control how the model learns, such as the learning rate, batch size, and number of training epochs. Each modification is tested and evaluated to identify the optimal model configuration. The final outcome is a trained machine learning model capable of classifying roof types and enabling the reconstruction of solid roof surfaces.

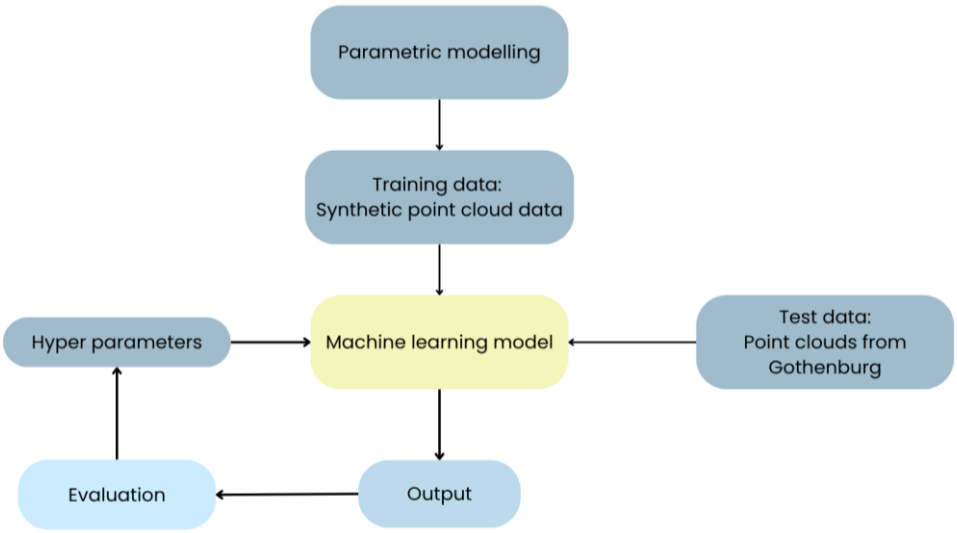


Figure 1.2: Illustration of the workflow

1.2 Problem Statement

Today 3D building models have many areas of application, including visualization, analysis and urban planning. Point clouds are one of the most common data formats used to create 3D representations of buildings. Generating detailed building models from point clouds is challenging, especially at higher levels of detail, which could include roof and wall structures (Bello et al., 2020). Current methods often fail to capture these details efficiently, making it difficult to analyze and use the point cloud data efficiently. Some applications, such as lighting simulations and visibility assessments, require geometrically detailed and semantically consistent 3D building models, which can be difficult to achieve using raw point cloud data alone (Yarroudh et al., 2024).

Göteborgs Stad has 3D roof models that are geometrically detailed and accurate. The current models consist of multiple polygons that are not fully connected, resulting in a non-coherent structure (see Figure 1.3). While the models demonstrate a high level of geometric detail, they lack a continuous and solid surface. This leads to gaps and inconsistencies, meaning the models do not fully represent real-world roof behavior, such as shadow casting or water flow across surfaces.

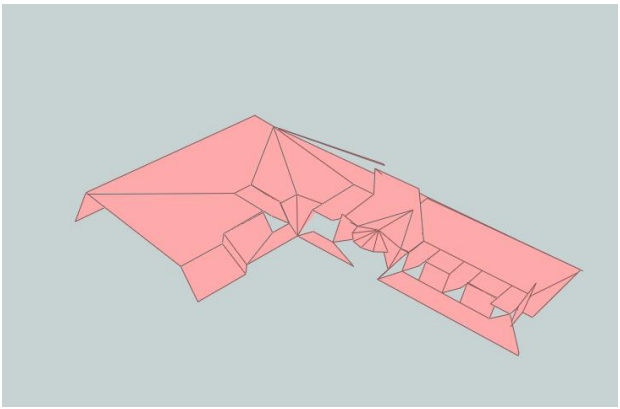


Figure 1.3: An example of a 3D roof model (Göteborgs Stad)

One possible approach is to classify roofs based on their specific roof type. By identifying the correct class, it becomes possible to reconstruct a solid roof surface that closely resembles reality. While some geometric detail may be lost in this process, the functional aspects are preserved. This enables the models to be used more effectively in analyses such as shadow simulation and stormwater management.

Roof structures represent one of the most important components of detailed building models, as they strongly influence both structural behavior and analyses. In particular, roof type is a critical building characteristic for tasks such as wind vulnerability modeling, where different roof geometries respond differently to wind loads. Despite its importance, roof type information is often missing from publicly available building databases, which typically only include basic attributes such as building footprint or height. This lack of detailed roof information limits the usefulness of current datasets and makes it difficult to perform large-scale analyses. Consequently, there is a need for automated approaches that can extract and classify roof types from available data sources, such as point clouds, to enrich building models and support downstream applications (Meng et al., 2023).

In addition, the lack of large, well-annotated point cloud datasets limits the effective use of deep learning, as manual labeling is time-consuming and costly (Wichmann et al., 2018). This motivates the use of alternative strategies, such as using synthetically generated point clouds derived from parametric models, which provide known ground truth for training machine learning approaches (Zeng et al., 2018).

1.3 Aim and Research Questions

The aim of this study is to classify real roof structures based on point cloud data and represent them using parametric descriptions to support the reconstruction of LOD2 building models. To achieve this, synthetically generated 3D building models are converted into point clouds and used to train a machine learning model. The trained model is then applied to real-world point cloud data in order to extract parametric representations of the roof structures.

The research questions are:

- How have previous studies approached detection and classification of roof structures and façade elements?

- How can roof types be classified by a machine learning model trained on synthetic point cloud data?

- How can roofs be described parametrically based on point cloud data?

1.4 Limitations

In this study, both façade elements and roof structures are addressed in the theoretical part and further discussed throughout the analysis. However, for the practical part of the study, the focus is restricted to the classification of roof structures, while façade elements are excluded.

This study focuses on classifying roof structures using a machine learning approach. The roof types considered are: Monopitch, Mansard, Hipped, Gabled & Flat. Other roof types or combinations thereof are not included, in order to reduce the complexity of the classification problem and ensure that the project remains feasible within the limited timeframe of this master's thesis. The testing of the machine learning model is limited to a relatively small area in Gothenburg where the number of roof types is limited as well.

1.5 Disposition

The report is structured into four parts: Introduction, Theoretical Background, Methodology, and a final part containing the Analysis and Conclusion.

First, the theoretical framework is presented through a literature review, providing an overview of topics related to the practical component of the study, including laser scanning, parametric design, 3D models, and machine learning.

Next, the focus shifts to the practical implementation of the study. Parameterized 3D building models are generated using various software tools, transformed into point clouds, and subsequently used as synthetic input data for training and evaluating a machine learning model.

Chapter 1 introduces the background of the study and describes the problem addressed in this research. It presents the aim of the project and the research questions, along with the limitations and the overall structure of the report.

Chapter 2 provides a literature review covering 3D building models and the standards describing their Levels of Detail. Parametric design is introduced as an approach for generating and structuring 3D building models based on predefined parameters. The chapter also addresses point cloud data and its application in 3D building reconstruction, as well as the different methods used to achieve this. In addition, fundamental concepts of machine learning are

presented, forming an essential theoretical foundation for the practical component of the study.

Chapter 3 describes the dataset used, the softwares applied and the overall methodology adopted to achieve the project aim. This includes generating training data, developing a machine learning model, and training and testing the model using various tools.

Chapter 4 presents the results of the study along with the evaluation metrics.

Chapter 5 provides a discussion of the results, including an evaluation of the methodology outcomes, as well as considerations for future improvements and practical applications.

Chapter 6 concludes the study and summarizes its main contributions.

Part II

Theoretical Background

2. Literature Review

This literature review introduces the key thematic areas addressed in this report. The chapter begins with an overview of laser scanning technologies, outlining the principles of data acquisition and the generation of point cloud data. It then presents background information on 3D city models and 3D building models, including a description of the most commonly used standard for representing their Levels of Detail (LOD).

Subsequently, the chapter explores parametric design as a methodology for creating and controlling 3D models through rule-based and parameter-driven approaches. An introduction to machine learning and deep learning is then provided, covering fundamental concepts as well as commonly used model types.

Finally, the chapter reviews existing approaches for roof classification and 3D building reconstruction from point cloud data, including both conventional and more recent machine learning-based methods. Together, these sections establish the theoretical and technical foundation for the research presented in the remainder of this report.

2.1 Laser Scanning

One of the most common methods for acquiring three-dimensional point clouds for 3D building modeling is laser scanning. In remote sensing and geospatial applications, this technology is often referred to as LiDAR (Light Detection and Ranging), which uses laser pulses to measure distances and generate three-dimensional point clouds. Laser scanning involves a sensor, mounted on an airborne or terrestrial platform, emitting a laser beam that is reflected from a surface, such as the ground or building façades, back to the sensor. The distance to an object is determined from the time difference between the emitted and returned laser signal. Together with the scanning angles, each laser hit can be positioned in a local three-dimensional coordinate system, which can be transformed into a global coordinate system using positioning and orientation data. Laser scanning systems are fundamentally composed of a laser ranging unit, an opto-mechanical scanning mechanism for directing the laser beam, and control components for positioning and orientation (Wehr & Lohr, 1999; Shen et al., 2023).

2.1.1 Airborne Laser Scanning

Airborne Laser Scanning (ALS) is a type of laser scanning where the sensor is mounted on an airborne platform, such as an airplane or a drone. As the platform moves over an area, laser pulses are emitted towards the ground, and the returned signals are used to measure distances. This enables efficient mapping of large areas and makes ALS particularly suitable for terrain and height modeling (Wehr & Lohr, 1999).

During data acquisition, the laser beam is systematically swept across the ground surface while the aircraft moves forward. Each emitted pulse results in one or several recorded returns, and repeated measurements are combined to form a three-dimensional point cloud representing the terrain and surface objects (Wehr & Lohr, 1999).

To determine the geographic position of each point, ALS systems use an integrated positioning and orientation system consisting of a Global Navigation Satellite System (GNSS) receiver for positioning and an inertial measurement unit (IMU) for orientation (Wehr & Lohr, 1999). The overall data acquisition geometry and positioning principle are illustrated in Figure 2.1.

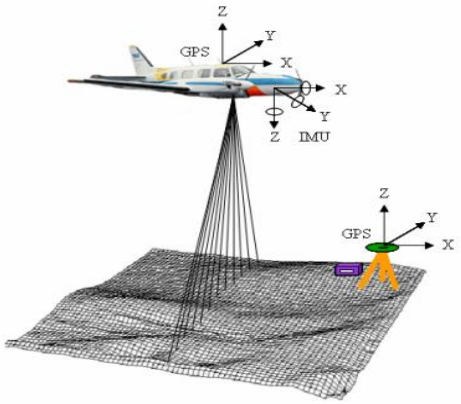


Figure 2.1: Principle of Airborne Laser Scanning (Lantmäteriet, 2021a, p. 254)

2.1.2 Terrestrial Laser Scanning

Terrestrial Laser Scanning (TLS), also referred to as ground-based laser scanning, is used to acquire high-resolution three-dimensional point clouds of objects and environments from ground level. The sensor is mounted on a stationary platform, and the laser beam is systematically swept across the object surface, enabling a detailed three-dimensional representation. By measuring the distance and the scanning angles, each laser hit can be positioned in a local three-dimensional coordinate system. The geometric measurement principle of TLS is illustrated in Figure 2.2. TLS is capable of achieving millimeter- to sub-millimeter-level precision under favorable conditions (Shen et al., 2023).

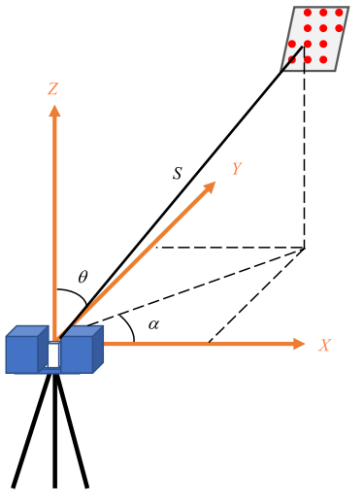


Figure 2.2: Principle of Terrestrial Laser Scanning (Shen et al., 2023, p. 6)

TLS systems are typically operated from fixed positions at close to medium ranges. This measurement configuration enables very high geometric detail and makes TLS particularly suitable for documenting objects with complex shapes (Shen et al., 2023).

Because TLS measurements are acquired from stationary scanner positions, parts of an object may be occluded, resulting in shadowed areas in the point cloud. To achieve complete coverage, multiple scans from different positions are often required and subsequently combined through registration. Registration can be performed using reference targets, cloud-to-cloud matching based on overlapping geometry, or by georeferencing the scans to an external coordinate system (Shen et al., 2023). The principle of scan registration through overlapping geometry and coordinate transformation between scanner positions is illustrated in Figure 2.3.

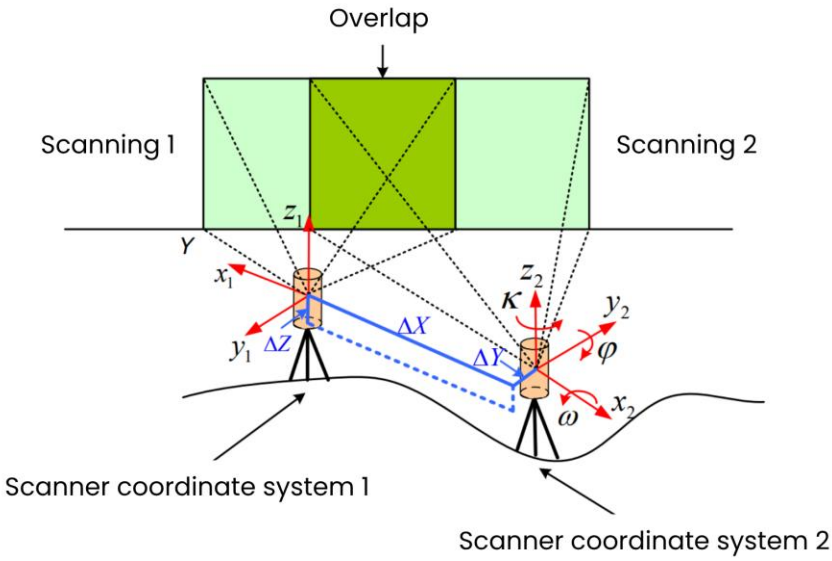


Figure 2.3: Registration of two overlapping point clouds. Redrawn and translated from (Lantmäteriet, 2021a, p. 271)

Overall, TLS is characterized by high geometric resolution and accuracy, as well as a detailed representation of vertical surfaces such as façades. TLS also offers flexible measurement planning, but has limited spatial coverage per scan and typically requires post-processing and registration when multiple scan positions are used. These characteristics make TLS particularly well suited for detailed documentation of buildings, façades, and technical structures (Shen et al., 2023).

2.1.3 Point Density

In ALS, point density is primarily controlled by laser system parameters, flight parameters, and scanning geometry. During data acquisition, the laser beam is swept across the ground surface at a predetermined scanning frequency, and repeated measurements are used to achieve the desired point density. Key factors influencing the spatial distribution of laser points include flight altitude, scan angle, aircraft speed, pulse repetition frequency, and the scanning mechanism of the sensor. In general, lower point densities are obtained at higher flight altitudes

and wider scan angles, whereas higher point densities are achieved at lower flight altitudes combined with higher pulse repetition frequencies. Consequently, the final point density represents a balance between data coverage, productivity, and the level of geometric detail required for a specific application (Baltsavias, 1999; Wehr & Lohr, 1999).

Point density in TLS is generally high compared to ALS, but it varies substantially within a dataset. Typical point densities in ALS range from approximately 0.5–30 points/m^2, whereas TLS datasets may exceed 1000 points/m^2 depending on the scanning setup (Lantmäteriet, 2015; Lantmäteriet, 2017). The density of points in TLS datasets is primarily controlled by scanner settings and measurement geometry, including angular resolution, scanning pattern, and the distance between the scanner and the object (Shen et al., 2023).

TLS systems emit laser beams at fixed angular increments in both horizontal and vertical directions. Consequently, points are uniformly distributed in angular space but not in object space. With increasing distance from the scanner, the spacing between adjacent points on the object surface increases geometrically, leading to a decrease in point density with distance (Chen et al., 2021).

This distance-dependent behavior means that objects located close to the scanner are represented with very high point densities, often at millimeter-level spacing, depending on angular resolution and range. Objects located at distances beyond approximately 100-150 meters may be represented at decimeter-level spacing, even within the same dataset (Chen et al., 2021). Such variations are inherent to the TLS measurement geometry and do not indicate reduced measurement quality, but rather reflect the angular sampling principle of the scanner.

The non-uniform point density has important implications for data analysis. Near-range objects allow detailed geometric modeling and feature extraction, while far-range objects may exhibit reduced geometric detail. Consequently, TLS-based analyses often need to account for local variations in point density when selecting processing methods and interpretation strategies (Shen et al., 2023).

2.1.4 Sources of Error

Distance measurements in laser scanning systems are subject to range errors, which affect the positional accuracy of recorded points (Baltsavias, 1999). In addition to range precision, overall point cloud accuracy is influenced by errors in sensor positioning (GNSS) and sensor orientation (INS/attitude). In airborne laser scanning, positioning and orientation errors often contribute more significantly to total coordinate uncertainty than pure range measurement errors (Baltsavias, 1999; Wehr & Lohr, 1999).

Additional factors that can affect the overall accuracy include time synchronization errors between sensors and transformation errors between different coordinate systems. These errors influence both planimetric accuracy (X, Y), and vertical accuracy (Z). In general, range errors mainly influence the height component, while orientation errors become more significant at

higher flight altitudes and larger scan angles. Typical attitude errors on the order of 0.01° can result in planimetric errors of approximately 15-20 cm at a flight altitude of about 1000 m, and this error increases proportionally with flight altitude. Planimetric errors therefore tend to be larger than height errors at small scan angles and high flight altitudes (Baltsavias, 1999).

Overall accuracy depends on flight altitude, scan angle, and flight direction. Although higher flight altitudes and wider swath widths increase productivity, they also place greater demands on the accuracy of sensor orientation and typically result in a lower point density (Baltsavias, 1999).

In TLS data, additional uncertainties may arise from occlusions caused by blocking objects, which can lead to incomplete surface representation. Furthermore, when multiple TLS scans are combined, small registration errors between scan positions may introduce geometric inconsistencies in the final point cloud (Chen et al., 2021).

2.1.5 National Use in Sweden

In Sweden, airborne laser scanning (ALS) forms the basis of the national elevation data infrastructure. Lantmäteriet (the Swedish mapping, cadastral and land registration authority) conducts national ALS programs that have resulted in openly available LiDAR datasets, including Laserdata Nedladdning (NH) and Laserdata Skog (Lantmäteriet, 2022; Lantmäteriet, 2024a). The earlier NH dataset provides nationwide coverage of Sweden with a typical point density of approximately 0.5-1 points/m² (Lantmäteriet, 2022). In contrast, the more recent Laserdata Skog dataset is being gradually implemented and is expected to cover approximately 75% of Sweden's land area, providing higher point densities of around 1-2 points/m² (Lantmäteriet, 2024a). These datasets support applications such as digital terrain modeling, forest attribute mapping, infrastructure planning and urban analysis (Lantmäteriet, 2022; Lantmäteriet, 2024a).

In addition to national datasets, municipalities often carry out their own airborne laser scanning to support local applications. These surveys are typically conducted at lower flight altitudes, resulting in higher point densities compared to national data. This allows for more detailed mapping of urban environments, including buildings, infrastructure, and fine-scale terrain features. However, unlike national LiDAR datasets, municipal data are not standardized, and their specifications may vary depending on local requirements and technical systems (Harrie, 2024). For example, LiDAR data from Stockholm are collected with point densities of at least 12–16 points/m² depending on the year, while data from Gothenburg are acquired at around 10 points/m², illustrating the variation in municipal specifications (Stockholms stad, 2024; Göteborgs Stad, n.d.).

In contrast to the nationwide ALS program, terrestrial laser scanning (TLS) in Sweden is primarily applied in project-based, local surveys. According to the Swedish Land Survey's TLS handbook (Lantmäteriet, 2021b), TLS is commonly used for detailed documentation of buildings, façades, infrastructure and cultural heritage objects, where higher geometric

resolution is required. TLS surveys are typically commissioned by municipalities, infrastructure agencies or private actors and follow standardized guidelines for planning, quality control and documentation (Lantmäteriet, 2021b).

Unlike ALS, TLS point density is not defined by a fixed national standard but is determined by project-specific resolution and accuracy requirements. Scanner resolution and object distance control the effective point spacing, which can reach millimeter-level spacing at short ranges. TLS is therefore well suited for capturing fine structural details that may not be sufficiently represented in airborne datasets (Lantmäteriet, 2021b).

Together, ALS and TLS provide complementary laser scanning (LiDAR) data in Sweden: ALS offers consistent nationwide coverage, while TLS enables high-resolution documentation of specific structures and urban elements.

2.2 Parametric Modeling

The basic idea of parametric modeling is to construct models based on a set of variables or parameters rather than fixed, explicit measurements. These parameters control the shape, size, and appearance of the model (Liu, 2024). Instead of modeling each geometric element individually, a system of variables, rules, and relationships defines how different parts of the model depend on and interact with one another (Woodbury, 2010). When a parameter is modified, the entire model is automatically updated, enabling rapid adjustments, flexibility across different scenarios, and efficient optimization and comparison. In this sense, parametric modeling functions as a flexible design system rather than a static representation (Liu, 2024)

A fundamental principle of parametric modeling is that the model is organized as a network of linked components. Each component can be represented by a node that either stores a value or performs a procedure (see Figure 2.4). The relationships between nodes describe the flow of information through the model and ensure that changes spread in a controlled and predictable manner. Consequently, adjusting parameters such as width, height, or spacing does not only affect a single element but may influence the structure of the entire model (Woodbury, 2010).

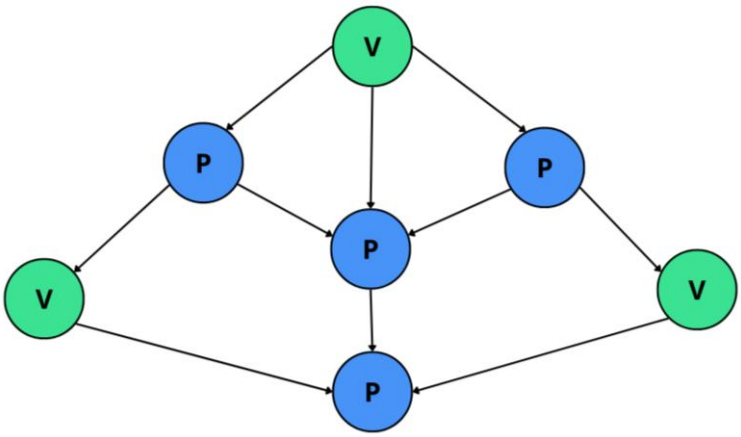


Figure 2.4: Node-based structure of a parametric model. V represents a value node, and P represents a procedure node. Illustration based on (Woodbury, 2010, p. 13).

Parametric models are typically constraint-based, meaning that relationships such as equality, proportionality, alignment, and distance are explicitly defined. These constraints encode design intent and preserve essential geometric conditions even when parameters change (Woodbury, 2010). This allows designers to modify individual parameter values and immediately observe their impact on other aspects of the model. New design alternatives can therefore be generated without rebuilding the model from scratch, as illustrated in Figure 2.5. In other words, parametric modeling allows the designer to modify the whole model, not just individual elements (Badwi, Ellaithy & Youssef, 2022).

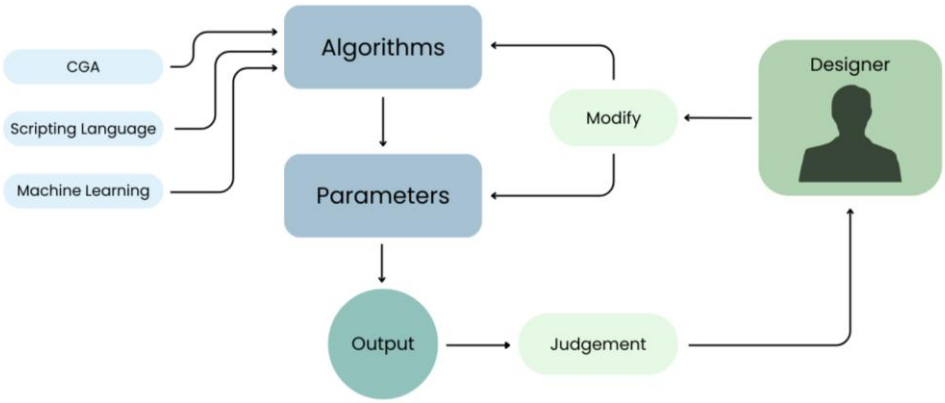


Figure 2.5: Conceptual workflow of parametric modeling. Simplified from (Badwi, Ellaithy & Youssef, 2022, p. 327)

In urban parametric modeling, a wide range of parameters can be defined, including building specifications, building types, coordinate parameters, and indoor characteristics. Building specifications typically include building area, covering both above- and underground sections, as well as building height. Building types may describe functional use, material properties, and layout characteristics. Spatial positioning is controlled through coordinate parameters (*X, Y, Z*), while indoor parameters such as interior layout and window area can be incorporated when a higher level of detail is required (Liu, 2024).

2.2.1 CityEngine

CityEngine is a software platform that integrates GIS with parametric modeling to generate 3D building models. It was originally developed by Procedural Inc. and later acquired by the Environmental Systems Research Institute (ESRI). The software utilizes Computer Generated Architecture (CGA), a rule-based scripting language, to generate 3D building models. A CGA rule can specify key attributes such as number of floors, floor heights, façade composition, roof structure, and overall building form (Badwi, Ellaithy & Youssef, 2022).

CityEngine provides a framework that enables the generation of 3D building models from 2D GIS data through the application of CGA rules to building footprints. This workflow supports efficient creation and modification of building geometries based on parameter values rather than manual modeling. In addition to the main building structure, supplementary elements such

as façade details and textures can be generated within the same environment (Badwi, Ellaithy & Youssef, 2022)

The CityEngine workflow can be divided into four main stages: importing 2D GIS data, defining building parameters, developing CGA scripts, and applying these scripts to the building footprints. The resulting models can be exported to other software platforms for visualization and further analysis. Furthermore, different design alternatives or development stages can be stored as separate scenes, which supports iterative design exploration and evaluation. Previous studies indicate that CityEngine is a cost-efficient and powerful tool for parametric urban modeling compared to many other applications (Badwi, Ellaithy & Youssef, 2022).

2.2.1.1 Procedural Modeling

Procedural modeling, a form of parametric modeling, is based on users writing rule-based code that defines, step by step, how an object is constructed rather than assembled manually. These rules describe how geometric elements are generated and organized within a model. Because the same rule set can be applied to multiple objects, procedural modeling enables high efficiency, consistency, and scalability across large numbers of models (Badwi, Ellaithy & Youssef, 2022).

Procedural modeling can be regarded as a programming paradigm in which complex 3D structures are generated through procedures, functions, and rule sets. Parameters and variables guide the generation process, allowing systematic control over geometry and appearance and enabling the automatic creation of variations by modifying parameter values. In this context, CGA rules are used to specify sequences of operations, procedures, and functions that generate the desired output (Badwi, Ellaithy & Youssef, 2022).

2.2.1.2 Computer Generated Architecture Rules

All models in CityEngine are generated according to Computer Generated Architecture (CGA) rules that describe how geometry is created step by step. These rules can be classified as standard rules, parameter rules, conditional rules, and random rules. Users may also define custom rules when predefined options are insufficient to meet specific modeling requirements (Liu, 2024).

The building modeling process begins by establishing building properties and selecting or creating appropriate CGA rules. Parameters and variables such as floor height, materials, and window dimensions are then defined. Next, the overall building form is generated based on height and volume, followed by the subdivision of floors, roof, and façade. Finally, textures may be applied using either predefined or user-provided images. Figure 2.6 illustrates an example of how CGA rules are structured and applied in building modeling, together with the resulting visual output (Liu, 2024).

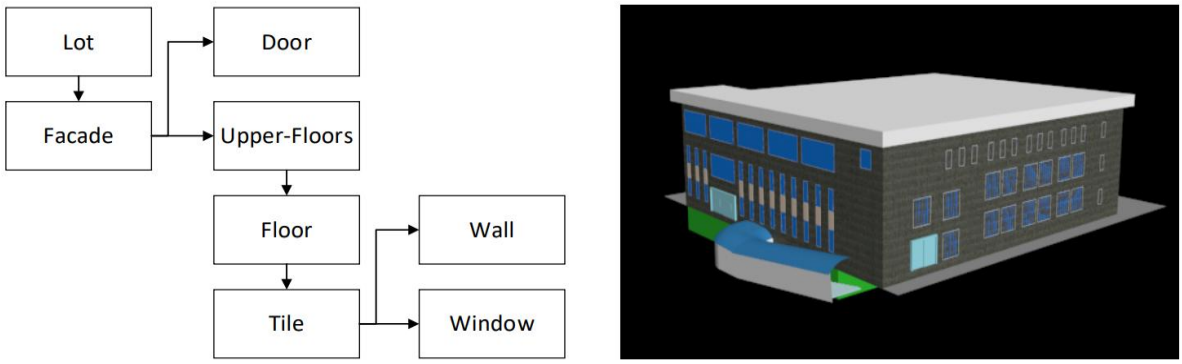


Figure 2.6: Example of CGA rules applied in building modeling and their resulting visual representation (Liu, 2024, p. 9)

The generated models are interactive and can be updated at any time by modifying the underlying rules. A single CGA rule can therefore produce many different 3D building variations. CGA rules can also assign attributes derived from 2D GIS data, such as number of floors, floor height, window and door dimensions, roof type, and wall material (Badwi, Ellaithy & Youssef, 2022).

2.3 3D Building Models

A 3D building model represents the three-dimensional geometry of an individual building. The models include elements such as walls, rooms, openings, roofs and more, and serve many practical and analytical purposes aside from visualization. Their area of application include simulations of lighting, acoustics, thermal or energy performance and fire behavior. This enables assessments of how a design performs under various conditions before construction begins. It can also be beneficial to have 3D models of existing buildings to analyze energy properties or how a potential fire would spread in order to study possible changes to the building. Furthermore, the models can assist in quantitative analysis such as cost estimation, since accurate geometric representations help calculate material quantities (Lewis & Séquin, 1998).

2.3.1 CityGML Buildings

Kolbe et al. (2021) defines CityGML as an open and standardized data model and exchange format designed to represent, store and share semantic 3D city models. It was developed by the Open Geospatial Consortium (OGC) and implemented as an application schema of the Geography Markup Language (GML). The standard enables detailed presentation of both geometry and semantics of urban features such as buildings and infrastructure, making it possible for different systems to interpret and analyze 3D spatial data.

Within CityGML, buildings are described in the Building module, which provides the representation of both thematic and spatial aspects of buildings. In the CityGML model, buildings are represented by the top-level feature type *Building,* which can be subdivided into building parts and further structured into stories and building units, such as apartments.

Buildings in CityGML can also be broken down into smaller structural components and described through different types of semantic surfaces. For example, the exterior of a building may be represented by surfaces such as walls, roof and ground surfaces, while interior spaces can include elements like floors, ceilings and internal wall surfaces. By organizing buildings in this hierarchical and semantically structured way, CityGML makes it possible to represent building geometry and attributes in a form that supports visualization, spatial analysis and the exchange of data between different geographic information systems and urban modeling applications.

2.3.1.1 3CIM

3CIM (3D-City Information Model) is a national 3D city model specification developed collaboratively by Stockholm, Gothenburg and Malmö, together with Lund University. It was created to meet the increased demand for semantically rich 3D geospatial data that can support modern urban applications beyond visualization, such as analysis and simulations (Uggla et al., 2023). In addition to its technical structure, 3CIM was developed to create a harmonized 3C data specification for Swedish municipalities, supporting consistent data management and reusable workflows across cities (Abdi et al. 2023; 3CIM, 2023).

3CIM is built as an extension of the international standard CityGML 2.0 by OGC. It is implemented as an application domain extension to CityGML, meaning that it adds new attributes and object types while preserving the underlying CityGML structure. This allows the model to be used both in standard CityGML tools and in workflows requiring integration with external data sources (Uggla et al., 2023).

The building theme in the 3CIM specification is similar to the CityGML 2.0 (Gröger et al., 2012) definition, but with additional attributes such as identifiers and status information. For example, whether a building is planned or existing can be added via the 3CIM namespace. These enhancements make the model useful for projects where buildings must be tracked and compared with external registers (Uggla et al., 2023).

2.3.1.2 NS Byggnad

NS Byggnad *(Nationell Specifikation Byggnad)* is the national Swedish data product specification for building information within the *Nationella geodataplattformen* (NGP). It defines how information about physical buildings in Sweden should be structured, described and exchanged in digital systems. A building in this context is defined according to the Swedish *Plan- och bygglagen* (PBL2010:900) as a permanent structure with a roof and walls that is intended to be occupied by people and permanently placed on or partly on the ground. The specification applies to buildings throughout Sweden and is maintained by Lantmäteriet (Lantmäteriet, 2024b).

The specification's primary purpose is to establish a standardized, semantically consistent model for building data that can be used across state, municipal and private systems. This

includes details such as the building's geometry, functional purpose based on hierarchical classifications (e.g. residential, industrial, service), status information, floor area and associated metadata. NS Byggnad also incorporates different NS-LOD levels for representing building geometry of varying degrees of detail (Lantmäteriet, 2024b).

Importantly, NS Byggnad is designed to support a wide range of applications throughout the planning, construction, and management of the built environment. Standardized building information can be used as an input for:

- Urban planning and physical infrastructure development
- Building permit processing and decisions
- 2D and 3D visualization and analysis, including modeling at different level of detail
- Statistical and environmental analyses
- Crisis preparedness and emergency response, where accurate and up to date building geometry and attributes improve planning.

NS Byggnad is a part of Sweden's broader strategy to enable digital information exchange in the society's construction process, ensuring that building data can be shared efficiently and consistently between stakeholders (Lantmäteriet, 2024b).

2.3.2 Level of Detail

In 3D building modeling, LOD describes how closely a digital model represents real-world objects in terms of geometry and semantic information. Originating from computer graphics, LOD is used in GIS to define representations of varying complexity, enabling 3D city models to be tailored for different applications, processing requirements and analyses (Biljecki et al. 2014).

In practice, standards define a series of discrete LODs for urban projects. These levels start from simple representations and increase to more complex representations. CityGML is a widely used OGC standard for 3D city models, the standard includes five main LODs (see Figure 2.7):

- LOD0: Basic footprint of an object with very little or no 3D geometry
- LOD1: Basic 3D block models where volumes are roughly approximated
- LOD2: Includes simplified roof shapes and more refined surface representations
- LOD3: Includes detailed architectural structure, such as precise wall geometries, roof details and sometimes façade elements
- LOD4: Extends the external detail of LOD3 with interior structures, including rooms, staircases etc. (Chajaei & Bagheri, 2025)

Figure 2.7: Representation of the LOD of CityGML 2.0 (Biljecki et al. 2016, p. 26)

2.3.3 Roof Type Classification in Inspire and in the Swedish National Specifications of Geospatial Data

Inspire (Infrastructure for Spatial Information in Europe) is an EU directive aimed at improving access to public geospatial data for environmental applications through a spatial data infrastructure within the EU. Inspire defines 34 spatial data themes, of which Building is one. The Inspire Buildings specification is influenced by CityGML 2.0 and shares a similar structure and attribute model, although additional classes and attributes are included (Lantmäteriet, 2024c).

Both Inspire and NS Building define a number of roof types. Table 2.1 presents a mapping between roof types in NS Building and the corresponding roof types in Inspire. Figure 2.8 presents images of the different roof types in Inspire Building.

Table 2.1: Mapping between roof types in NS building and in Inspire Building (Lantmäteriet, 2024c, p. 68)

NS Building	**Inspire Building**
brutet pulpettak	dualPentRoof
bågtak	archRoof
dubbelkrökt tak	hyperbolicParabaloidalRoof
halvvalmat tak	halfHippedRoof
kupoltak	cupolaRoof
kägeltak	conicalRoof
mansardtak	mansardRoof
platt tak	flatRoof
pulpettak	monopitchRoof
pyramidtak	pyramidalBroachRoof
sadeltak	gabledRoof
sågtandtak	sawToothRoof
tälttak	pavilionRoof
valmat tak	hippedRoof

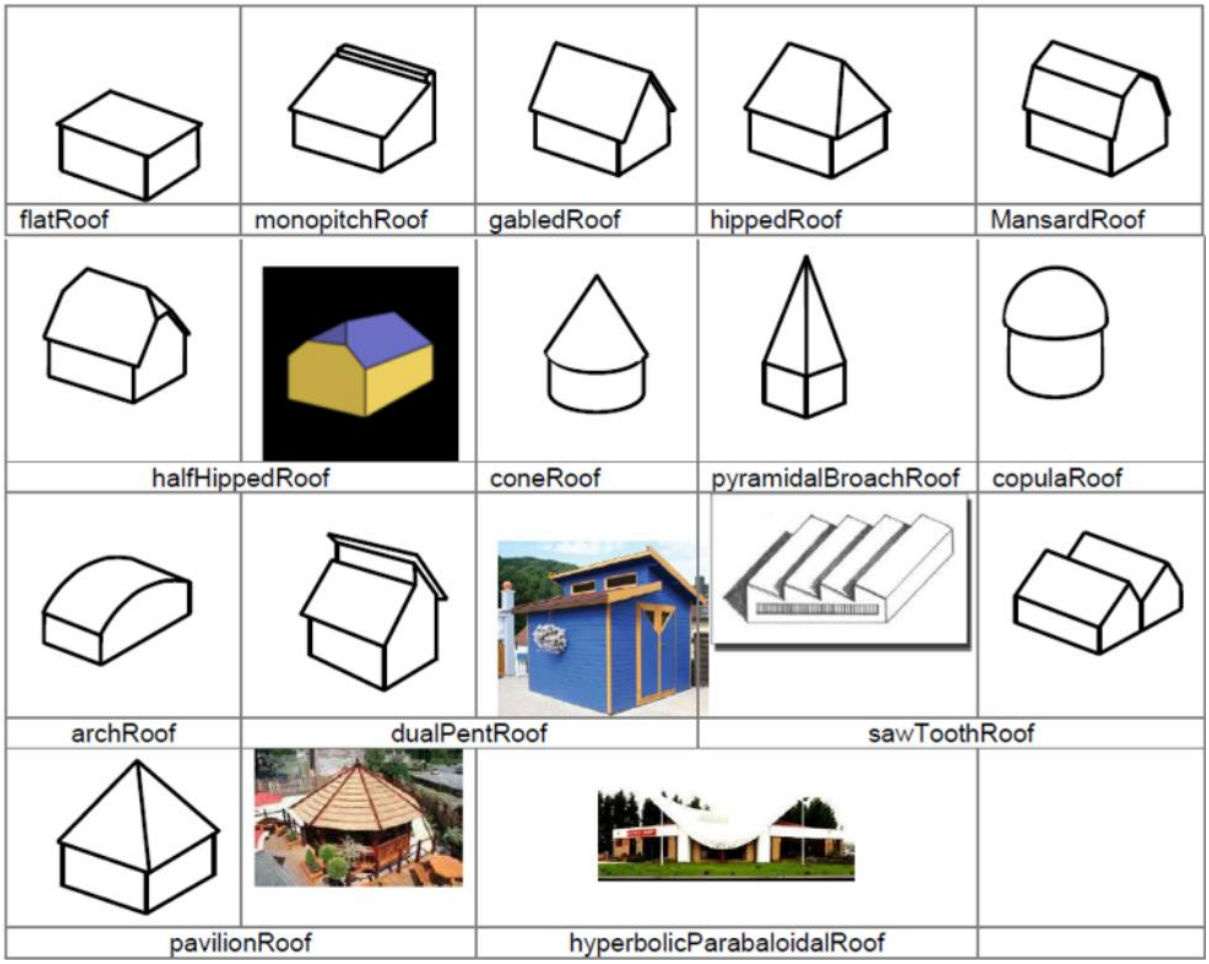


Figure 2.8: Images of each roof type in Inspire Building (Lantmäteriet, 2024c, p. 69)

2.4 Data Integration & Manipulation

Data integration is defined as the process of combining data from multiple sources to create a unified and consistent view of the information. The purpose is to make data more accessible and useful for users and systems, regardless of where it is originally stored. In practice, this involves collecting, processing, and organizing data so that it can be analyzed and used to support decision-making within organizations.

An important aspect of data integration is ensuring data quality by cleaning, sorting, and enriching the information before it is used. This allows data from several sources to be standardized and function together within a common system. In this way, data integration helps reduce issues related to data silos and enables a more comprehensive understanding of organizational activities.

Data integration therefore constitutes a central component in modern information systems and data analytics environments, where the need to manage large and diverse datasets is essential. To implement this process, structured methods are often used, one of the most established being ETL (Extract, Transform, Load) (Microsoft, n.d.).

Data manipulation refers to the process of modifying and transforming data within a dataset to make it suitable for analysis. Common operations include filtering, aggregating, restructuring and reformatting data. These operations allow users to extract relevant information, prepare datasets for specific tasks, and generate insights from the data. Data manipulation is therefore an important component of data analysis and decision-making processes (Eads, 2025).

2.4.1 Extract, Transform, Load (ETL) Process

ETL (Extract, Transform, Load) is a data integration process used to combine, clean, and organize data from multiple sources into a single, consistent dataset. The process consists of three stages: extraction of data from source systems, transformation of the data through operations such as filtering, restructuring, and validation, and loading of the processed data into a target system for storage and analysis. By ensuring data consistency and quality, ETL provides a foundation for data analytics and decision-making (IBM, n.d.-a).

The ETL concept is implemented in tools such as FME (Feature Manipulation Engine), where data is extracted from a source, transformed according to user requirements, and loaded into a destination system (Safe Software, n.d.-b).

2.4.2 Feature Manipulation Engine (FME)

FME (Feature Manipulation Engine) is a data integration tool that was developed as one of the first tools focusing on spatial ETL, meaning that the process is specifically oriented toward the transformation and transfer of geographic data between different formats and systems. It was designed to address limitations in traditional data translation methods, where data was often forced through limited data models, leading to a loss of meaning in the translation process.

FME supports a wide range of data formats and functions as a centralized platform for reading, transforming, and writing geographic data. It uses a rich data model that preserves both geometric and attribute information throughout the integration process, reducing the risk of information loss during translation. The platform also provides extensive capabilities for processing, restructuring, and enriching data, allowing datasets to be adapted to specific requirements and workflows (Safe Software, n.d.-b).

A key feature of FME is its ability to manipulate data during format translation. This is carried out using transformers, which process features by performing operations such as filtering, restructuring, and modifying both geometry and attributes as the data flows through the workspace. Through this approach, data can be gradually transformed and adapted to meet specific requirements (Safe Software, n.d.-a).

2.5 Machine Learning

Machine learning (ML) is a subfield of computer science and statistics that focuses on developing systems that improve their performance automatically through experience rather than through explicit programming. According to Jordan and Mitchell (2015), ML is a core component of modern artificial intelligence and data science research. The fundamental goal of machine learning is to construct algorithms that can learn patterns from data and use those patterns to make predictions or decisions without being manually coded for every possible situation.

The progress in ML over the recent decades has been driven mainly by two factors: the development of new learning algorithms and theoretical advances, and the explosive growth in available data combined with low-cost computational resources. This combination has enabled machine learning methods to scale very large datasets, enabling a large area of application (Jordan & Mitchell, 2015). Machine learning has become widely adopted across a broad range of domains, including healthcare, manufacturing, finance, marketing, and scientific research, where it is used for prediction, automation, pattern recognition, and decision support (Rane et al. 2024).

Modern machine learning includes several approaches. Supervised learning trains models on labeled data, where each input has a known output, allowing the system to predict outcomes for new data. Common tasks include classification, such as identifying whether an image contains a cat or a dog, and regression, such as predicting a numerical value. Unsupervised learning deals with unlabeled data, aiming to discover patterns, groupings or structures within the dataset. An example of this could be clustering. Reinforcement learning involves systems learning to make decisions by interacting with an environment and receiving feedback in the form of rewards or penalties. These different approaches reflect the variety of learning problems that ML can address. Machine learning's influence has grown because it often proves to be easier to train systems by example than to manually specify all rules and behaviors a program must follow. (Jordan & Mitchell, 2015; Zhu et al. 2023).

2.5.1 Machine Learning, Deep Learning and Artificial Intelligence

Artificial intelligence (AI), Machine Learning (ML) and Deep Learning (DL) are closely related concepts, but they do not mean the same thing. The main difference between them can be understood as a hierarchy: AI is the broadest concept, ML is a part of AI and DL is a part of ML (see Figure 2.9) (Jeong, 2020).

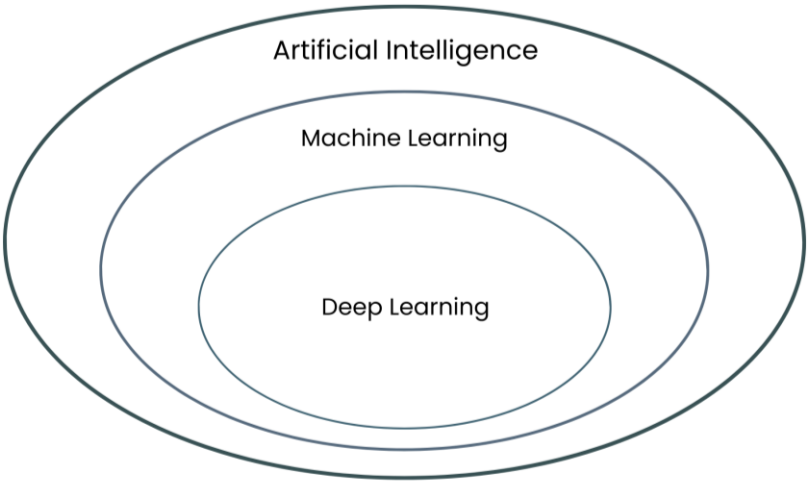


Figure 2.9: The relationship between the terms AI, ML and DL.

Artificial Intelligence refers to computer systems that are designed to perform tasks that normally require human intelligence. AI does not describe one single method, but includes many different approaches such as rule-based systems, logic based systems and learning-based systems. Machine Learning, as aforementioned, is a subfield of AI and includes systems that learn from data by analyzing patterns and making predictions (Jeong, 2020).

Deep Learning is a further specialization within machine learning. It is based on artificial neural networks with many layers, often called "deep" networks. These networks are designed to learn complex patterns by processing information in multiple steps. Deep learning models are especially useful for handling large amounts of unstructured data, such as images, speech and text. Compared to traditional machine learning methods, deep learning can automatically learn important features from raw data, reducing the need for manual feature design (Jeong, 2020; Madakam et al. 2022).

Clarifying the distinction between AI, ML and DL is important because the terms are often used interchangeably in popular discourse, despite referring to different conceptual levels and methodological approaches.

2.5.2 The Learning Process

Machine learning can be understood as a process in which a system uses data to build a model that makes predictions or decisions. According to Jung (2024), machine learning methods are composed of three key components: data, a model and a loss function. Data refers to the examples from which the system learns, models refer to the set of candidate hypotheses used to map inputs to outputs and the loss function quantifies the difference between predicted and actual outcomes, guiding the adaptation of the model over time (Jung, 2024).

This process operates iteratively similar to "trial and error" where a model is repeatedly adjusted to reduce the discrepancy between its predictions and the observed data. The machine learning algorithm selects a hypothesis (a specific mapping from input features to predicted

outputs) from a space of possible candidate hypotheses, and then improves it based on feedback from the loss function, see Figure 2.10 (Jung, 2024).

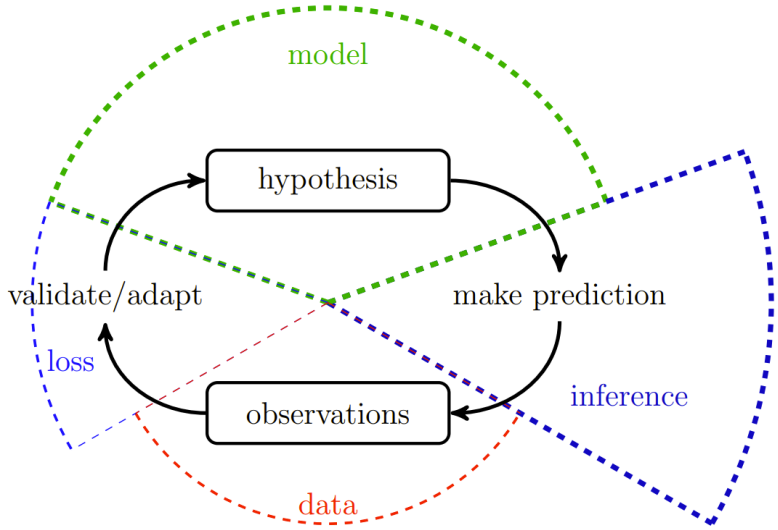


Figure 2.10: The learning process (Jung, 2024, p. 1)

Central to successful learning is the concept of generalization, which refers to a model's ability to perform well on new, unseen data. Du et al. (2025) describe learning as the process of identifying the relationship between input data and output values based on examples, while generalization refers to the model's ability to apply the learned relationship to new data that was not part of the training process. A well-trained model does not merely memorize the training examples, but capture patterns that allow it to make accurate predictions to similar, unseen data (Du et al., 2025).

The difference between fitting the training data and generalizing to new data is a central concern in machine learning. Although the learning process aims to minimize the loss function on the training set, it must also avoid fitting noise or irrelevant patterns that do not represent the actual features that are informative. Techniques such as model selection, regularization and evaluation on separate validation or test sets are used to estimate and improve generalization performance (Du et al., 2025).

In summary, the machine learning process involves using data and a chosen model structure to iteratively reduce prediction error, while ensuring that the resulting model can generalize beyond the specific examples it was trained on. This framework represents many different types of learning algorithms, from simple linear regression to complex neural networks (Jung, 2018; Du et al., 2025).

2.5.3 Overfitting, Underfitting and Data Augmentation

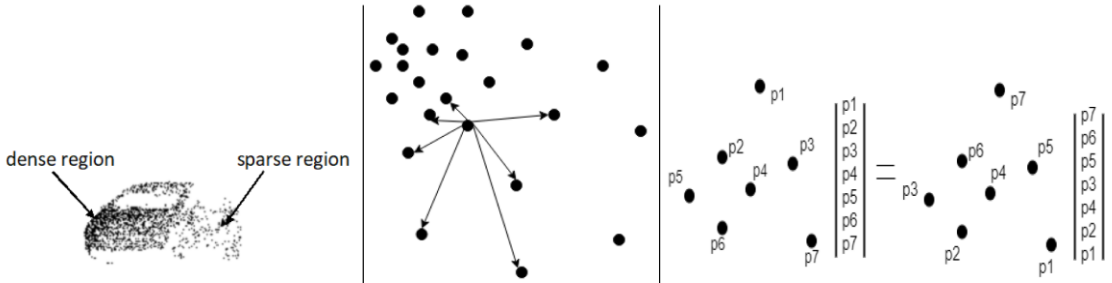
A known challenge in machine learning is ensuring that a model generalizes well to unseen data. Two common problems that can occur during training are overfitting and underfitting. Overfitting happens when a model learns the training data too well, capturing not only the

underlying patterns but also noise or specific details that are not representative of the broader data patterns. As a result, the model performs well on the training data set but poorly on new data. Underfitting occurs when a model is too simple to capture the structure of the data, leading to poor performance on both training and test data (López, 2022).

Data augmentation refers to a set of techniques used to artificially increase the size and diversity of the training dataset by transforming existing data into ways that it preserves its original label. In deep learning, especially for computer vision tasks, augmentation helps models learn more robust representations and reduces overfitting when training data is limited by simulating variations that the model might encounter in real world data. Techniques range from data warping, such as geometric transformations (e.g. rotation, flipping) and color adjustments. These augmentations effectively expand the training distribution, enabling models to generalize better to unseen examples and improving predictive performance on tasks like image recognition and object detection. Data augmentation is a valuable strategy for addressing limited or imbalance datasets by providing additional varied input patterns without requiring new labeled data (Shorten & Khoshgoftaar, 2019)

2.6 Reconstructing 3D Models from Point Clouds

Reconstructing detailed and reliable 3D building models from point cloud data is a complex task that has received significant attention in recent research. While deep learning methods have proven highly effective for structured data arranged on regular grids, point clouds present additional challenges due to their unstructured, unordered, and irregular nature, as well as varying point density and noise (Bello et al., 2020). These characteristics complicate feature extraction and reduce the robustness of both traditional and learning-based reconstruction approaches (Diab et al., 2022). Several semi-automated tools have been developed to structure and preprocess point cloud data, but these approaches often still require manual work or careful parameter tuning (Xiao et al., 2023). Figure 2.11 illustrates some of the main challenges associated with point cloud data.



(**a**) Irregular. Sparse and dense regions. (**b**) Unstructured. No grid; each point is independent and the distance between neighboring points is not fixed. (**c**) Unordered. As a set, point clouds are invariant to permutation.

Figure 2.11: Challenges of point cloud data (Bello et al., 2020, p. 4)

Furthermore, generating high levels of geometric detail, such as roof and wall structures corresponding to NS LOD 2.2 and higher, remains particularly difficult (Bello et al., 2020). Many current methods struggle to efficiently capture such details, which has motivated the

development of both traditional reconstruction techniques and machine learning-based approaches. The following sections therefore review representative methods for 3D building reconstruction from point clouds, including conventional geometry-based techniques and more recent machine learning and deep learning approaches.

2.6.1 Geometric Modeling

Traditional 3D building reconstruction models rely heavily on the quality of the data and source, as well as manual work. The conventional methods for 3D building reconstruction include combining 2D footprints with Digital Surface Models (DSM) or combining an algorithm that can extract planes with point clouds (Buyukdemircioglu, 2022).

Sajadian and Arefi (2014) present a data-driven method for reconstructing 3D building models from LiDAR point clouds. In the study, the authors mention that raw LiDAR data contain several objects such as ground, vegetation and buildings. To be able to successfully reconstruct a building one has to separate building points from the other classes. To do this, the authors proposed a multi-agent extraction and segmentation strategy that uses five criteria to distinguish building points from non-building points. The criteria include height, number of returned pulses, triangle lengths, normal vector directions and area. After segmentation, the method focuses on detecting edge points of the building segments using a method called “Grid Erosion”, followed by RANSAC-based line extraction to approximate building boundaries. Regularization constraints are then applied to refine the extracted lines, so that they represent consistent building outlines. Roofs and walls are modeled by fitting planes to the segmented building points to reconstruct a complete 3D building model.

Sadeghi et al. (2015) present an approach for 3D building façade reconstruction using dense point cloud data acquired from handheld laser scanning systems. This method relies on geometric analysis and predefined architectural rules to identify and reconstruct façade elements such as walls, windows and doors. The method uses a combination of data-driven and model-driven strategies. Firstly, flat surfaces are detected from the point cloud using a conditional RANSAC approach to identify façade planes. The point cloud is then segmented based on geometric properties, such as point density and spatial distribution, to distinguish façade components from other structures. Histogram-based analysis is used to identify repetitive façade patterns.

The key idea in the study is the use of façade grammar, which is a set of predefined rules describing how building façades are structured. The grammar is used to guide the reconstruction process and helps the method fill in parts of the façade that are missing or hidden in the point cloud data. This results in reconstructed façades that are more complete and better organized, even when the input data is incomplete (Sadeghi et al. 2015).

Zhao et al. (2025) propose a comprehensive workflow for automatically reconstructing 3D building models from airborne LiDAR (ALS) point clouds by placing special emphasis on façade geometry and semantic detail rather than relying primarily on rooftop information. The

method introduced three major components: accurate ground-attached footprints, robust detection of façade openings (windows and doors) and integration of these features into a 3D model with structural and semantic accuracy.

The methodology begins by separating the roof points from the façade points in the point cloud. This is done by using a plane segmentation and classification process based on RANSAC, a model-driven algorithm that iteratively fits planes to subsets of the point cloud. The planes are extracted from the point cloud by setting a distance threshold that decides if the point is an inlier in a plane or not. This results in a number of dominant planes that represent the surfaces of the building. The planes with normals close to vertical are classified as façades and others as roofs. This step ensures that vertical surfaces can be processed separately from horizontal or sloped roof surfaces, avoiding distortions that are common in roof-based footprint generation (Zhao et al. 2025).

Instead of projecting roof edges vertically to estimate building outlines, the method uses façade point clouds for more stable footprint extraction. The extracted façade points are projected onto the ground plane and an Alpha Shape algorithm with binary search strategy is used to compute a closed contour that represents the 2D building footprint. To reduce redundancy and simplify the point sets, a grid was used. After extracting the initial contour, the footprint is simplified with the Douglas-Peucker algorithm (Douglas & Peucker, 1973) to ensure straight lines and angles (Zhao et al. 2025)

2.6.2 Machine Learning Methods

A known challenge in generating 3D city models from point clouds is ensuring that building height and shape estimates are accurate and not distorted by errors in the dataset or irrelevant objects. Park and Guldmann (2019) address this issue by applying a machine learning-based point cloud classification approach that improves the extraction of building heights by combining building footprint and LiDAR data. Their method was to use a Random Forest classifier to categorize LiDAR points into four classes: rooftop, wall, ground, and high outliers. The points located on the rooftop were then identified and used to estimate building heights. This method resulted in significantly more accurate height estimates compared to using unclassified point clouds. The overall classification accuracy reported in the study was 96.5 % and the method produced reliable 3D building models.

Deep learning methods have been increasingly applied to 3D building reconstruction using Earth Observation (EO) data, such as aerial imagery, satellite imagery and point clouds. According to Buyukdemircioglu (2022), convolutional neural networks (CNNs) have shown strong potential for automatically extracting building features, including building outlines, roof structures and height information, which are essential for generating 3D building models. Unlike traditional reconstruction, deep learning-based methods are able to learn complex spatial patterns directly from the data. This allows for more robust reconstruction of buildings with varying shapes and levels of complexity.

A study by Zeng et al. (2018) introduces an innovative approach called Neural Procedural Reconstruction (NPR) which includes training deep neural networks (DNNs) to create high quality 3D building models directly from point cloud data. The method represents buildings using rules and parameters that describe how the buildings are constructed, rather than only as raw geometry such as points or triangles. Instead of only analyzing small geometric details in the point cloud, the method uses deep learning to understand the overall structure of a building and reconstruct it as a clean, CAD-like 3D model. This makes it possible to reconstruct buildings even when the input point clouds are incomplete or sparse. The approach is demonstrated using LiDAR data of residential buildings and produces compact models with clearly defined building parts. The results show that NPR performs better than previous reconstruction methods in both visual quality and accuracy.

High quality and well-annotated datasets are essential for training deep learning models to reconstruct accurate 3D building models from point clouds. One notable contribution in this area is the RoofN3D dataset presented by Wichmann, Agoub and Kada (2018). Many deep learning methods have demonstrated the potential for building reconstruction, but their application to point clouds has been limited by the lack of large, labeled training data, particularly for complex building structures. RoofN3D addresses this gap by providing a point cloud dataset derived from aerial LiDAR that includes both geometric and semantic information, such as roof, wall and ground points. This enables deep learning models to learn to identify building components and generate accurate 3D models.

Another recent contribution relevant to 3D city modeling is the study by Yarroudh et al. (2024), which proposes an automated method for upgrading LOD 2.2 3D building models to LOD3 using point cloud data and advanced machine learning techniques. In order to do estimations and analyses of this kind, detailed façade features such as windows and doors are essential but challenging to extract. Their methodology begins with extracting façade surfaces from the LOD 2.2 building models and converting the 3D point cloud data into 2D image format. The images are then processed with Grounding DINO, a deep learning model designed for object detection and segmentation. The model detects and segments façade elements, which are then reintegrated into the 3D structure to produce an enhanced LOD3 representation. This automated pipeline offers a practical solution for generating more detailed 3D building models from point cloud data, which is particularly valuable for large scale urban modeling and digital twin applications.

Windows and doors can be challenging to extract from ALS data because the point clouds lack color and texture cues. In a method used by Zhao et al. (2025), the openings appear as local holes or sparse regions in the vertical surfaces. To detect these features, the façade points are projected into 2D images. A deep learning object detector named YOLO is trained on the manually annotated façade images to locate windows and doors using bounding-box detection. The detected bounding boxes are then back-projected into 3D coordinates. This results in accurate contours of façade openings.

After defining footprints and identifying windows/doors, the method uses a model-driven volumetric reconstruction strategy based on existing tools like City3D and geometric processing libraries. Candidate walls and roof patches are generated from the segmented footprints and roof points. These surfaces are then combined to form a closed 3D building geometry, meaning that the building is represented as a solid and complete object. During this process, the method selects the most suitable building shape by considering how well the model fits the point cloud data, while preserving geometry simple and accurate. After building the geometric framework, windows and doors are extruded into solids and embedded into the wall meshes using Boolean operations, ensuring a fully integrated model with both geometric and semantic detail. The authors evaluate their method on real urban buildings and report high geometric accuracy with the average RMSE ≈ 0.42 m. The façade detection performance was high, with precision, recall and F1-score all above 97% (Zhao et al. 2025).

2.7 Roof Type Classification

2.7.1 Geometry-based Methods

Lingfors et al. (2017) presents a method for classifying roof types using LiDAR data by combining geometric processing with a model-based approach. First, building footprints were used to isolate individual buildings from the LiDAR data. The building shapes were then simplified into rectangles using a largest inscribed rectangle (LIR). The LiDAR points within each building were preprocessed. This included removing outliers such as trees and noise, adjusting the position of the roof ridge if necessary and normalizing the height of the points relative to the ground level.

After preprocessing, buildings were grouped into similarity classes based on their size and shape. This process, referred to as co-classing, allows multiple buildings with similar geometry to be analyzed together which improves the reliability of the classification. The actual roof classification is then performed by comparing the LiDAR-derived geometry to a predefined library of roof templates such as flat, gabled, hipped and pyramid roofs. For each possible roof type, regression analysis is used to fit planar surfaces to the point cloud, and the roof type yielding the lowest fitting error is selected as the final classification (Lingfors et al., 2017).

Alexander et al. (2009) propose a geometry-based approach for classifying roof types by combining LiDAR elevation data with building footprint polygons. In their method, LiDAR points within each building footprint are used to generate a triangulated surface representation, from which geometric properties such as slope and aspect are derived. These parameters are then used to distinguish between different roof types, primarily separating flat roofs from pitched roofs based on slope thresholds. The method further segments roof surfaces by analyzing elevation differences and aspect direction, allowing multiple roof planes to be identified within individual buildings.

The authors also evaluate the effect of LiDAR point density on classification accuracy and show that higher point densities improve the detection of roof structures. Their results indicate

classification accuracies of 73%, 77% and 86% for point densities of 1, 16 and 40 points/m^2, respectively (Alexander et al. (2009). This study demonstrates that roof classification can be performed using geometric features extracted directly from LiDAR data. However, the method is mainly designed for simple roof structures. In addition, features such as chimneys and roof extensions can cause errors in the triangulated surface representation (Alexander et al. (2009).

Awrangheb et al. (2013) propose a data-driven method for automatic roof extraction by integrating LiDAR and aerial imagery. The approach is based on detecting planar roof segments within LiDAR point clouds and refining them using image-derived features. First, candidate roof planes are identified by searching for regions in the LiDAR data where points belong to the same planar surface. These regions are defined using image lines extracted from aerial imagery, which guide the direction of roof edges and ridgelines. The regions are then iteratively expanded using neighboring LiDAR points to form complete roof planes.

To improve classification reliability, the method applies a rule-based procedure to remove false planes caused by vegetation. Non-ground LiDAR points near detected roof edges or ridgelines are used to estimate planar roof surfaces, and additional filtering is performed to eliminate planes generated on trees. This allows the algorithm to separate building roofs from surrounding objects and produce cleaner roof segmentation results (Awrangheb et al. 2013).

The approach is fully automatic and combines both geometric information from LiDAR and structural cues from imagery. After plane detection, the extracted planar segments represent different roof surfaces, which can be used to describe roof structure and geometry (Awrangheb et al. 2013).

2.7.2 Machine Learning Methods

Several studies have been done for roof type classification by combining multiple geospatial data sources with machine learning techniques. Castagno and Atkins (2018) propose a method that combines airborne LiDAR data and high-resolution satellite imagery to automatically classify building roof shapes. In their approach, building footprints are first used to distinguish individual buildings from LiDAR and satellite data. The LiDAR point clouds are then processed and converted into depth image representations of the roofs, projecting 3D point clouds onto a 2D grid. To improve the data quality, several preprocessing steps were applied, such as removal of ground points, filtering of wall points using surface normals, and outlier detection. In parallel, satellite imagery was cropped around each building footprint to generate corresponding RGB patches. The processed datasets were then manually labeled into predefined roof categories, creating a supervised training dataset. The classification framework consists of a two-stage machine learning pipeline. In the first stage, convolutional neural networks (CNNs) such as ResNet and Inception were used to extract high-level features from both LiDAR-derived images and RGB images. In the second stage, the extracted features were used as input to classical machine learning classifiers, including support vector machine and random forest models, to perform final roof type classification. The results show that

combining features from LiDAR and satellite images can lead to higher accuracy than using either data independently (Castagne & Atkins, 2018).

Meng et al. (2023) propose a data driven method for roof type classification based on deep learning and image representations of buildings. The authors construct a dataset of labeled roof images and train convolutional neural networks (CNNs) to automatically distinguish between different roof types. The workflow begins with collecting aerial imagery and extracting image patches corresponding to individual buildings. Each patch is manually labeled into predefined roof categories, creating a supervised dataset used for training and evaluation. The study defines five primary roof classes to represent common residential buildings: simple gable, simple cross-gable, complex cross-gable, simple hip and cross-hip. These classes differentiate between variations in roof complexity, such as the number of ridgelines and building footprint shape. An additional class labeled “unknown” is introduced to prevent incorrect classifications caused by low-quality images (Meng et al., 2023). To perform the classification, the study employs a CNN-based architecture that learns visual features directly from the roof images, The network is trained to recognize patterns such as a shape, edges and texture that differentiate roof types. During training, the dataset is split into training and testing subsets to evaluate generalization performance. Data augmentation techniques, including rotation and flipping of images, are applied to increase diversity and robustness. The extracted features are then used by the classifier to assign each building image to a specific roof category (Meng et al., 2023).

Part III

Methodology

3. Method

The study presents an approach for roof classification from point cloud data using a machine learning model trained on synthetically generated data. This chapter is structured into four parts, beginning with an overview of the method. This is followed by a description of the datasets used throughout the workflow. The next part describes the study area and the motivation for its selection. Finally, the practical implementation of the method is presented, including data preprocessing, generation of synthetic training data, preparation of testing data from real LiDAR point clouds, development and evaluation of the machine learning model, and the reconstruction of 3D building models based on the predicted roof types.

3.1 An Overview of the Method

The method used to achieve the aim of the project is based on training a machine learning model using synthetically generated data and subsequently applying the trained model to real LiDAR data. First, parametric 3D roof models are created to represent different roof types. These models are then converted into synthetic point clouds, which simulate LiDAR data. The synthetic point clouds are then rasterized into image representations that are used as input to the machine learning model during training.

To classify real buildings, similar preprocessing steps are applied to real LiDAR data. Point clouds corresponding to individual building footprints are extracted, normalized, and converted into raster images representing the roof geometry. These rasterized roof images are then used as input to the trained model, allowing roof types of real buildings to be predicted. The predicted roof classes are subsequently used together with extracted height information and reapplied to the same parametric rule package used during the synthetic data generation process in order to reconstruct simplified 3D building models.

This workflow ensures consistency between the synthetic training data and the real-world input data, enabling the model to generalize from simulated roof structures to real LiDAR-derived roofs. An overview of the method is provided in Figure 3.1.

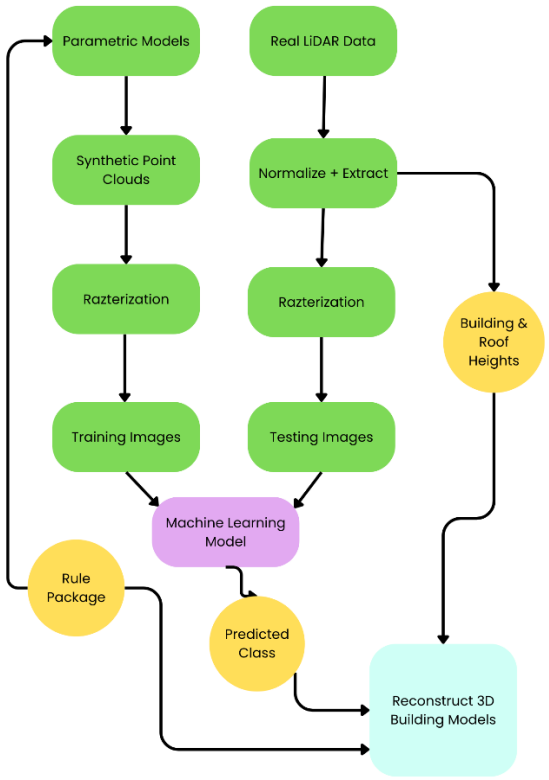


Figure 3.1: Overview of the method

3.2 Analyzing Data

The data provided by Göteborgs Stad consists of airborne LiDAR data covering part of Gothenburg's city center, along with corresponding building footprints, a digital elevation model (DEM) and roof geometries in LOD3. In addition, building footprints from OpenStreetMap were used to generate a larger and more diverse set of synthetic training data. Table 3.1 provides an overview of the datasets used in this study.

Table 3.1: Overview of the datasets

Data	**Retrieval**	**Format**	**Description**
Building footprints	year 2026	JSON Source File	Digital footprints of buildings in the area
LiDAR data	year 2017	LAS	Airborne LiDAR data of the area
DEM data	year 2022	TIF	Digital elevation model representing ground surface elevation
3D roof models	year 2014	JSON Source File	Geometric roofs of buildings in the area in LOD3.
OSM data	year 2026	OSM	Building footprints from larger cities in Sweden

The JSON files contain buildings within the designated study area. The building footprints dataset represents the overall geometry of buildings in 2D, while the 3D roof models provide a more detailed representation of roof structures in 3D. Figure 3.2 illustrates a comparison between the two datasets.

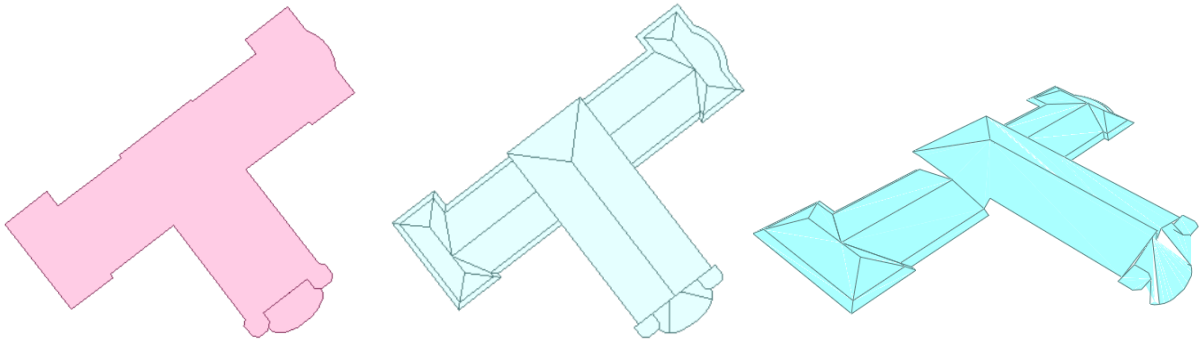


Figure 3.2: Building footprint in 2D (left), roof structures in 2D (middle) and roof structure in 3D (Göteborgs Stad)

The building footprints from OpenStreetMap and Göteborgs Stad represent registered buildings. The OpenStreetMap footprints are used to generate synthetic training data, while the building footprints provided by Göteborgs Stad are used for testing and evaluation. The roof classification is based on the corresponding 3D roof models. This dataset is analyzed manually to identify the most common roof types in the study area. Based on this manual analysis, the following roof types are considered relevant: *Monopitch*, *Mansard*, *Hipped*, *Gabled* & *Flat*. In addition, an "Other" class is introduced to account for undefined or uncommon roof types, these roofs were excluded from the classification task.

The point density of the LiDAR data varies across the study area, generally ranging from 10 points/m² and above. This is consistent with previously described municipal LiDAR datasets, which typically exhibit higher point densities than national data, allowing for more precise analysis and calculations.

In addition to the LiDAR data, a DEM representing the terrain elevation of the study area is used. The DEM data is utilized to calculate relative building heights by separating ground elevation from the absolute height values in the LiDAR point clouds. Figure 3.3 presents a comparison between the LiDAR point cloud data and the raster representation of the DEM.

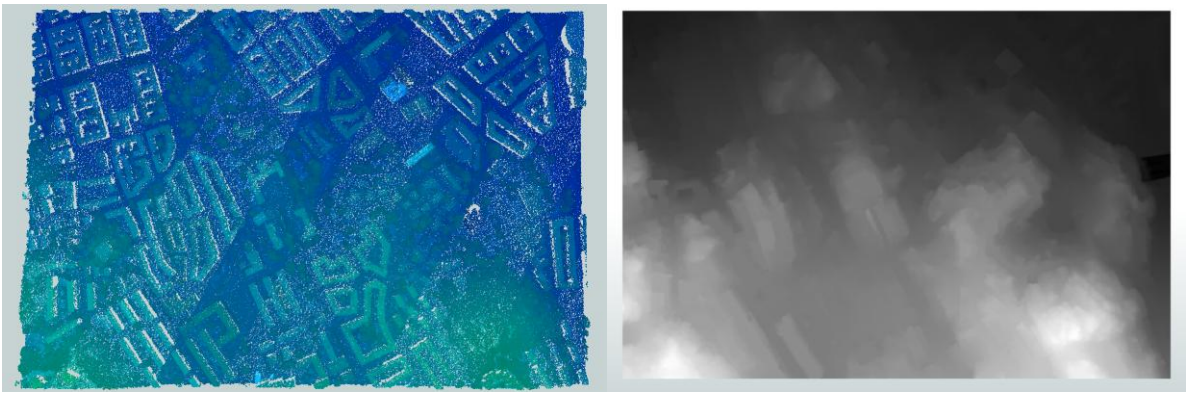


Figure 3.3: LiDAR point cloud data (left) and rasterized DEM representation (right) (Göteborgs Stad)

Since the datasets were collected at different time periods, discrepancies between them may occur. Furthermore, as the data is a couple of years old, differences compared to the current state of the area are possible. However, these discrepancies are not critical for this study, as the objective is classification rather than an accurate representation of the current conditions of the area.

3.3 Study Area

The selected study area of 138 hectare is located in the central parts of the city, with focus on the districts of Lorensberg and Landala, see Figure 3.4.

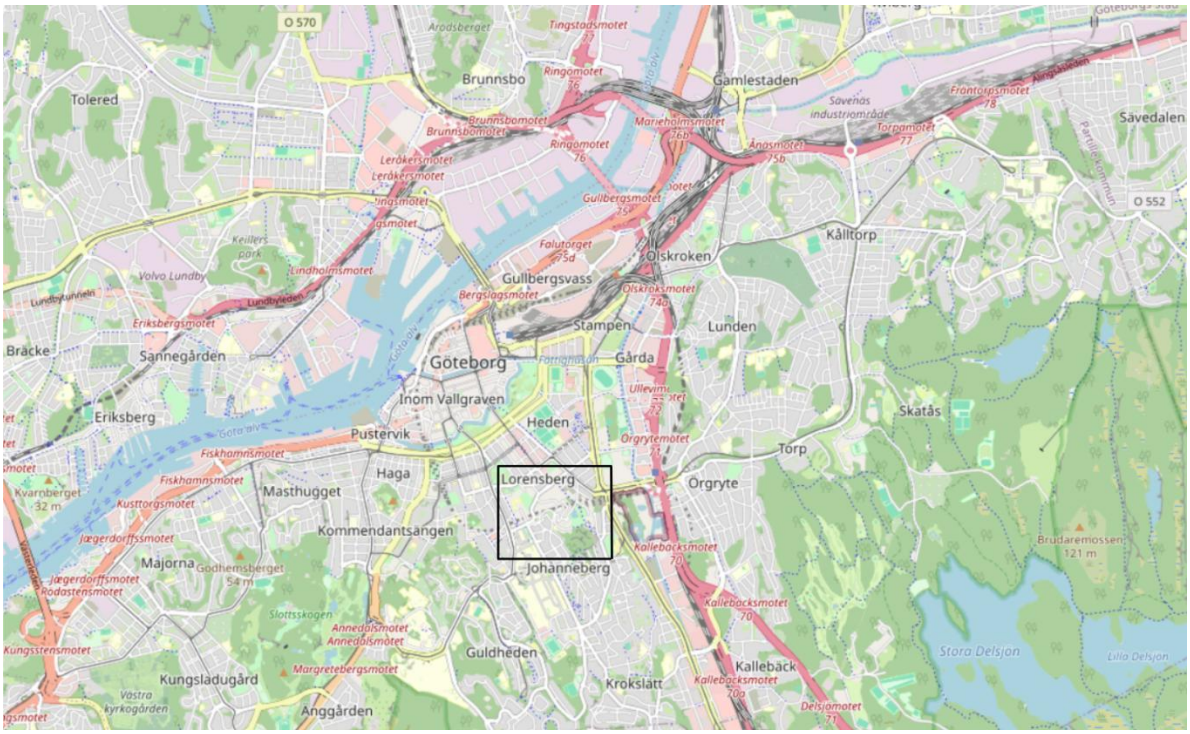


Figure 3.4: Overview of Gothenburg with study area marked (OpenStreetMap contributors).

This area represents a typical urban environment, characterized by a mix of architectural styles and building types. The built environment includes both residential and commercial land uses, with buildings varying in size, height, and structure. The area consists of everything from larger multi-storey apartment blocks to smaller, more traditional buildings, reflecting the spatial and functional diversity commonly found in inner-city settings. Figure 3.5 gives an overview of the study area.

Figure 3.5: Overview of the study area (Göteborgs Stad)

3.4 Generation of Training Data

3.4.1 Generating Rule Package from CityEngine

In CityEngine (see Chapter 2.2.1), several pre-made rules are available. One of these, named *"Building_From_Footprint.cga"*, is used as inspiration for the generated rule package "*Roof_Types.cga"* used in this project. Since the aim is to classify different roof types, the section of the CGA script containing these roof variations is used together with other relevant parts. For this study, five roof types, according to Inspire, are analyzed and selected to represent the urban structure and building characteristics of Gothenburg: *MansardRoof*, *hippedRoof*, *gabledRoof*, *monopitchRoof* and *flatRoof*. The corresponding roof geometries generated in CityEngine are analyzed and matched to these definitions. Based on this comparison, the following CityEngine roof types are chosen as representatives: *Gambrel*, *Hip*, *Gable*, *Shed* and *Flat* (Figure 3.6).

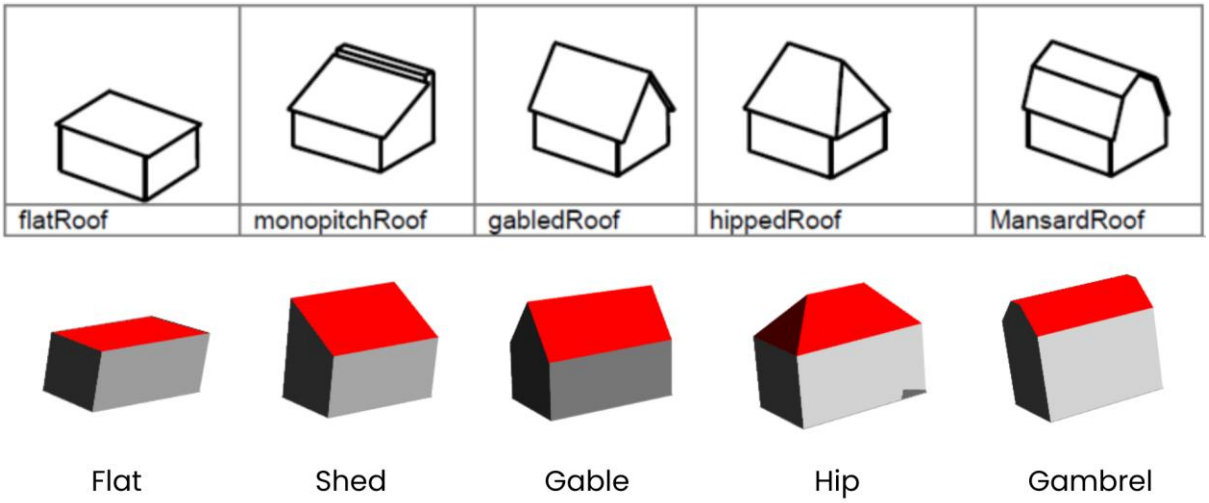


Figure 3.6: Comparison of roof types in CityEngine (below) and their corresponding Inspire classification (above). The upper part of the figure is adapted from (Lantmäteriet, 2024c, p. 69)

The CityEngine roof type names were adjusted to correspond to the Inspire classification. The CGA script from CityEngine is then exported as a rule package (RPK) which is a suitable format to be used in FME.

3.4.2 Generation of Parametric 3D Building Models in FME

The rule package "*Roof_Types.rpk*", created in CityEngine, is used in FME to generate 3D building models from building footprints by defining parameters such as height, roof shape, and visual properties. The rule package contains the procedural roof definitions and enables the same CGA-based building logic to be reused for generating large numbers of synthetic building variations. To create diverse training data, key attributes like roof type and dimensions are varied using randomized values. The building footprints used for generating the 3D buildings are obtained from OpenStreetMap and originate from larger Swedish cities such as Gothenburg, Stockholm, and Malmö in order to resemble the study area.

The 3D building models are generated in FME using the *CityEngineModelGenerator* transformer. To introduce variation in the training data, some of these parameters are randomized using the *RandomNumberGenerator*. The parameters that are randomized include roof type, building height and roof height. The JSON script is created using the *AttributeCreator*, where the JSON syntax from the *CityEngineModelGenerator* is combined with the generated random values. These steps are illustrated in Figure 3.7, together with additional transformers used for supporting tasks, including coordinate reprojection (*Reprojector)*, area extraction (*GeometryFilter*) and mapping randomized roof type values to descriptive names (*AttributeValueMapper*).

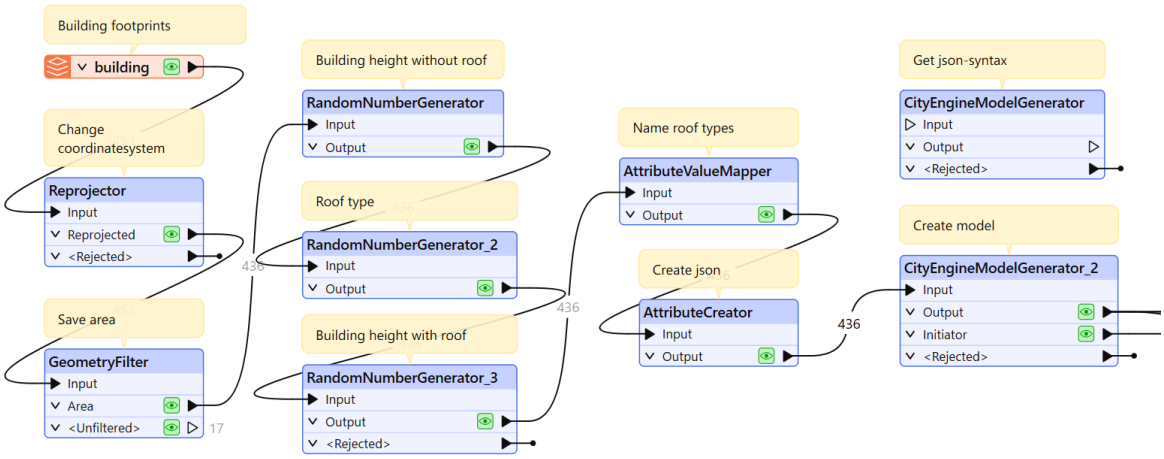


Figure 3.7: Workflow for generating 3D building from a rule package

The *CityEngineModelGenerator* transformer receives building footprints, a rule package and a JSON script as input, and outputs 3D buildings extruded from the footprints with varying roof types and heights (see Figure 3.8).

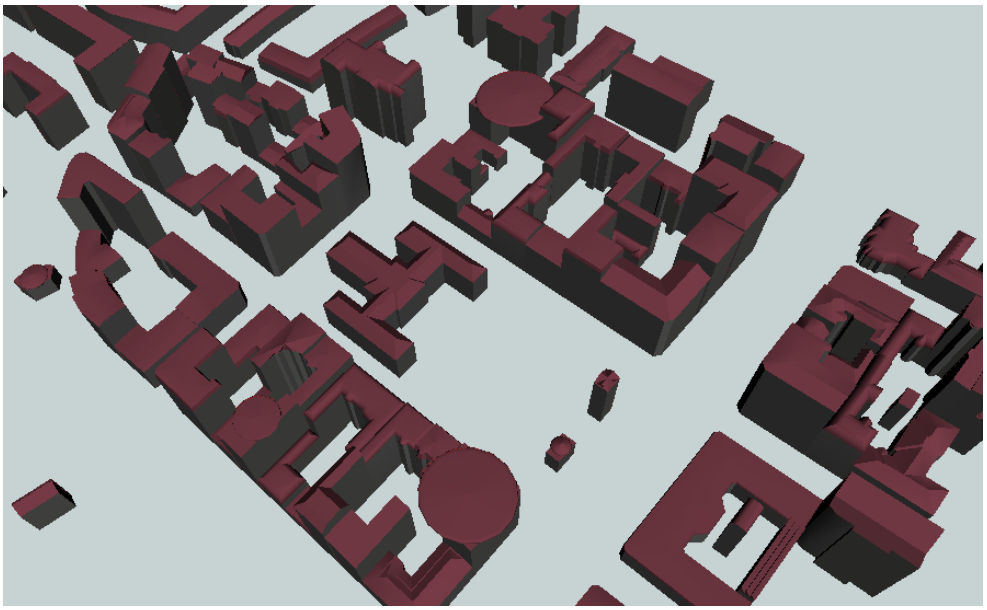


Figure 3.8: 3D buildings generated from rule package in FME.

3.4.3 Generation of Synthetic Point Clouds

The next step is to convert the 3D building models into point clouds. This is performed using the *PointCloudCombiner* transformer in FME. The transformer takes the 3D building geometries as input and generates synthetic point clouds by sampling points across the surfaces of the 3D objects. The output consists of separate point cloud representations of each building. A key parameter in this process is the point interval, which controls the spacing between generated points. A smaller interval results in a denser point cloud, while a larger interval produces fewer points and a more generalized representation. In this study, the point interval is set to 0.25, which is about 16 points/m^2, to approximate the density of the original LiDAR data as closely as possible. To further emulate real point cloud data, the synthetic point clouds

are “spoiled” by introducing random distortions in x, y and z directions within a range of ±0.1 m. The *PointCloudExpressionEvaluator* is used for this purpose, ensuring that the generated point clouds resemble real measurements. The resulting point clouds (see Figure 3.9) are then used in subsequent processing steps for roof extraction and classification.

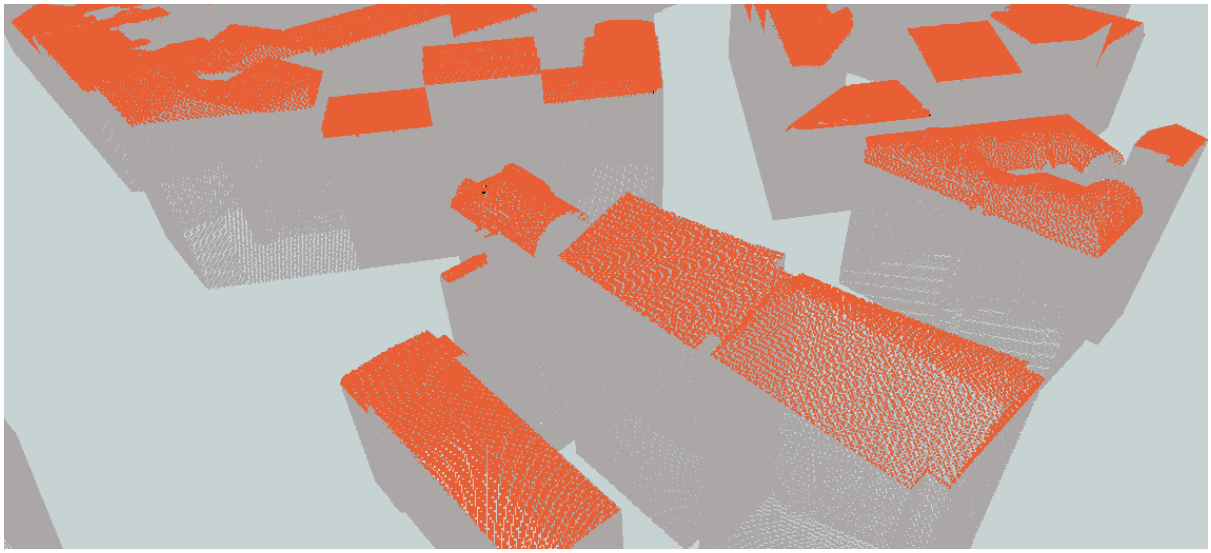


Figure 3.9: Synthetically generated point cloud data.

3.4.4 Point Cloud Preprocessing and Normalization

Some adjustments are made to the point clouds to facilitate their transformation into rasterized images. To generate a consistent image representation for each building, the roofs are visualized from a top-down perspective, corresponding to an orthogonal projection along the z-axis. First, each building point cloud is translated so that it is centered in the coordinate system. The buildings are then rotated such that the longest side of the footprint is aligned with the x-axis. This normalization step ensures that roofs with similar shapes have a consistent orientation in the resulting images, improving the robustness of the subsequent classification. The adjustments to the orientation of each building's point cloud, along with the applied transformers, are illustrated in Figure 3.10.

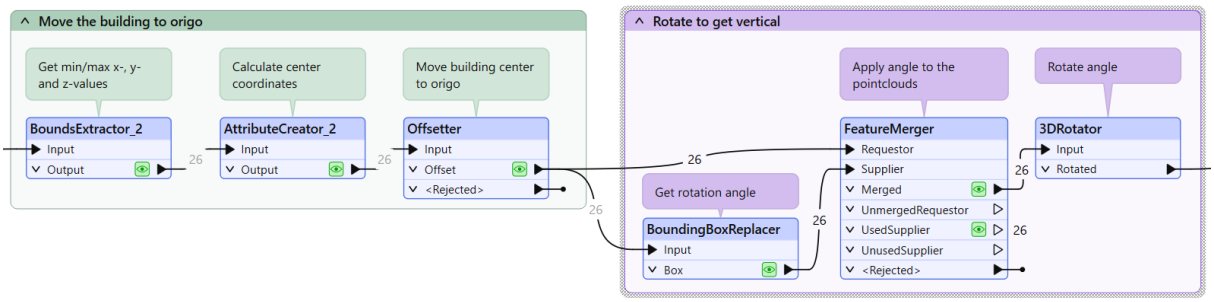


Figure 3.10: Orientation workflow

To further improve the rasterization process, points belonging to building walls are removed. These points can otherwise introduce noise and distort the roof representation when converted

to raster format. Therefore, each building point cloud is vertically clipped just below the roof surface. This is performed using the *PointCloudFilter* transformer and ensures that the rasterized images primarily contain roof geometry, resulting in cleaner and more consistent input data for the classification step (see Figure 3.11).

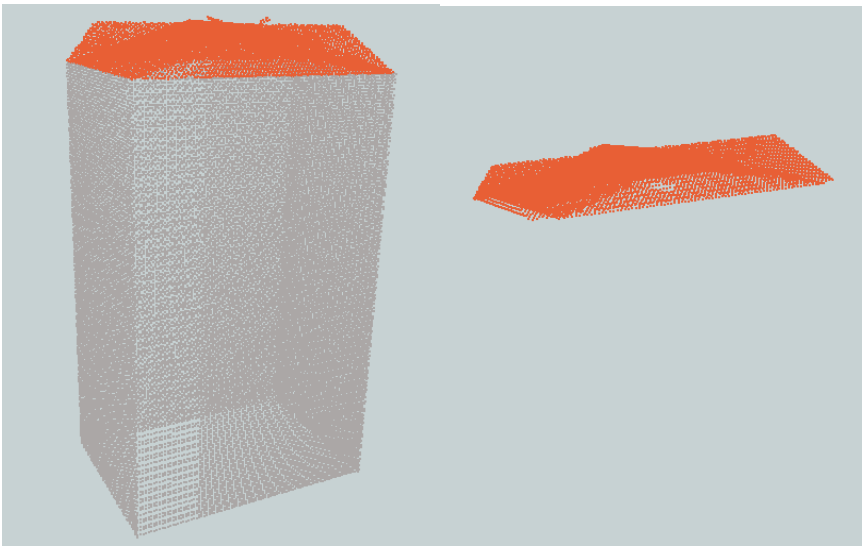


Figure 3.11: Point cloud of the full building (left) and roof-only point cloud (right).

3.4.5 Rasterization of Point Clouds

The roof points are then converted into raster images. In this step, the 3D point cloud data is projected into a two-dimensional grid, where each cell represents a pixel in the resulting raster image. The height values (z-values) of the points are used to assign numeric values to the raster cells, creating an image that preserves the geometric structure of the roof surface (see Figure 3.12).

The rasterization is done using the *NumericRasterizer* transformer in FME. The resolution of the raster is defined by the chosen cell size, which determines the level of detail captured in the image. Smaller cell sizes result in higher-resolution images that better represent roof features such as ridgelines and slopes. By rasterizing the point clouds in this way, the 3D roof geometry is transformed into a structured image format suitable for further analysis and machine learning-based classification.

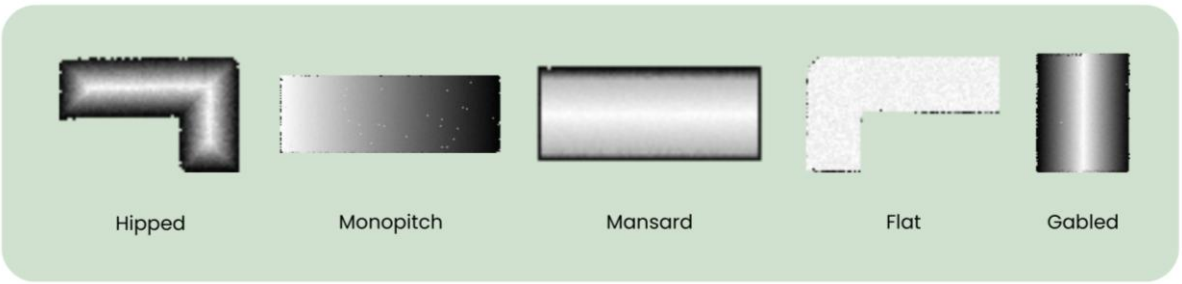


Figure 3.12: Rasterized images derived from point cloud data.

3.4.6 Generation of Roof Objects

After examining real roofs, it was observed that many buildings contain objects such as chimneys, ventilation units and antennas. These structures can affect the roof shape and may interfere with the classification. To better represent real-world conditions, similar objects are added to the synthetic training data. During this study, two methods were explored to conduct this.

3.4.6.1 Method A - Creating 3D Roof Objects

The aim of this step is to generate roof objects across the entire study area and subsequently remove those located outside building footprints. This method is performed after the 3D building models are created, but before they are converted into point clouds.

First, a rectangular extent covering all buildings is created using the *BoundingBoxAccumulator*. The minimum and maximum *x* and *y* coordinates of this rectangle are then extracted using the *BoundsExtractor*. Next, the number of objects to be created is defined. Each object is assigned a unique identifier through a combination of *RandomNumberGenerator* and *Cloner*. The spatial placement of the objects is determined by generating random *x* and *y* coordinates within the bounding box. Additionally, the generated objects are divided into two types, chimneys (represented as boxes) and pipes (represented as cylinders). These steps are implemented through separate instances of the *RandomNumberGenerator* for each parameter. Figure 3.13 illustrates the overall workflow for this process.

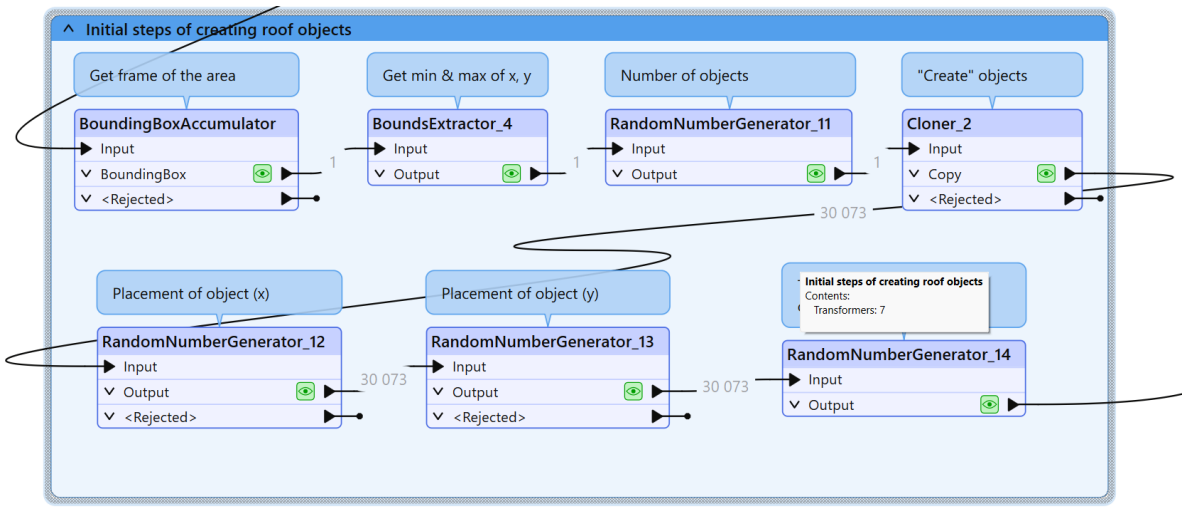


Figure 3.13: The first steps of creating roof objects

The two object types are processed through separate workflows. The *Tester* transformer divides the features into pipes and chimneys according to the previously assigned classification value. For pipe objects, the geometry is defined by a radius, whereas chimney objects are defined by width and length parameters. These parameters are generated through randomization using the *RandomNumberGenerator*. Pipe geometries are created using the *2DEllipseReplacer*, where the radius is combined with the previously generated *x* and *y* coordinates to produce circular features at varying locations. Correspondingly, chimney geometries are generated using the

2DBoxReplacer, where width and length parameters are applied to create rectangular features. Figure 3.14 illustrates the separate processing paths for the two object types.

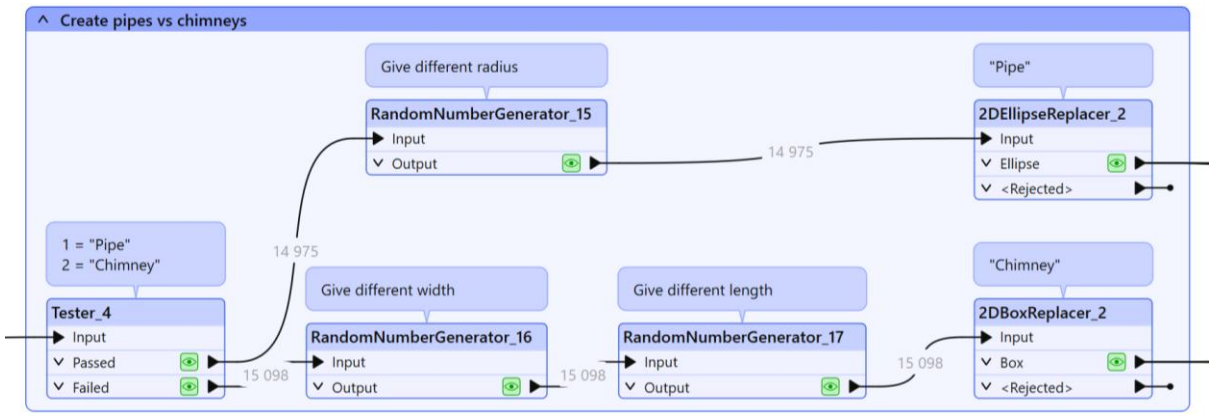


Figure 3.14 : The different paths for creating pipes vs chimneys

The generated geometries are not guaranteed to be spatially separated and may therefore intersect or overlap. To address this, the *Intersector* and *AreaOnAreaOverlayer* transformers are used to identify intersections and overlaps. The *Tester* transformer extracts features representing overlaps and intersections based on the conditions segments > 1 or overlaps > 1. Duplicate geometries are subsequently removed using the *DuplicateFilter*, and the *FeatureJoiner* is applied to eliminate unwanted features. The *SpatialFilter* uses building footprints as a filter to remove geometries that are not located on buildings, i.e. those positioned outside the building extents. Finally, the vertical positioning of the objects is adjusted using the *Offsetter*, which raises the base of each object, and the *Extruder* is used to generate three-dimensional geometries and define their height. Figure 3.15 illustrates the final steps involved in generating and filtering the roof objects, while Figure 3.16 and Figure 3.17 shows the resulting objects positioned on top of the buildings as complete building models and rasterized images.

Figure 3.15: The final steps of removing overlaps and objects outside buildings

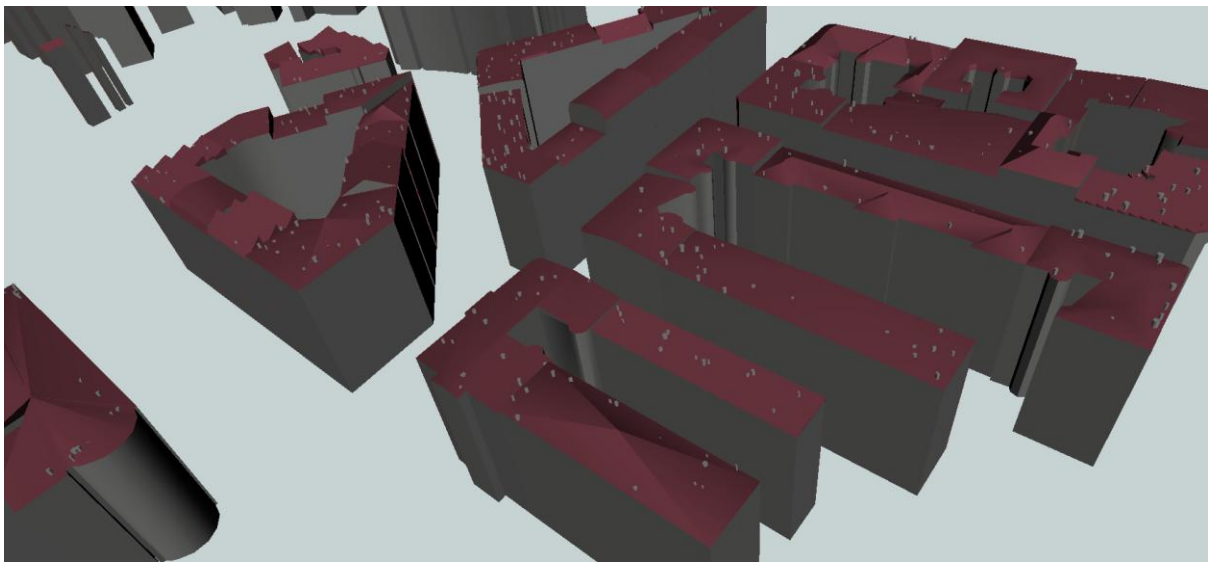


Figure 3.16: The generated buildings with roof objects

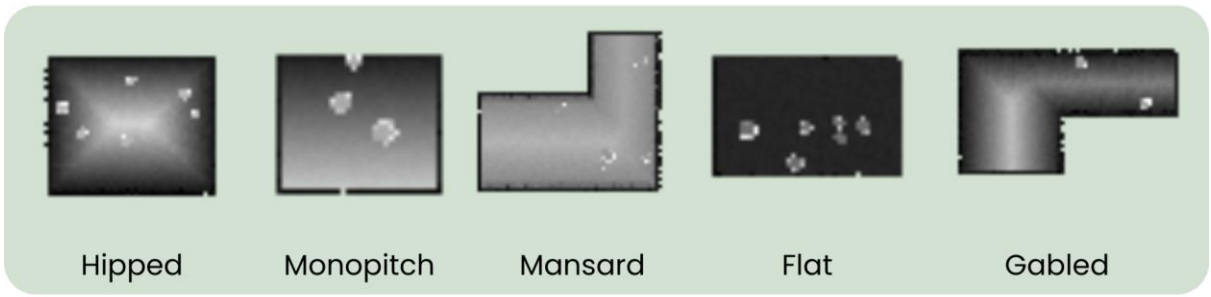


Figure 3.17: The generated rasterized images with roof objects

3.4.6.2 Method B - Modification of Point Clouds to Generate Roof Objects

This method is done after the 3D building models have been converted to point clouds and before rasterization. By increasing the *z*-value in random areas of the point cloud within the building footprints, roof-objects can be synthetically created.

First, the building footprints are processed using an *AreaCalculator* to compute their area (see Figure 3.18). Areas smaller than 30 m^2 are excluded, since they are not suitable for adding additional objects. The *BoundsExtractor* is then used to determine the spatial extent of each footprint, within which random points are generated to represent potential object locations. These points are buffered with a randomly generated size to create small polygon areas representing rooftop objects.

Because some buildings have irregular shapes, such as L-shaped footprints, some randomly generated polygons may fall outside the actual building area. A *SpatialFilter* is therefore applied to ensure that only polygons located within the building footprints are kept. Next, a *Clipper* is used to extract the portions of the synthetic point clouds that overlap with the buffered object areas. The extracted points are then modified by increasing their *z*-values using

a *PointCloudExpressionEvaluator*, effectively raising them above the roof surface to simulate rooftop structures.

Finally, these modified point cloud segments are merged back with the remaining roof points using the *PointCloudCombiner*. This results in synthetic point clouds that include small, elevated structures, making the training data more representative of real roofs.

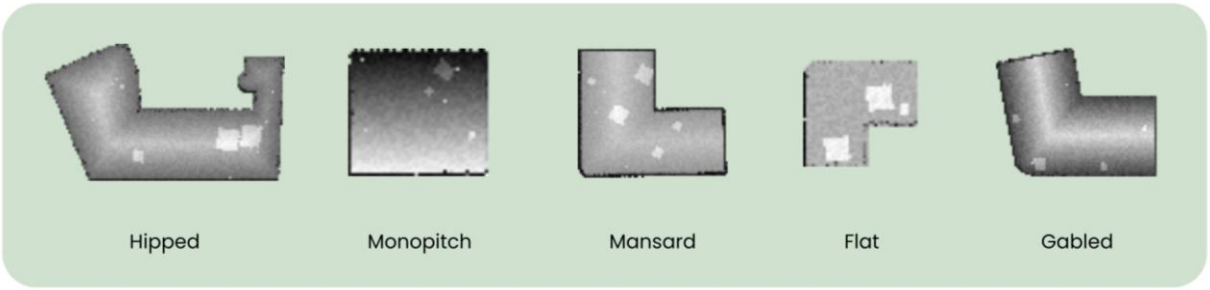


Figure 3.18: The process of adding artifacts to the point clouds.

3.4.7 Chosen method of generating roof objects

Both methods work as expected. However, to continue the study, one method has to be selected. Method A is chosen because it most closely resembles the actual LiDAR derived rasterized images (see Section 3.5.5). In addition, Method A generates a greater variation of roof shapes, including both circular and square-like forms. Figure 3.19 illustrates the differences between the two methods and compares them with the real rasterized roofs.

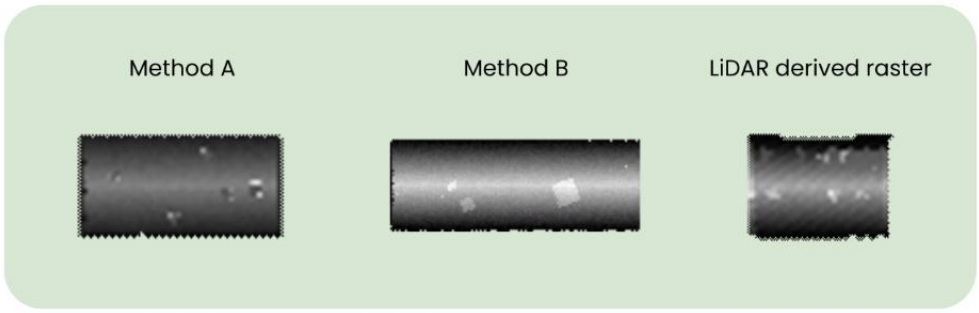


Figure 3.19: Comparison of rasterized images generated using Method A, Method B, and the real LiDAR derived image.

3.5 Generation of Testing Data

3.5.1 Preprocessing of Test Data

The building footprints consist only of building IDs and object 2D geometries. For this study, an additional attribute describing the roof type of each building is required for classification in later steps. The 3D roof models dataset is therefore analyzed to determine the roof type for each building footprint in order to manually label the testing data. This is achieved by

examining orthophotos of the study area together with the building footprints and 3D roof models.

To identify which roof parts belong to each building, FME is used. The *Clipper* tool is applied to separate the roof geometries and associate them with the correct building. Each building ID is then selected using the *Tester* tool to extract only the relevant roof components for that specific building footprint. Figure 3.20 illustrates the overall workflow.

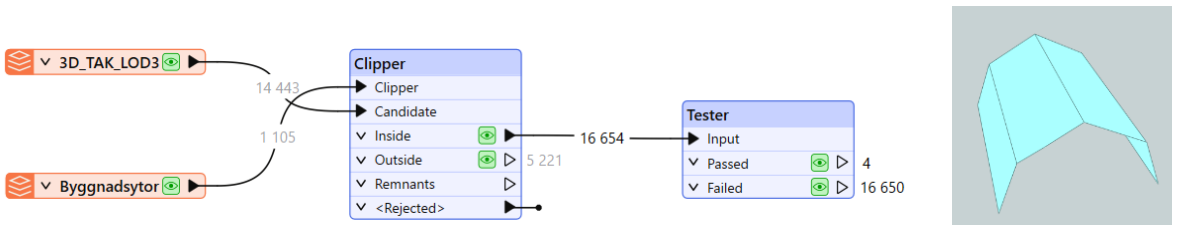


Figure 3.20: Overall workflow for classifying the roof types

To add an additional attribute to each building ID, QGIS is used. A new attribute, "Roof_type," is created, and the buildings are then analyzed using both QGIS and FME to assign a roof type to each building. The classification is designed to match the roof type naming scheme used in Inspire (see Figure 3.6) as closely as possible. Figure 3.21 presents examples of the identified roof types along with their corresponding classifications.

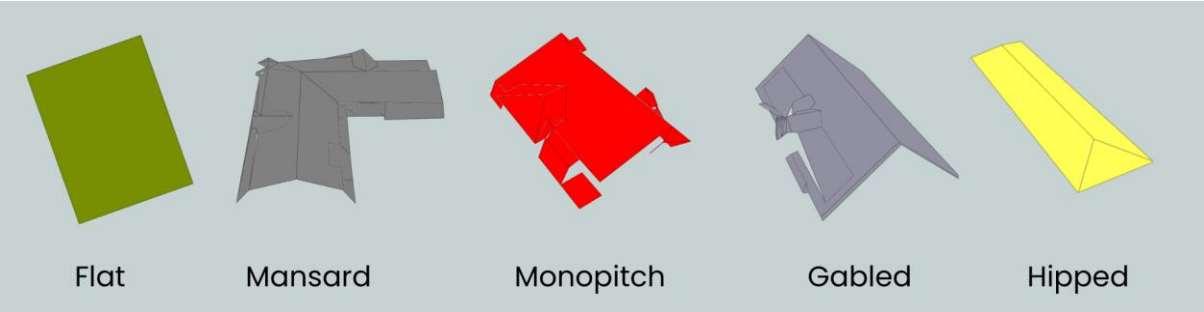


Figure 3.21: Examples from the 3D roof models with the applied classification based on the 3D roof models (Göteborgs Stad)

Once a sufficient number of building footprints have been assigned roof types to ensure a varied test dataset, the updated Building footprints file is exported and used in place of the original dataset.

Since the classification process is largely manual and requires visual interpretation of the roof geometries, it is both time-consuming and difficult to apply to larger datasets. As a result, only a subset of 205 buildings within the study area are classified and 155 of them are included in the test dataset. Table 3.2 presents the distribution of roof types following the manual classification.

Table 3.2: Distribution of roof types in training dataset

Roof type	Number of roofs identified
Monopitch	34
Mansard	20
Hipped	24
Gabled	67
Flat	10
Other	50

The "Other" class represents a relatively large proportion of the dataset, primarily because several roof structures are difficult to assign to a single predefined roof category. Figure 3.22 illustrates some examples of roof types included in the "Other" class.

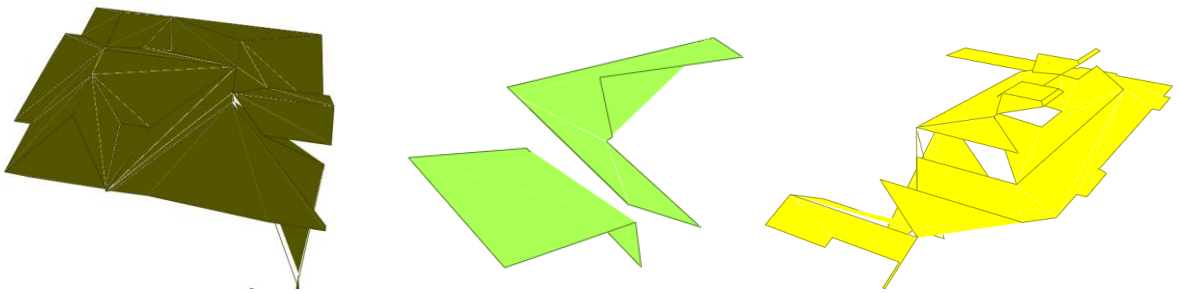


Figure 3.22: Examples of roofs in the "Other" class.

3.5.2 Adjustment of Preprocessed Data in FME

In QGIS, preprocessing of the building footprints was carried out to assign an attribute representing the roof type. However, not all buildings were classified due to how time consuming the process was. Additional preparation steps were required to isolate the relevant features before further processing.

First, features with missing or irrelevant classifications were removed using the *Tester* transformer in FME to ensure that only meaningful and usable data is included in the analysis.

Furthermore, to evaluate how different classification scenarios affect the results, additional filtering was performed. Selected roof types were selectively excluded to create variations in the number and composition of classes. These configurations were also implemented using the *Tester*, allowing flexible control over the input data for experimentation.

3.5.3 Calculation of Relative Heights Using DEM Data

The received LiDAR point clouds are stored in absolute height values. Since roof and building heights are used as parameters in the CGA script, the point clouds are normalized by removing the terrain elevation using DEM data.

First the attributes of the two datasets are combined so that each point in the LiDAR point cloud receives a corresponding terrain height value from the DEM. This is done using the *PointCloudOnRasterComponentSetter* transformer. Next the difference between the LiDAR *z*-values and the DEM elevation values is calculated using the *PointCloudExpressionEvaluator*. The resulting values represent the relative building heights, making it possible to derive the actual building and roof heights within the study area.

Finally the original *z*-values are replaced with the calculated relative *z*-value using the *PointCloudComponentCopier* transformer. Figure 3.23 illustrates the workflow for calculating relative *z*-values.

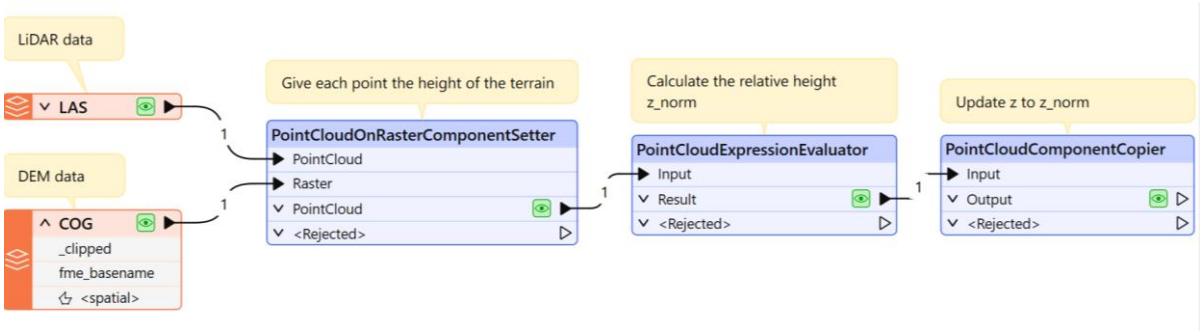


Figure 3.23: Workflow for calculating relative building heights using LiDAR and DEM data.

3.5.4 Integration of LiDAR Data and Building Footprints in FME

The objective is to associate the LiDAR point cloud with each building footprint. This is done by clipping out the LiDAR points that are located within each specific building footprint and connecting the points to the building footprints ID. First, the bounding box of the LAS data is generated using the *BoundingBoxAccumulator* transformer, followed by the use of a *Clipper* to identify buildings that are not fully covered by the LiDAR data. The *FeatureJoiner* is then applied to the Building footprints file to retain only buildings that are completely covered by the LAS. To remove buildings smaller than 50 m^2 and other irrelevant features, the *AreaCalculator* and *Tester* transformers are used. These steps ensure that buildings partially covered by the LiDAR data, as well as irrelevant elements, are excluded from further analysis. Figure 3.24 illustrates the overall workflow in FME, while Figure 3.25 shows how LAS data and building footprints are spatially connected for an individual building.

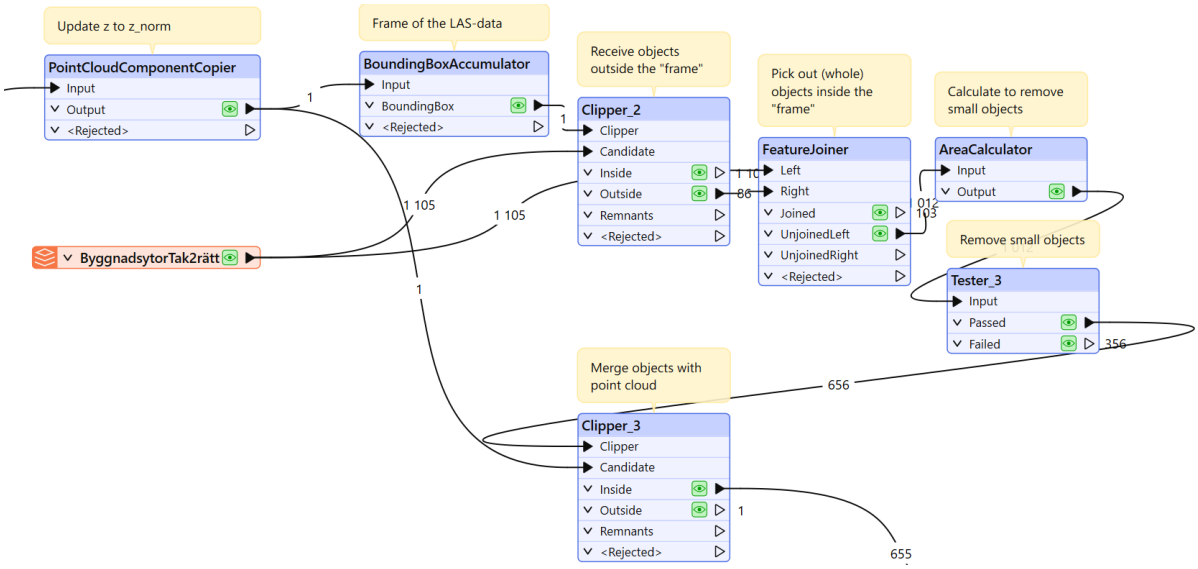


Figure 3.24: Workflow for associating building footprints with LiDAR data

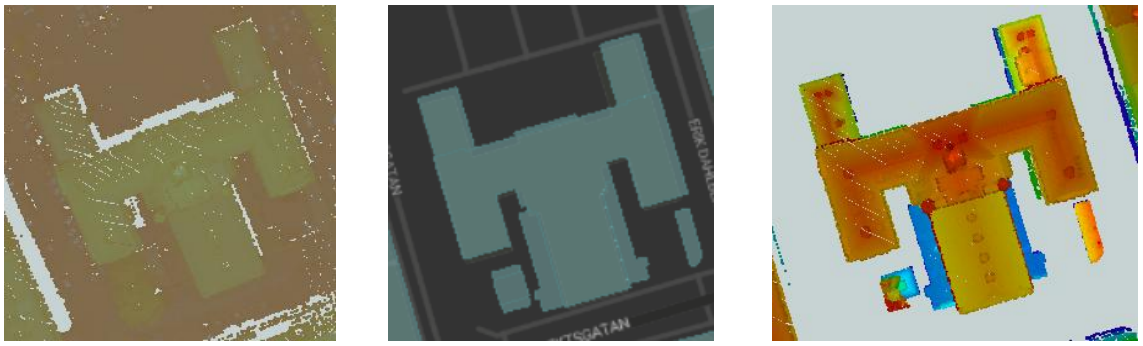


Figure 3.25: LAS data (left), building footprint (middle), and combined result (right)

3.5.5 Generating Rasterized Images in FME

The following steps of creating rasterized images are almost identical to parts in the previous section describing the creation of training data (see section 3.4.4 & 3.4.5). The LiDAR data is, for each building footprint, converted to a raster image. Some of the differences are that the points are not distorted to create noise and no artificial objects, such as chimneys, are added. The LiDAR data is unchanged, to not introduce any errors.

One difference concerns the clipping of the point cloud below the roof. For the synthetic data, the roof height is known beforehand, whereas this information is not available for the real buildings. To address this, the clipping threshold is determined individually for each building by calculating the median height of the point cloud. The point cloud is then clipped just below this value to remove most wall points while retaining the roof geometry.

The point clouds are then converted to raster images just like the training data. Figure 3.26 illustrates how the building from above looks rasterized, and Figure 3.27 illustrates examples of every rasterized roof type.

Figure 3.26: An example of a rasterized image for testing data

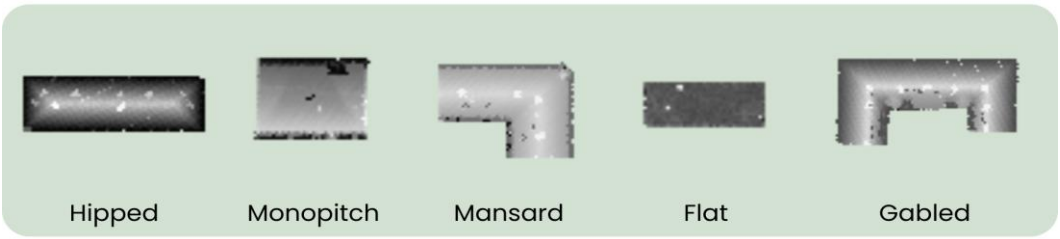


Figure 3.27: Examples of rasterized images for testing data

3.5.6 Extraction of Building Height Values

To enable reconstruction of the building models, height information derived from the processed point clouds is stored in a CSV file. After filtering the point clouds to retain only roof points, the *PointCloudToPointCoercer* transformer is used to convert the point cloud data into individual point features, making it possible to apply the *PercentileCalculator* transformer.

The *PercentileCalculator* is used to remove extreme height values from the dataset by excluding both lower and upper extreme values. Prior to this step, some manual filtering was performed to remove obvious extreme outliers, caused by e.g. vegetation. In this study, the lower percentile threshold was set to 2 and the upper percentile threshold to 95. The lower threshold was chosen because most low elevation points had already been removed during the roof filtering process. The upper threshold was adjusted experimentally to identify the most suitable value, where the 95th percentile produced the best results. This process helps reduce noise and improve the quality of the point cloud data.

Finally the *StatisticsCalculator* is used to calculate the minimum and maximum height values for each building. These values are then exported to a CSV file, where the minimum height values represent the building eave heights and the maximum values represent the roof ridge heights. Figure 3.28 illustrates the workflow for extracting building height values.

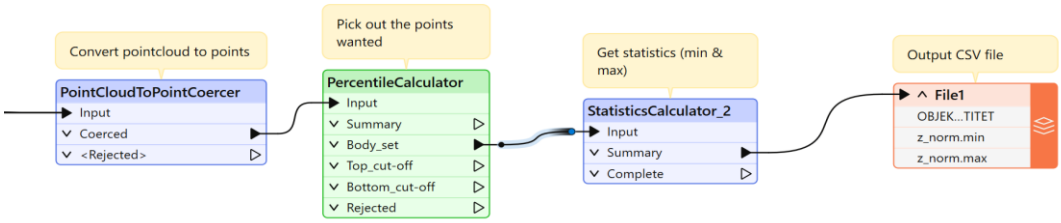


Figure 3.28: Workflow for extracting building height values from roof point clouds.

3.6 Machine Learning Model

The rasterized images generated in previous steps are used as input to a machine learning model based on the deep learning architecture ResNet (see section 3.6.1.2 below). This section describes the selection and structure of the model, as well as the preprocessing applied to the input data. Furthermore, the training process is presented, including data splitting, augmentation techniques, and chosen hyperparameters, followed by an evaluation of the model's performance on a separate test dataset.

3.6.1 Defining the Model

3.6.1.1 PyTorch

PyTorch is an open-source deep learning framework used to develop and train neural networks. It provides tools for defining neural network architectures, managing training processes, and performing efficient numerical computations. In this study, PyTorch is used to implement, train, and evaluate the convolutional neural network for roof classification. Its flexible architecture and extensive library support make it well suited for machine learning applications involving image-based data (PyTorch, n.d.).

3.6.1.2 ResNet

When studying model architecture, ResNet was presented as suitable for image classification. ResNet is a deep learning architecture that was introduced by He et al. (2015) to address the difficulties of training deep convolutional networks. Increasing the number of layers in a model can improve the performance, however deeper networks often become harder to optimize and may show higher training error. This effect, known as the degradation problem, occurs because adding more layers makes it difficult for the network to learn useful mappings, even when overfitting is not present.

To overcome this problem, He et al. (2015) propose a residual learning framework. Instead of directly learning a mapping $H(x)$, the network is reformulated to learn $F(x) = H(x) - x$, so that the final output is $F(x) + x$. This formulation is implemented by using shortcut connections that bypass one or more layers and add the input directly to the output of the stacked layers. These shortcut connections perform identity mapping and do not introduce additional parameters or computational cost. The network can still be trained end-to-end using standard back-propagation.

Using this design enables very deep networks to be trained effectively. He et al. (2015) demonstrate residual networks with depths up to 152 layers that outperform previous architectures while still maintaining a lower computational complexity.

Given that the project focuses on classifying roof images with varying shapes, sizes, and structural complexity, a model capable of learning detailed visual features was required.

ResNet was therefore considered suitable due to its proven image classification performance and its ability to train deeper networks effectively.

3.6.1.3 Model Implementation

In this project, a ResNet architecture is implemented in Pytorch, without using a pre-trained model. The architecture resembles ResNet-34 and consists of an initial convolutional layer followed by four residual stages containing 3, 4, 6 and 3 residual blocks respectively.

The first convolutional layer uses a 7 × 7 kernel, meaning a filter of size 7 × 7 pixels that moves across the image to extract spatial features. A stride value of 2 is used, this means that the filter moves 2 pixels at a time across the image, reducing the spatial resolution while lowering computational cost. The first layer produces 64 output channels, where each channel represents a learned feature map highlighting different image characteristics. This layer is followed by max pooling, which reduces the image dimensions by retaining only maximum value within small neighborhoods of the feature map. This allows the network to preserve dominant features while decreasing the amount of data processed in later layers.

Each residual block contains two 3 × 3 convolutional layers with batch normalization and ReLU activation function. The convolutional layers are responsible for extracting increasingly complex image features, while batch normalization stabilizes the training process by normalizing intermediate outputs between layers. The ReLU activation function introduces non-linearity by replacing negative values with zero, enabling the network to learn more complex patterns. Shortcut connections are used to add the input of each block to its output, allowing information to bypass intermediate layers and helping the network avoid degradation problems in deeper architerctures.

After the final residual stage, global average pooling is applied to reduce the spatial dimensions of the feature maps into a single feature vector. The resulting vector is passed to a fully connected layer with five output neurons corresponding to the five roof classes used in this study. Since the input data consisted of grayscale roof images, the network was modified to accept one input channel instead of the standard three-channel RGB input.

Before being used as input to the model, all images were passed through a standardized preprocessing pipeline to ensure consistent dimensions. Since the generated roof images varied in size, directly resizing them to a fixed resolution would distort the roof geometry by stretching or compressing the image. To avoid this, zero-padding was first applied to the shorter side of each image until equal height and width was obtained, resulting in a square image. The padding is distributed symmetrically around the image, keeping the roof structure centered within the frame.

This approach preserves the original proportions of the roof shapes while providing a uniform image format suitable for ResNet. After padding, all images were resized to 224 × 224 pixels,

which corresponds to a common input resolution for ResNet-based architectures and ensures that every sample has identical spatial dimensions during training.

3.6.2 Training the Model

During training, the available synthetic training data is divided into a training subset and a validation subset. The training subset is used to update the model parameters, while the validation subset is used after each training cycle to evaluate how well the model performs on unseen synthetic images. This makes it possible to monitor the learning process and detect signs of overfitting.

To improve the model's ability to generalize, data augmentation techniques are applied to the training images. This introduces small random changes to the input images during training in order to increase the variation in the dataset. In this study, random horizontal flipping and small random rotations were used. These transformations help the model learn roof characteristics that are less sensitive to image orientation and positioning.

Training is performed over multiple epochs, where one epoch means that the model has processed all training images once. During each epoch, batches of images are passed through the network and the model produces predictions for each sample. These predictions are compared with the correct class labels using the cross-entropy loss function, which measures how well the predicted classes match the true classes. Lower loss values indicate better performance. The training process was configured using a set of predefined hyperparameters. Some parameters remained fixed throughout the experiments, while others were adjusted depending on the specific training configuration. Table 3.3 presents the hyperparameters and value ranges used during the training experiments.

Table 3.3: Hyperparameters and value ranges used for model training

Hyperparameter	**Value**
Number of classes	3-5
Learning rate	0.0001 - 0.0003
Batch size	32-64
Epochs	30-40
Loss function	CrossEntropyLoss
Optimizer	Adam

The numerical weights inside the neural network were iteratively updated using the Adam optimization algorithm (Kingma & Ba 2015), which adjusts the network weights iteratively to reduce the loss function during training. After each epoch, both training accuracy and validation accuracy are calculated. Accuracy represents the proportion of correctly classified

images and is used as the main performance metric during training. Monitoring these values over time makes it possible to evaluate the model progress and compare different training settings.

3.6.3 Testing the Model

After the training process is completed, the final model is evaluated using the separate test dataset. These images are not used during training and validation, which means they provide an independent assessment of how well the model can generalize to real roof data. This step is important for determining whether the patterns learned from the synthetic training images can be transferred successfully to the real-world data. Several evaluation metrics are used during the testing phase to assess the classification performance of the model. Classification accuracy is calculated as the proportion of correctly classified test images out of the total number of test samples (Sokolova & Lapalme, 2009).

In addition to classification accuracy, precision, recall, and F1-score are used to evaluate the classification performance for each roof class. Precision describes how many images predicted as a certain class are correctly classified, while recall measures how many images belonging to a class are successfully identified by the model. The F1-score combines precision and recall into a single metric by calculating their harmonic mean. These metrics provide a more detailed evaluation of the model performance, particularly in cases where some roof classes are more difficult to classify or where the class distribution is uneven (Sokolova & Lapalme, 2009).

To obtain a more detailed understanding of the model behavior, a confusion matrix is generated. A confusion matrix shows how many images from each true class are predicted as each possible class. Figure 3.29 illustrates an example of a confusion matrix with three different classes. It provides insight into how well the model performs for each class, highlighting both correct classifications and misclassifications (Sokolova & Lapalme, 2009). This information is valuable for identifying recurring classification errors and evaluating the strengths and limitations of the model.

Figure 3.29: An example of a confusion matrix

The classified images are also saved together with their predicted and true labels to visually inspect the individual predictions. This makes it possible to identify recurring error patterns and image characteristics that influence the model's decision. The predicted roof types generated by the ResNet model are stored in a separate CSV file (*"Predicted_Roofs.csv"*). This file is used in the subsequent reconstruction of the 3D building models.

3.7 Generation of 3D Reconstruction from Classification

The reconstruction process aims to generate 3D building models by combining building footprints with predicted roof types and height information. By assigning geometric and semantic attributes to each building footprint, the resulting models can be reconstructed with different roof geometries representing the classified roof structures. To reconstruct the building models of the study area, the same rule package (*"Roof_Types.rpk"*) previously used for synthetic data generation is applied again in FME. The reconstruction process combines building footprints from the study area with CSV files generated in FME and through Python scripts. Building and roof heights are obtained from the first CSV file ("*Roof_heights.csv*") generated in FME, while the predicted roof types are obtained from the second CSV file ("*Predicted_Roofs.csv*") generated through a Python script.

These datasets are merged using the *FeatureMerger* tool to assign all required attributes to each building. The same tool is subsequently used to merge the attributes with the geometry of the building footprints. Additional processing steps were carried out to remove unwanted features, including objects with null values and roof types that are not required for the analysis. This is done using the *Tester* and *Counter* tools.

Before generating the final models, the roof type values are renamed using the *AttributeValueMapper* tool to correspond to the naming convention recognized by the *CityEngineModelGenerator*. The predicted roof classes together with the extracted height attributes are used in the JSON configuration supplied to the *CityEngineModelGenerator* transformer. Together with the previously generated CGA rule package, these parameters are used to reconstruct simplified 3D building models. The overall workflow is illustrated in Figure 3.30, while Figure 3.31 presents the generated building models.

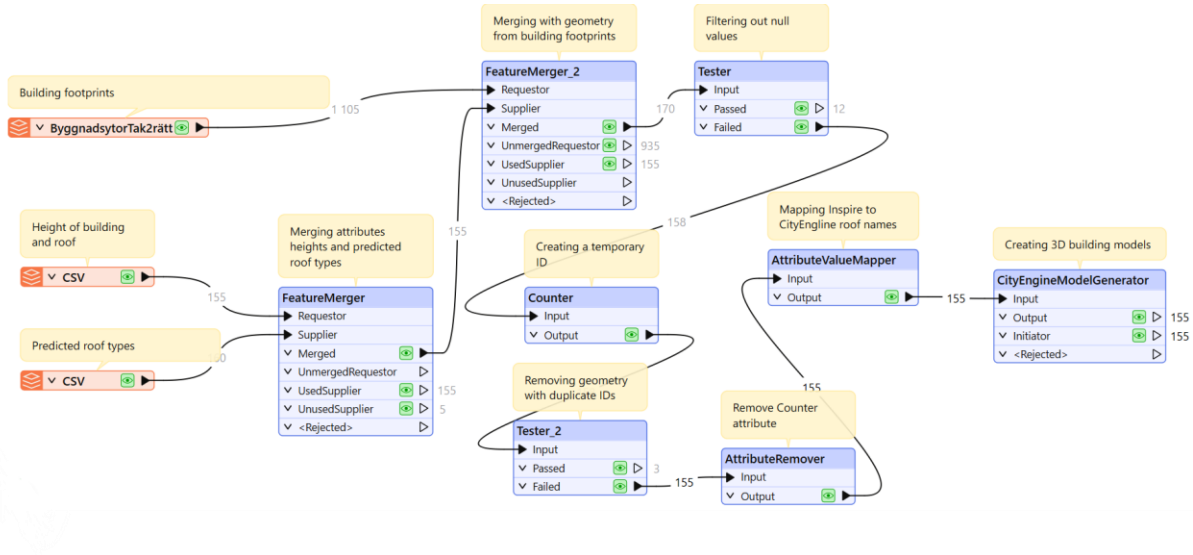


Figure 3.30: Overview of the workflow for reconstructing 3D building models with predicted roof types.

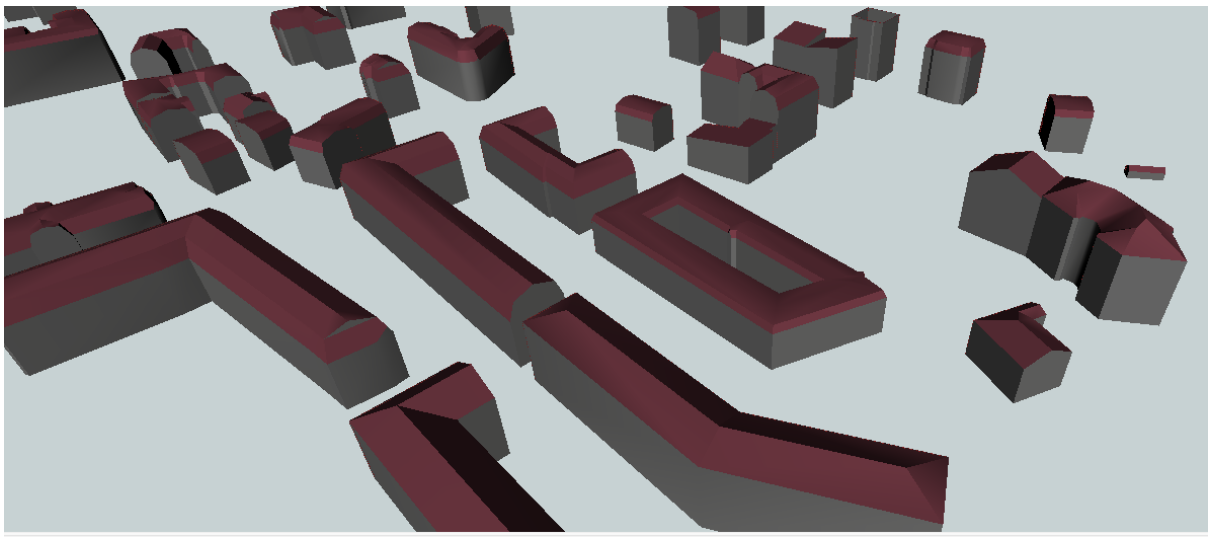


Figure 3.31: Reconstructed building models generated from the classified roof types

Part IV

Analysis and Conclusions

4. Result

In this chapter, the results of the roof type classification experiments are presented using appropriate evaluation metrics. Throughout the study, several parameters were varied in order to evaluate their influence on the classification performance.

In addition, qualitative observations from the classified roof images are included to provide further insight into the behavior of the model and the types of errors that occur during classification.

4.1 Number of Training Images

4.1.1 Only Synthetic Generated Training Images

4.1.1.1 Classification Accuracy

The size of the training dataset is gradually increased to observe the impact it has on the model performance. The synthetic dataset is split into 80% for training, 10% for validation and 10% for testing. For example, a dataset containing 1024 images per class corresponds to approximately 820 training images, 102 validation images and 102 testing images per class after the dataset split. This split allows the majority of the synthetic data to be used for training. Table 4.1 presents the classification accuracy achieved for different synthetic dataset sizes.

Table 4.1: Synthetic dataset size per class and corresponding classification accuracy.

Total images per class	**Accuracy**
1024	93.36%
3879	98.40%
7869	98.50%
11039	98.88%

Table 4.1 shows that increasing the synthetic dataset size improves the classification accuracy on synthetic data. However, the improvement becomes smaller as the dataset size increases.

4.1.1.2 Confusion Matrices

Table 4.1 presents the classification accuracy achieved for different synthetic training dataset sizes. Although the classification accuracy improves as the dataset size increases, the confusion matrices show similar overall misclassification patterns.

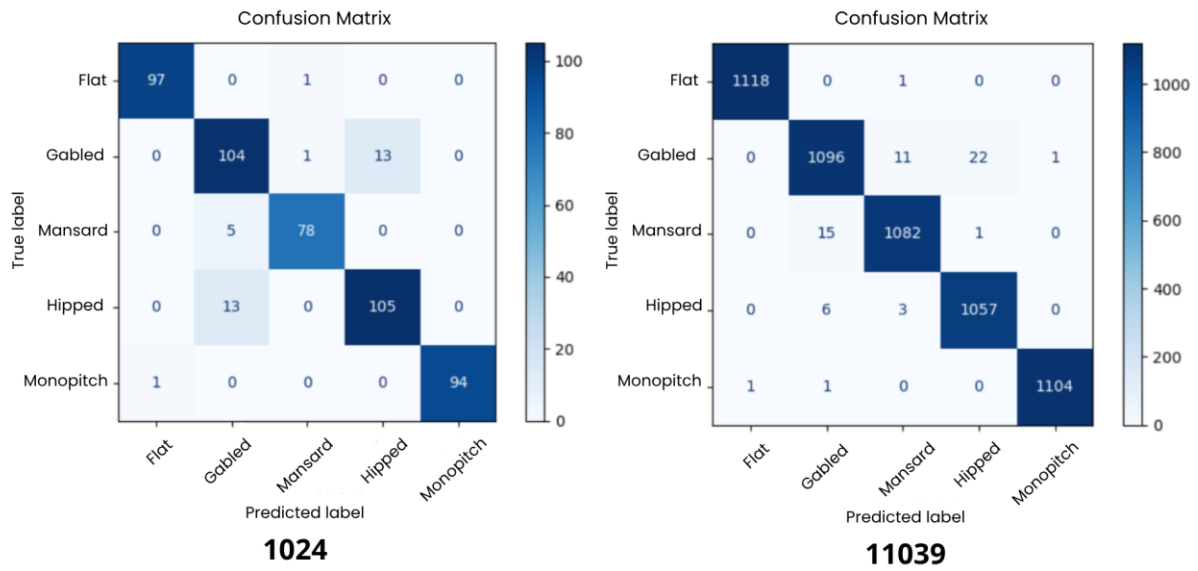


Figure 4.1: Confusion matrices for models trained using 1024 and 11039 synthetic training images per class. The bold numbers below each confusion matrix indicate the number of training images per class used in the corresponding experiment.

The confusion matrices in Figure 4.1 show that Gabled and Hipped roofs are the roof types most frequently misclassified, often being predicted as each other. This pattern is more pronounced for the model trained with 1024 images per class, while the model trained with 11039 images shows fewer misclassifications overall.

4.1.1.3 Classification Report

Table 4.2 presents the precision, recall, and F1-score for the best-performing model evaluated on synthetic roof images. Similar to the confusion matrices, the classification reports show highly consistent results across the different dataset sizes.

Table 4.2: Precision, recall, and F1-score for the best-performing model evaluated on synthetic roof images.

Roof type	**Precision**	**Recall**	**F1-score**
Flat	1.00	1.00	1.00
Gabled	0.98	0.97	0.98
Mansard	0.99	0.99	0.99
Hipped	0.98	0.99	0.99
Monopitch	1.00	1.00	1.00

The classification report for the best-performing model shows consistently high precision, recall and F1-score values across all roof classes. Flat and Monopitch roofs achieve near-perfect classification performance, while slightly lower values are observed for Gabled roofs.

4.1.2 Combination of Synthetic Training Images and Real Test Images

4.1.2.1 Classification Accuracy

The real-world test dataset consists of 155 images, while the size of the synthetic dataset is gradually increased following the same procedure as in Section 4.1.1.1.. In this experiment the synthetic dataset is split into 90 % for training and 10 % for validation, while the real LiDAR-derived roof images are used exclusively for testing. For example, a dataset containing 1024 images per class corresponds to approximately 922 training images and 102 validation images per class after the dataset split. Table 4.3 presents the classification accuracy achieved for different synthetic dataset sizes evaluated on the real-world test dataset.

Table 4.3: Synthetic dataset size per class and corresponding classification accuracy on real LiDAR-derived roof images.

Synthetic images per class	**Accuracy**
1024	47.74 %
3879	54.84 %
7869	64.52 %
11039	65.16 %

The results show that increasing the synthetic dataset size improves the classification accuracy on real LiDAR-derived roof images. The largest increase in accuracy occurs up to 7869 training images per class, while only minor improvements are observed beyond this point.

4.1.2.2 Confusion Matrices

Figure 4.2 presents the confusion matrices for the models trained using 1024, 3879, 7869 and 11039 synthetic training images per class.

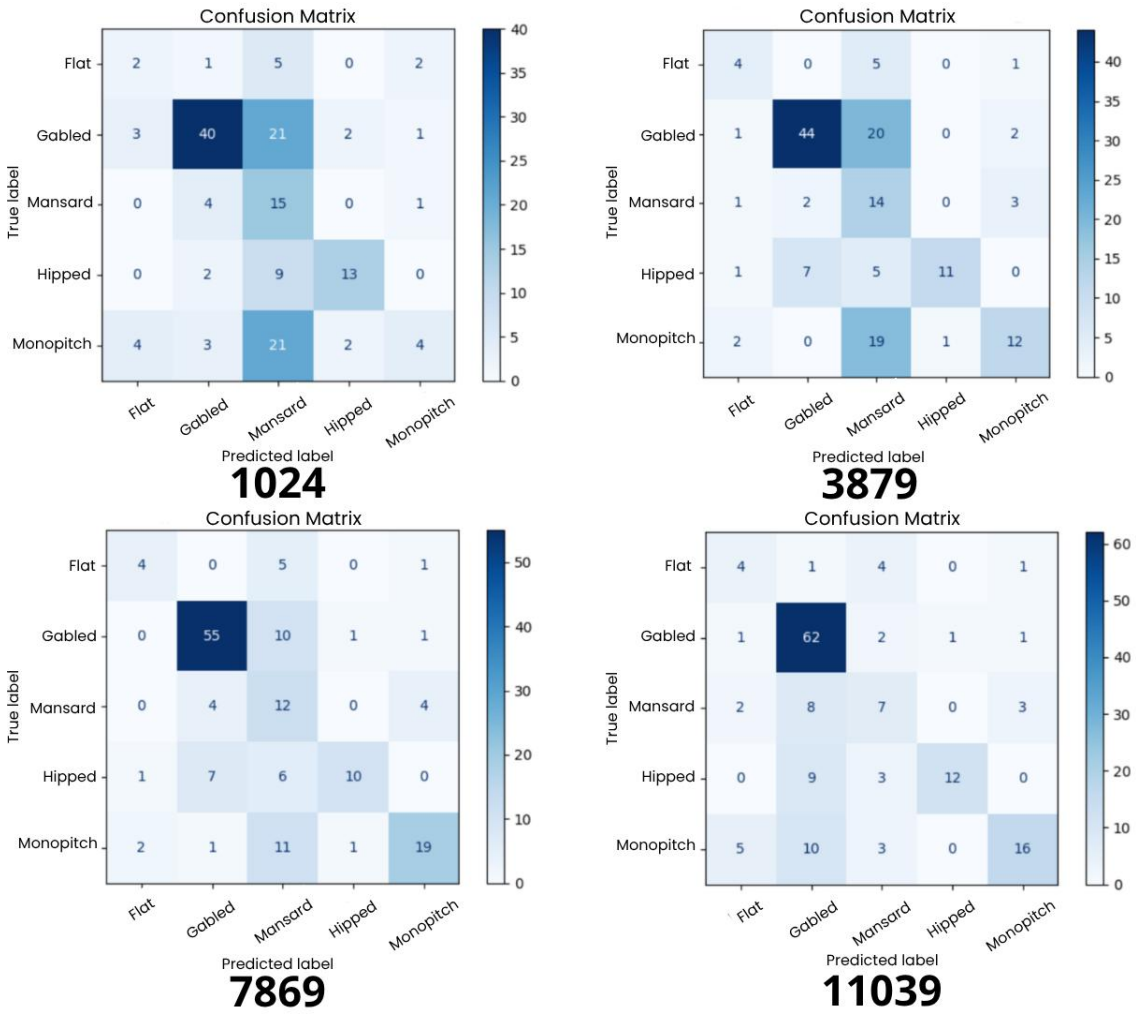


Figure 4.2: Confusion matrices for models trained using different numbers of synthetic training images per class and evaluated on real LiDAR-derived roof images. The bold numbers below each confusion matrix indicate the number of training images per class used in each experiment.

The confusion matrices show that the number of correct classifications increases as the training dataset becomes larger. The model trained using 1024 images per class shows frequent misclassifications between several roof types, particularly between Gabled, Mansard, and Monopitch roofs. As the dataset size increases, the number of correct classifications becomes higher and the confusion between roof classes decreases.

Gabled roofs achieve the highest classification performance for the larger training datasets, while Mansard roofs remain difficult to classify correctly across all experiments. Several roof types, particularly Gabled and Monopitch roofs, are repeatedly misclassified as Mansard, indicating persistent confusion between Mansard roofs and other roof classes across all experiments.

4.1.2.3 Classification Report

Table 4.4 presents the precision, recall and F1-score for the best-performing model evaluated on real LiDAR-derived roof images. Similar to the confusion matrices, the classification reports show noticeable differences in performance between the roof classes.

Table 4.4: Precision, recall, and F1-score for the best-performing model evaluated on real LiDAR roof images.

Roof type	Precision	Recall	F1-score
Flat	0.57	0.40	0.47
Gabled	0.82	0.82	0.82
Mansard	0.27	0.60	0.38
Hipped	0.83	0.42	0.56
Monopitch	0.76	0.56	0.64

Gabled roofs achieve the highest overall classification performance, with high precision, recall, and F1-score. Hipped roofs also achieve high precision values, indicating that predictions of this roof type are often correct, although the lower recall values show that several Hipped roofs are misclassified as other roof types.

Mansard roofs achieve comparatively low precision values but higher recall values, indicating that the model frequently predicts the Mansard class for roofs belonging to other categories. Flat roofs achieve the lowest overall classification performance among the evaluated roof types.

4.2 Number of Classes

The synthetic dataset is used for training and validation, while the real LiDAR-derived roof images are used as a separate test dataset. To evaluate how the number of roof classes affects the classification performance, experiments are conducted using different class combinations. In each experiment, one or more roof classes are excluded from both the synthetic training dataset and the real-world test dataset. Images belonging to the removed classes are therefore not included in either the training or evaluation process.

4.2.1 Combination of Synthetic Training Images and Real Test Images

4.2.1.1 Classification Accuracy

Table 4.5 below presents the average accuracy for the best performing model when experimenting with different numbers of classes.

Table 4.5: Number of classes and corresponding accuracy on test data.

Number of classes	Class names	Accuracy
5	Flat, Gabled, Hipped, Mansard and Monopitch	65.16 %
4	Flat, Gabled, Hipped and Monopitch	79.41%
3	Gabled, Hipped and Monopitch	86.10%

Table 4.5 shows that the accuracy increases as the number of classes is reduced. When using five classes, the model achieved an accuracy of 65.16% (as presented in section 4.1.2.1 above). When reducing the number of classes to four, the accuracy increased to 79.41%. This result indicated that the classification task becomes significantly easier when the number of classes is reduced.

4.2.1.2 Confusion Matrices

The confusion matrix for the five-class setup is presented in Figure 4.2, illustrating the distribution of predictions across the different roof types. The corresponding confusion matrix for the four-class setup is presented in Figure 4.3. Compared to the five-class setup, the overall level of confusion between the roof types is reduced. Gabled roofs achieve the highest classification performance for the larger training datasets, while Hipped roofs remain more difficult to classify correctly across the experiments. Some misclassification still occurs, particularly between geometrically similar roof types, but they are less frequent than in the five-class setup.

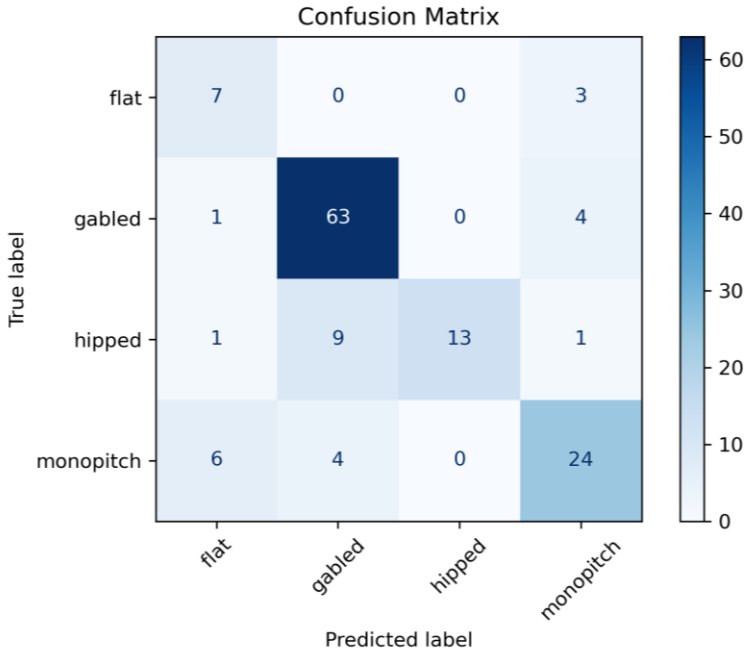


Figure 4.3: Confusion matrix for model trained on four classes

The confusion matrix for the three-class setup is presented in Figure 4.4. Compared to the four-class setup, the level of confusion between the classes has been further reduced. The classification of Gabled and Monopich roofs has improved. Hipped roofs still remain more difficult to classify correctly across the experiments, with no significant improvement.

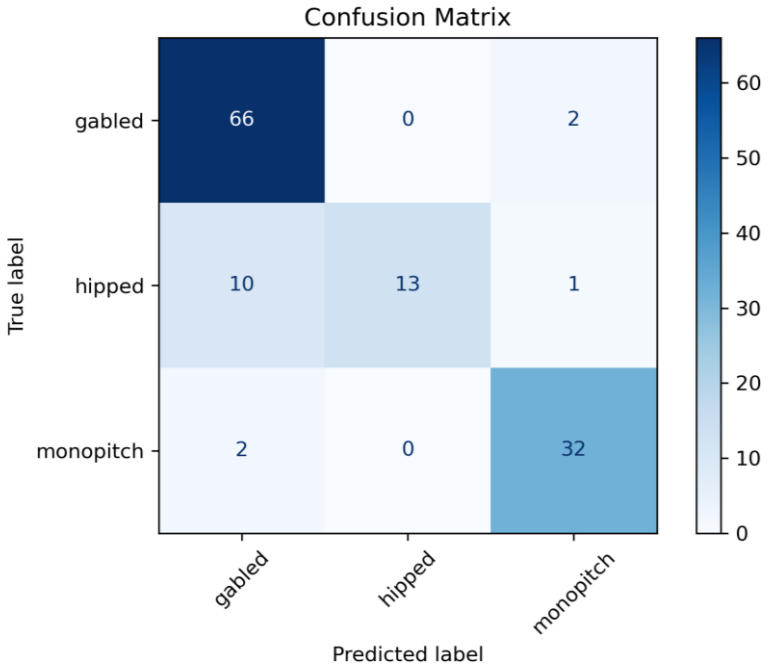


Figure 4.4: Confusion matrix for model trained on three classes

4.2.1.3 Classification Report

Table 4.6 presents the precision, recall and F1-score for the best-performing model evaluated on real LiDAR-derived roof images for four classes. Similar to the confusion matrices, the classification reports show noticeable differences in performance between the roof classes.

Table 4.6: Precision, recall, and F1-score for the best-performing model evaluated on real LiDAR roof images, for 4 classes.

Roof type	**Precision**	**Recall**	**F1-score**
Flat	0.50	0.60	0.55
Gabled	0.83	0.93	0.88
Hipped	0.94	0.63	0.75
Monopitch	0.72	0.68	0.70

Gabled roofs achieve the strongest overall classification performance. The high recall value indicates that the majority of gabled roofs present in the dataset are successfully identified by the model. Hipped roofs obtain the highest precision value among the evaluated roof types,

with a precision of 0.94. This indicates that predictions assigned to the hipped class are highly reliable. However, the recall value of 0.63 shows that a considerable proportion of hipped roofs are instead classified as other roof types, resulting in an F1-score of 0.75.

Monopitch roofs achieve a more balanced distribution between correctly identified roofs and misclassifications, although the overall performance is lower compared to gabled and hipped roofs. Flat roofs achieve the lowest overall classification performance, with a precision of 0.50, a recall of 0.60, and an F1-score of 0.55. These results indicate that flat roofs are more difficult for the model to distinguish from the other evaluated roof types.

Table 4.6 presents the precision, recall and F1-score obtained from the model evaluated on real LiDar images using three roof classes.

Table 4.7: Precision, recall, and F1-score for the best-performing model evaluated on real LiDAR roof images, for 3 classes.

Roof type	**Precision**	**Recall**	**F1-score**
Gabled	0.79	0.94	0.86
Hipped	1.00	0.58	0.74
Monopitch	0.84	0.76	0.80

The model achieves the strongest performance for gabled roofs, with a precision of 0.79, a recall of 0.94, and an F1-score of 0.85. The high recall value indicates that the majority of gabled roofs present in the dataset are identified correctly by the model.

Hipped roofs obtain a precision value of 1.00, meaning that all predictions assigned to this class are correct. However, the recall value of 0.58 indicates that a considerable proportion of hipped roofs are instead classified as other roof types. This results in a lower F1-score of 0.73 compared to the other evaluated classes.

Monopitch roofs achieve a precision of 0.83 and a recall of 0.76, resulting in an F1-score of 0.8000. These values indicate a more even balance between correctly identified roofs and misclassifications compared to the other roof types.

4.3 Visual Evaluation of Classified Roof Images

This section presents a visual evaluation of the classified roof images from the real-world test dataset. Examples of both correctly classified and misclassified roof images are included to provide further insight into the model performance and the types of classification errors observed during the experiments.

4.3.1 Correctly Classified Roof Images

Figure 4.5-4.9 present examples of correctly classified roof images from the real-world test dataset. The examples show that the model successfully classifies several roof geometries despite variations in roof shape, size, orientation, and structural complexity.

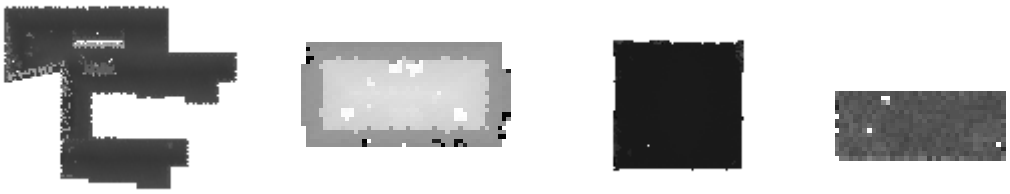


Figure 4.5: Examples of correctly classified flat roofs.

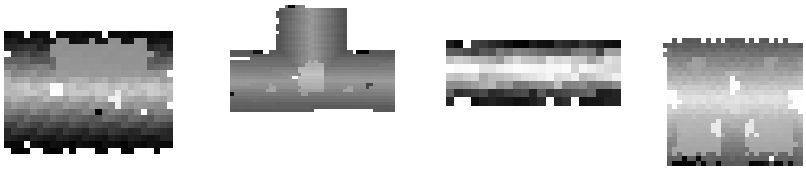


Figure 4.6: Examples of correctly classified gabled roofs.

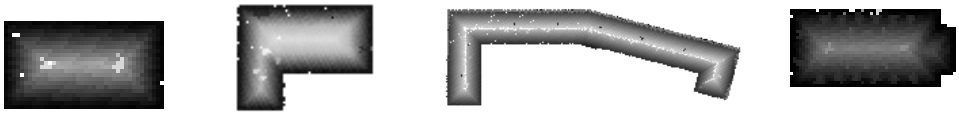


Figure 4.7: Examples of correctly classified hipped roofs.

Figure 4.8: Examples of correctly classified monopitch roofs.

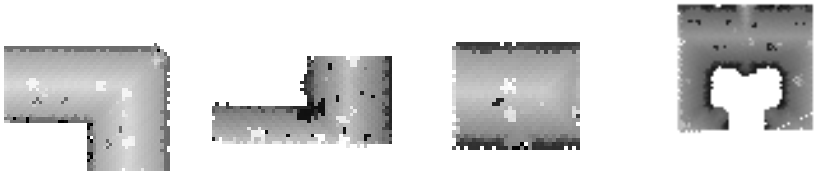


Figure 4.9: Examples of correctly classified mansard roofs.

4.3.2 Misclassified Roof Images

Figures 4.10–4.14 present examples of misclassified roof images from the real-world test dataset, together with their true and predicted roof classes. The examples illustrate the visual similarity between several roof geometries and demonstrate the complexity of the classification task, where some roof types differ only by subtle geometric characteristics.

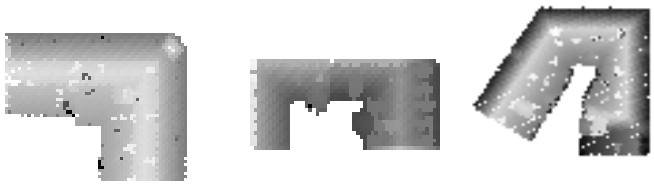


Figure 4.10: Examples of misclassified gabled roof, classified as mansard, monopitch and flat

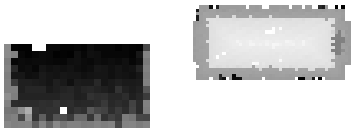


Figure 4.11: Examples of misclassified flat roof, classified as monopitch and mansard

Figure 4.12: Examples of misclassified mansard roof, classified as gabled and monopitch

Figure 4.13: Examples of misclassified hipped roof, classified as gabled, mansard and flat

Figure 4.14: Examples of misclassified monopitch roof, classified as flat, mansard and gabled

4.3.3 Generated 3D Buildings Models from Classified Roofs

Figures 4.15–4.17 illustrate examples of roof structures from the 3D roof model dataset provided by Göteborgs Stad together with their corresponding 3D reconstructions based on the predicted roof type classifications. The reconstructed 3D building models were generated in FME using the same rule package described in Section 3.7.

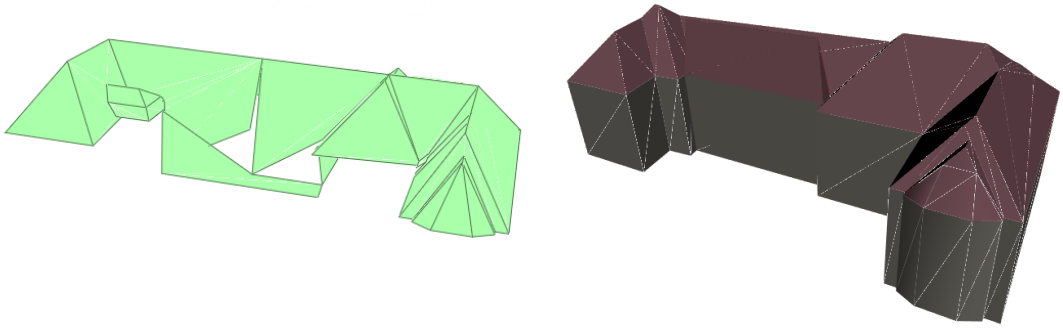


Figure 4.15: 3D roof models from Göteborgs Stad (left) and parametrically generated 3D building based on classification (right). Correctly classified as hipped roof.

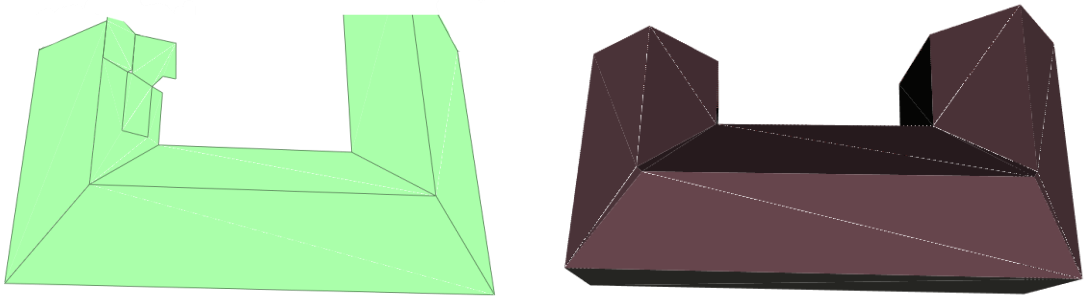


Figure 4.16: 3D roof models from Göteborgs Stad (left) and parametrically generated 3D building based on classification (right). Correctly classified as gabled.

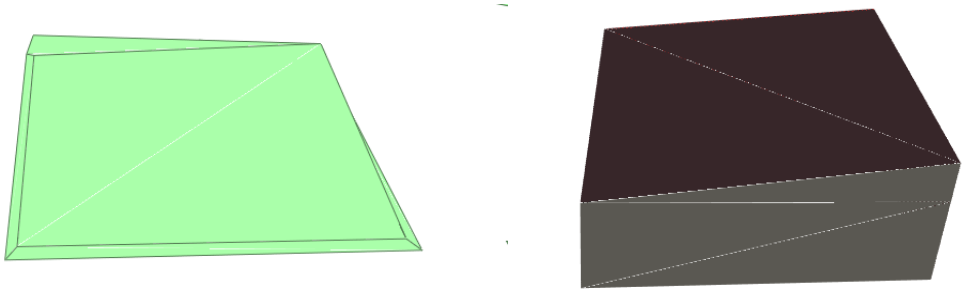


Figure 4.17: 3D roof models from Göteborgs Stad (left) and parametrically generated 3D building based on classification (right). Real roof type flat, misclassified as monopitch.

5. Discussion

5.1 Challenges

5.1.1 Synthetic Versus Real-World Images

One important factor affecting the model performance is the difference between the synthetic training data and the real-world test data. Although the synthetic data images were generated to resemble real roof structures, they still differ from real data in several aspects such as noise and complexity (see Figure 5.1). As a result of this, the model learns patterns from the synthetic images that do not fully represent the roofs in real-world data.

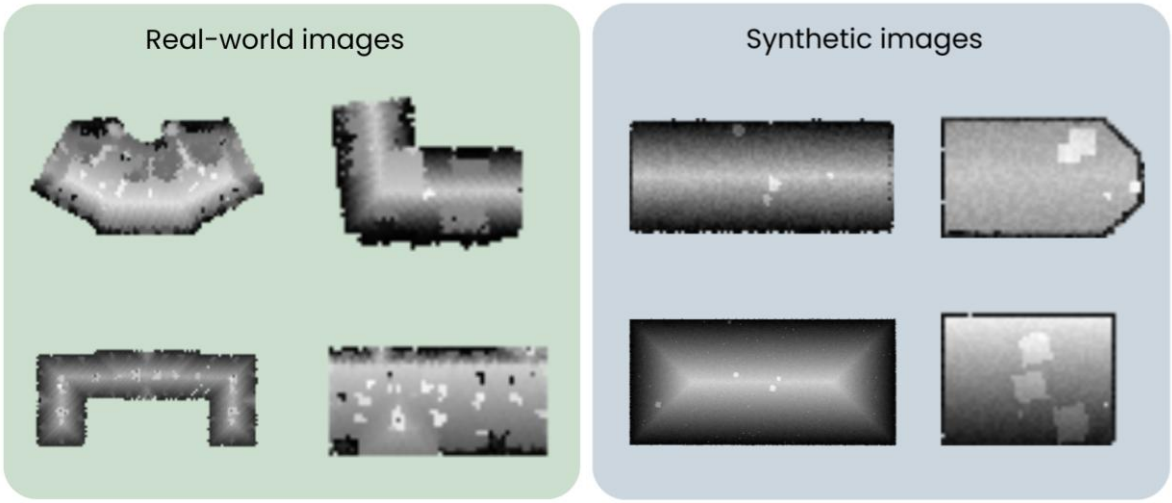


Figure 5.1: Examples of rasterized point clouds from real roofs in Gothenburg (left) and synthetic roofs (right).

The difference between the training and test datasets can reduce the model's ability to generalize to unseen real images. While the model achieves high accuracy in the synthetic data (see table 4.1), the performance decreases when evaluated on real roof images (see table 4.2). For example, when using 11 039 synthetic training images per class, the model achieves an accuracy of 98.88% on synthetic test data, while the accuracy on real test data reaches only 65.16%. Similar differences can be observed for all evaluated dataset sizes. These results indicate that the model is able to learn and classify synthetic roof patterns effectively, but that the learned representations do not transfer equally well to real-world roof images.

In this study, synthetic images generally contain cleaner and more simplified roof representations compared to the real-world data. Real roof images include additional variation caused by factors such as noise in the point cloud data, varying roof complexity or differences in roof height distribution. Consequently, certain roof features that are clear in the synthetic data become less distinguishable in the real data, making classification more difficult for the model.

5.1.2 Similarity in Rasterized Images

An additional challenge is that certain roof types appear very similar in the rasterized images, which makes them difficult to distinguish for the model. This is particularly evident in the real-world data for roof types such as mansard and gabled roofs as well as for flat and monopitch roofs (see Figure 5.2). When converted from point cloud data to raster images, some of the geometric characteristics that distinguish these roof types become less pronounced. For example, differences in slope or subtle variations in roof structure may not be clearly visible, especially when the resolution is limited or when only a portion of the roof is captured within the footprint.

As a result, these roof types are more prone to confusion, which is also reflected in the confusion matrices where misclassification between similar classes occur more frequently. This effect is further supported by the results from the experiments with different numbers of classes (see table 4.4). When reducing the number of classes from five to four, the overall classification accuracy increases noticeably. In this case, the Mansard class was removed entirely, which simplifies the classification task. This indicates that certain roof types, such as Mansard roofs, are more difficult to distinguish and classify correctly, likely due to their visual similarity to other roof types and their less clearly defined characteristics in the rasterized images. By removing this class, the overall ambiguity in the classification task is reduced, resulting in improved performance.

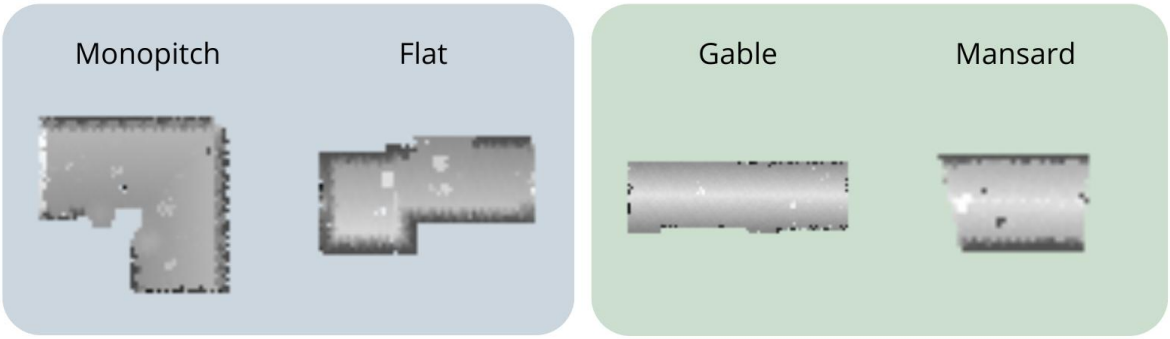


Figure 5.2: Examples of similar classes in the real data.

5.1.3 Physical Buildings Versus Registered Buildings

Another aspect affecting the classification performance is that the study uses building footprints from the building registry (*registerbyggnad*), which do not always correspond to complete physical buildings. As a result, real-world roof structures are often not confined to a single building footprint. In many cases, a continuous roof structure can extend across multiple adjacent footprints, even though these are represented as separate buildings in the dataset (see Figure 5.3). This increase the complexity of assigning a single roof type to each individual footprint, as the visible roof geometry may represent only a small section of a larger roof structure.

As illustrated in Figure 5.3, the roof corresponding to the footprint on the far right is difficult to classify, as it is only partially hipped. In this case, the roof does not fully match a single,

well-defined roof type, which introduces ambiguity in the labeling process. To ensure consistency across the dataset, a decision was made to classify all roofs that are hipped on at least one side as hipped. However, it has not been evaluated whether this labeling strategy is optimal, and alternative interpretations could be equally valid.

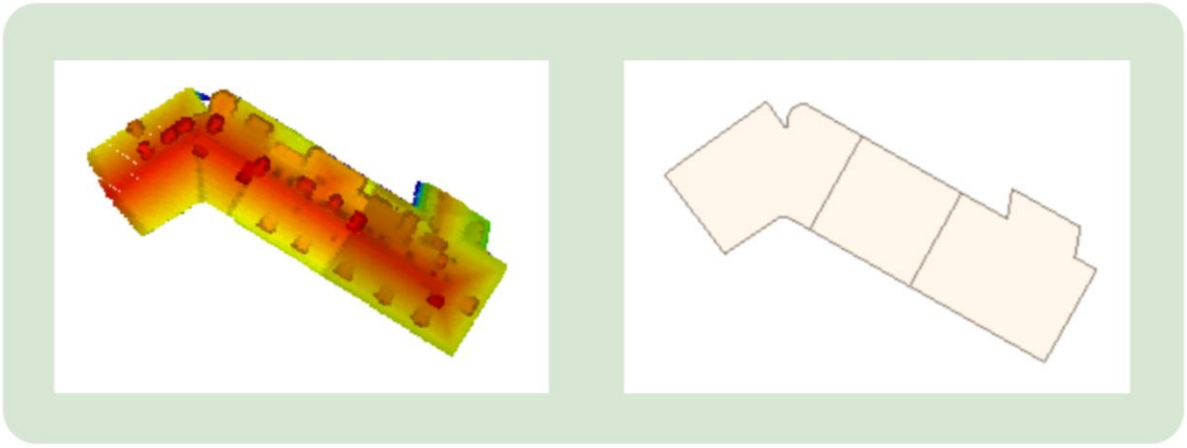


Figure 5.3: Example of a point cloud (left) and the corresponding building footprint (right).

The presence of multiple roof structures, or only partial roof structures, within a single building footprint is difficult to reproduce in the synthetic training data. In the data generation process, each footprint is treated independently and assigned a single roof type using the rule package and the *CityEngineModelGenerator* transformer in FME. As a result, all synthetic roofs are cleanly separated and fully contained within their respective footprints. This creates a simplified representation where each image corresponds clearly to one roof type, without ambiguity or overlap between buildings.

Such difference in training and test data makes classifying the roof type of an individual footprint more complex, as the visible geometry may only represent a portion of a larger roof structure. This can lead to incomplete or misleading roof representations in the rasterized images where only a section of the roof is visible within the footprint. Consequently, the model may struggle to classify such cases correctly, as the input does not fully correspond to the patterns it has learned from the synthetic data. This further contributes to the gap between the synthetic data and the real data.

5.1.4 Ambiguity in Manual Roof Classification

The complexity of roof structures makes the manual classification of the real-world roof types more challenging. Since the roof associated with a single building footprint can consist of several roof types or a section of a roof type, assigning a single roof type becomes ambiguous. In such cases, the manual classification cannot always rely on clear geometric criteria and instead rely on interpretation. As a result, the manual labeling process is influenced by the author's judgement, which can result in inconsistencies across the dataset.

The difficulties with labeling is further increased by the fact that real-world roofs rarely correspond perfectly to the predefined roof types described in the INSPIRE specification. While these categories provide a useful conceptual framework, many real roofs contain irregular shapes, combinations of multiple roof forms, or modifications that do not clearly fit into a single class. Consequently, even when the overall structure resembles a known roof type, deviations in geometry can make it unclear which category is most appropriate.

Together, these factors introduce a level of uncertainty in the ground truth labels used for evaluation. This uncertainty may affect the reliability of the performance assessment, as some misclassifications by the model could be related to ambiguities in the reference labels rather than purely incorrect predictions.

5.1.5 Impact on Classification Performance

The challenges mentioned above can be reflected in the results presented in Chapter 4. As shown in Table 4.1 and 4.3, the model achieves significantly higher accuracy when both training and testing are performed on synthetic data compared to when the model is evaluated on real-world data. For instance, the model reaches an accuracy of 98.88% on synthetic test data, while the corresponding accuracy on real-world data is 65.16% for the same training configuration.

The large difference in accuracy highlights how the model is able to learn the simplified and well-defined patterns present in the synthetic data, but struggles when these patterns are less distinct or only partially visible in real-world images. Furthermore, the ambiguities in the manual labeling process contribute to misclassifications that are observed in the confusion matrices, where certain roof types are frequently confused.

Taken together, these results indicate that the reduced performance on real world data is not caused by a single factor, but rather by a combination of sources of error.

5.2 Comparison with Previous Studies

The method proposed in this study differs from several previous approaches for roof classification and 3D reconstruction. Traditional geometry-based methods, such as those proposed by Lingfors et al. (2017) and Alexander et al. (2009), rely on extracting geometric features directly from LiDAR point clouds, including slope, aspect and planar roof segments. These methods often use predefined roof templates to classify roof structures. In contrast, the method used in this study is primarily data-driven and relies on a convolutional neural network to learn roof characteristics automatically from rasterized point cloud representations, rather than explicitly defining geometric rules.

Similar to the studies by Castagno and Atkins (2018) and Meng et al. (2023), this study applies deep learning for roof classification using image-based representations of buildings. However, while these studies use manually labeled real-world datasets and, in some cases,

combine LiDAR data with aerial or satellite imagery, the present study focuses on training the model using synthetically generated data. This enabled the creation of large labeled training datasets with limited manual work, but also introduces challenges related to the difference in synthetic and real data.

This study also shares similarities with recent research that converts point cloud data into 2D image representations before applying machine learning methods. For example, Yarroudh et al. (2024) transform façade point clouds into images for object detection, while Castagno and Atkins (2018) generate depth-image representations from LiDAR data for roof classification. In a similar manner, the present study converts roof point clouds into rasterized top-down images that can be processed by a convolutional neural network.

Unlike several previous studies that focus on reconstructing detailed 3D geometry directly from point clouds, this study instead investigated whether roof structures can first be classified and described parametrically using machine learning. This provides a simplified but structures representation of roof geometry, which could potentially support future reconstruction workflows and automated city modelling.

5.3 Areas of Improvement

The results of this study indicate that there are several areas where the proposed method can be improved. One of the most important aspects is the gap between the synthetic training data and the real-world test data. Although the synthetic data enables the generation of large labeled datasets, it does not fully capture the variability and complexity of real roof structures. Improving the synthetic data to emulate the real roof types could enhance the model's ability to generalize to real-world data.

Another area of improvement concerns the representation of building footprints, where each footprint corresponds to a register building. In the current approach, each register building is treated as an independent unit and assigned a single roof type. However, a physical building may extend across multiple register buildings, and real-world roofs may also consist of several roof types within the same physical structure. This simplification introduces ambiguity in the classification task. Future improvements could therefore include methods that better account for the relationship between register buildings and physical buildings, for example by merging adjacent footprints or by allowing multiple roof types to be assigned to a single building.

The preprocessing of the data also presents opportunities for improvement. The transformation of point cloud data into raster images leads to a loss of geometric information, particularly regarding height variations and fine structural details. While normalization simplifies the learning process, it may also reduce important differences between roof types. Alternative representations, such as preserving absolute height information or combining raster images with additional features, could increase the area of applications for the project.

Additionally, the manual classification of real-world test data introduces uncertainty in the evaluation process. As discussed, ambiguities in roof structures and deviations from standard roof type definitions make consistent labeling difficult. Improving the quality and consistency of the ground truth data, for example by refining the classification criteria, could lead to more reliable evaluation of the model.

Another possible improvement concerns the handling of roof structures that do not correspond clearly to any of the selected roof classes. In the current study, real-world roofs that could not be labeled according to Inspire roof types were excluded from the test dataset. While this simplified the classification task and reduced ambiguity during manual labeling, it also limits the applicability of the method to more complex or irregular roof structures. To handle this, one possible approach could be to introduce an additional "unknown" class representing roof types that do not fit the predefined categories. Another possibility could be to allow multi-label classification for buildings containing characteristics of several roof types.

5.4 Future Applications and Development

The results of this study indicate that the proposed approach has potential for automated roof type classification, but further development is required before it can be applied in practical use cases. One possible future application is within GIS and urban modelling workflows, where automated classification of roof types could support the generation or enrichment of 3D city models. By combining roof type classification with geometric parameters, such as building height and roof slope, the method could contribute to the reconstruction of parametric building models suitable for applications such as visualization, urban planning and analyses such as daylight and water flow.

However, for such applications, the model must be able to handle the variability and complexity of real-world data. This highlights the need for improved generalization, either through more representative training data or by incorporating real-world data into the training process. Additionally, methods that allow for more detailed representation of roof structures, such as identifying multiple roof types within a single building, could further increase the applicability of the approach.

Overall, the method demonstrated potential as a component in automated workflows for building analysis and modelling, but requires further refinement to ensure robustness and reliability when applied to real-world datasets.

The increasing availability of open geospatial data may further improve the applicability of methods such as the one proposed in this study. Many municipalities and organizations, including Göteborgs Stad through its open data service, provide publicly accessible geospatial datasets such as building footprints, elevation models, and other spatial data. This creates opportunities for applying similar workflows in different geographic areas without requiring proprietary datasets.

5.5 Limitations

One limitation of this study is the uneven distribution of roof types within the real-world test dataset. Certain roof types, such as gabled roofs, are highly represented while others, such as flat roofs, occur much less frequently. As a result, the reported accuracy is influenced by the class distribution within the selected study area. For example, gabled roofs achieve a high classification accuracy and are also overrepresented in the test dataset. This increases the overall accuracy, which may give a misleading impression of the model performance since the remaining roof classes are not classified with the same level of accuracy.

Another limitation is that roof type distribution may vary considerably between different geographic regions and urban environments. Architectural styles, building regulations and historical building patterns can influence the presence of certain roof structures. The roof characteristics present in the study area may therefore not be representative of other locations, meaning that the classification performance observed in this study may differ when applied to datasets from other regions.

6. Conclusions

The aim of this study was to classify real roof structures based on point cloud data and represent them using parametric descriptions. To achieve this, a workflow was developed in which synthetically generated 3D building models were converted into point clouds and raster images, and used to train a neural network based on the ResNet architecture. The trained model was then applied to real-world data.

Regarding the first research question, previous studies have approached the detection and classification of roof structures using both geometry-based and machine learning-based methods. Traditional approaches commonly rely on geometric analysis of point clouds, DSM data, building footprints, or algorithms such as RANSAC to reconstruct roof surfaces. More recent studies increasingly apply machine learning and deep learning methods to classify roof structures directly from point cloud or image-based representations. Several studies also highlight challenges related to noisy point cloud data, complex roof geometries, and the limited availability of labeled training datasets. To address these limitations, this study explores the use of synthetically generated point cloud data derived from parametric 3D building models as training data for a deep learning model.

Regarding the second research question, the results show that roof types can be successfully classified by a machine learning model when trained and tested on synthetic data, achieving high classification accuracy. This demonstrates that the proposed method is effective in learning roof characteristics from controlled and well-defined datasets. However, when applied to real-world data, the classification accuracy decreases significantly. This indicates that the model has limited ability to generalize from synthetic to real data, primarily due to differences in noise, complexity, and representation between the datasets.

Regarding the third research question, the study also demonstrates that roofs can be described parametrically based on point cloud data, as the classification results can be linked to predefined roof categories. The predicted roof classes can subsequently be used in the reconstruction of simplified 3D building models. However, several challenges were identified that affect this process, including cases where roofs extend across multiple building footprints and situations where a single footprint contains several roof types, and limitations in how real roofs correspond to standardized classifications such as INSPIRE. These factors complicate both the classification task and the extraction of reliable parametric descriptions.

In conclusion, the proposed approach demonstrated potential for automated roof type classification and parametric representation of buildings. However, further development is required to improve the model's ability to generalize to real-world data and to better handle the complexity and variability of real roof structures.

References

3CIM. (2023). *3CIM2.0.* GitHub. https://github.com/3CIM/3CIM2.0

Abdi, B., Axelsson, B., Calvert, M., Carlsson, K., Christensen, U., Gardevärn, D., Harrie, L., Hirsch, G., Jeansson, E., Kadric, Z., Lord, J., Loreman, A., Olsson, P., Persson, A., Setterby, O., Inge, M., Sjöberger, M., Stewart, P., Rudenå, A. & Uggla, M. (2023). *3CIM - Smart Built Environment projekt 2020-2022.)*
https://www.smartbuilt.se/media/hobe5pn5/3cim_slutrapport.pdf

Alexander, C., Smith-Voysey, S., Jarvis, C. & Tansey, K. (2009). *Integrating building footprints and LiDAR elevation data to classify roof structures and visualise buildings.*
https://doi.org/10.1016/j.compenvurbsys.2009.01.009

Audrey Eads. (2025, December 15). *Data manipulation: Definition, Importance and Tips*. Indeed. https://www.indeed.com/career-advice/career-development/data-manipulation

Badwi, I. M., Ellaithy, H. M., & Youssef, H. E. (2022). *3D-GIS Parametric Modelling for Virtual Urban Simulation Using CityEngine.* Annals of GIS, 28(3), 325–341.
https://doi.org/10.1080/19475683.2022.2037019

Baltsavias, E. P. (1999). *Airborne laser scanning: Basic relations and formulas*. ISPRS Journal of Photogrammetry and Remote Sensing, 54(2–3), 199–214.
https://doi.org/10.1016/S0924-2716(99)00015-5

Bello, S. A., Yu, S., Wang, C., Adam, J. M., & Li, J. (2020). *Review: Deep learning on 3D point clouds*. Remote Sensing, 12(11), 1729. https://doi.org/10.3390/rs12111729

Biljecki, F., Ledoux, H., Stoter, J. & Zhao, J. (2014) *Formalisation of the level of detail in 3D city modelling.* Computers, Environment and Urban Systems, 48, 1-15.
https://doi.org/10.1016/j.compenvurbsys.2014.05.004

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(4), 2842-2889. https://doi.org/10.3390/ijgi4042842

Biljecki, F., Ledoux, H. & Stoter, J. (2016). *An improved LOD specification for 3D building models.* Computers, Environment and Urban Systems, 59, 25-37.
https://doi.org/10.1016/j.compenvurbsys.2016.04.005

Boltcheva, D., Basselin, J., Poull, C., Barthélemy, H., & Sokolov, D. (2020). *Topological-based roof modeling from 3D point clouds*. *Journal of WSCG, 28*(1–2), 137–146.

https://hal.science/hal-02983073v1/file/G53.pdf

Buyukdemircioglu, M. Kocaman, S., Kada, M. (2022). *DEEP LEARNING FOR 3D BUILDING RECONSTRUCTION: A REVIEW.* https://isprs-archives.copernicus.org/articles/XLIII-B2-2022/359/2022/

Castagno, J. & Atkins, E. (2018). *Roof Shape Classification from LiDAR and Satellite Image Data Fusion Using Supervised Learning.* Sensors, 18(11), 3960. https://doi.org/10.3390/s18113960

Chajaei, F. & Bagheri, H. (2025). *LOD1 3D city model from LiDAR: The impact of segmentation accuracy on quality of urban 3D modeling and morphology extraction*. Remote Sensing Applications: Society and Environment, 38. https://doi.org/10.1016/j.rsase.2025.101534

Chen, M., Liu, X., Zhang, X., Wang, M., & Zhao, L. (2021). *Building Extraction from Terrestrial Laser Scanning Data with Density of Projected Points on Polar Grid and Adaptive Threshold*. Remote Sensing, 13(21), 4392. https://doi.org/10.3390/rs13214392

Chen, R. (2011). *The development of 3D city model and its applications in urban planning.* Proceedings of the 19th International Conference on Geoinformatics (pp. 1–5). IEEE. https://doi.org/10.1109/GeoInformatics.2011.5981007

Diab, A., Kashef, R., & Shaker, A. (2022). *Deep learning for LiDAR point cloud classification in remote sensing*. Sensors, 22(20), 7868. https://doi.org/10.3390/s22207868

Ding, Z., Lu, Y., Shao, S., Qin, Y. Lu, M., Song, Z. & Sun, D. (2025). *Research on 3D Reconstruction Methods for Incomplete Building Point Clouds Using Deep Learning and Geometric Primitives.* Remote Sensing, 17(3), 399. https://doi.org/10.3390/rs17030399

Douglas, D. H., & Peucker, T. K. (1973). Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: the international journal for geographic information and geovisualization, 10(2), 112-122.

Du, K-L., Zhang, R., Jiang, B., Zeng, J. & Lu, J. (2025). *Understanding Machine Learning Principles: Learning, Inference, Generalization, and Computational Learning Theory.* https://doi.org/10.3390/math13030451

El-Mekawy, M., Östman, A. & Hijazi, I. (2012). *A Unified Building Model for 3D Urban GIS*. https://doi.org/10.3390/ijgi1020120

Gröger, G., Kolbe, T.H., Nagel, C. & Häfele, K.H. (2012). *OGC City Geography Markup Language (CityGML) Encoding Standard, ver. 2.0.* http://www.opengeospatial.org/standards/CityGML

Göteborgs Stad. (n.d.). *3D-data*. https://goteborg.se/wps/portal/start/bygga-bo-och-leva-hallbart/lantmateri-kartor-och-matning/kartor-och-geodata/kartprodukter/3d-data

Hanke, F., Bieringer, A., Wysocki, O. & Jutzi, B. (2025). *CM2LoD3: Reconstructing LoD3 Building Models Using Semantic Conflict Maps.* https://doi.org/10.48550/arXiv.2508.15672

Harrie, L. (2024). *Nationella specifikationer för storskaliga geodata.* Smart Built Environment. https://www.smartbuilt.se/Media/olokfab4/nationella-specifikationer-for-storskaliga-geodata.pdf

He, K., Zhang, X., Ren, S. & Sun, J. (2015)- *Deep Residual Learning for Image Recognition.* https://doi.org/10.48550/arXiv.1512.03385

IBM. (n.d.-a). *What is ETL (extract, transform, load)?*. https://www.ibm.com/think/topics/etl

Jeong, G.H. (2020). *Artificial intelligence, machine learning, and deep learning in women's health nursing.* https://doi.org/10.4069/kjwhn.2020.03.11

Jordan, M.I., Mitchell, T.M. (2015) *Machine learning: Trends, perspectives, and prospects.* Science, 349(6245), 255-260. https://doi.org/10.1126/SCIENCE.AAA8415

Jung, A. (2022). *Machine Learning: The Basics.* Springer. https://arxiv.org/pdf/1805.05052

Kingma, D. P., & Ba, J. (2015). *Adam: A Method for Stochastic Optimization.* In *Proceedings of the 3rd International Conference on Learning Representations (ICLR).* https://doi.org/10.48550/arXiv.1412.6980

Kolbe, T. H., Kutzner, T., Smyth, C. S., Nagel, C., Roensdorf, C. & Heazel, C. (2021). *OGC City Geography Markup Language (CityGML) Part 1: Conceptual Model Standard.* Open Geospatial Consortium. https://docs.ogc.org/is/20-010/20-010.html#toc45

Lantmäteriet. (2015). *HMK – Terrester laserskanning 2015*. https://www.lantmateriet.se/globalassets/om-lantmateriet/var-samverkan-med-andra/hmk/handbocker/hmk-terrester_laserskanning_2015.pdf

Lantmäteriet. (2017). *HMK – Flygburen laserskanning 2017*. https://www.lantmateriet.se/globalassets/om-lantmateriet/var-samverkan-med-andra/hmk/handbocker/hmk_flygburen_laserskanning_2017.pdf

Lantmäteriet. (2021b). *HMK – Terrestrisk laserskanning 2021*. https://www.lantmateriet.se/globalassets/om-lantmateriet/var-samverkan-med-andra/hmk/handbocker/hmk-terlas_2021.pdf

Lantmäteriet. (2021a). *Geodetisk och fotogrammetrisk mätnings- och beräkningsteknik*. https://www.lantmateriet.se/globalassets/om-lantmateriet/var-samverkan-med-andra/hmk/kurskompendium/kompendium20211014.pdf

Lantmäteriet. (2022). *Laserdata nedladdning (NH)*. https://www.lantmateriet.se/globalassets/geodata/geodataprodukter/hojddata/pb_laserdata_nedladdning_nh.pdf

Lantmäteriet. (2024a). *Laserdata nedladdning – skog.* https://www.lantmateriet.se/globalassets/geodata/geodataprodukter/hojddata/pb_laserdata_nedladdning_skog.pdf

Lantmäteriet. (2024b). *Nationell dataproduktspecifikation - Byggnad.* https://www.lantmateriet.se/globalassets/temawebbar/smartare-samhallsbyggnadsprocess/nationella-specifikationer/natspec-dps-t-byggnad-version1.0.pdf

Lantmäteriet. (2024c). *Vägledning Nationell specifikation Byggnad.* https://www.lantmateriet.se/globalassets/temawebbar/smartare-samhallsbyggnadsprocess/nationella-specifikationer/vagledning-natspec-byggnad-version1.0.pdf

Lewis, R. & Séquin C. (1998). *Generation of 3D building models from 2D architectural plans.* Computer-Aides Design, 30(10), 765-779. https://doi.org/10.1016/S0010-4485(98)00031-1

Lingfors, D., Bright, J.M., Engerer, N.A., Ahlberg, J., Killinger, S. & Widén, J. (2017). *Comparing the capability of low- and high-resolution LiDAR data with application to solar resource assessment, roof type classification and shading analysis.* https://doi.org/10.1016/j.apenergy.2017.08.045

Liu, X. (2024) *Research on Urban Building Parametric Modelling Method Based on CityEngine.* Journal of World Architecture. 8(1). https://doi.org/10.26689/jwa.v8i1.6188

López, O. A. M., et al. (2022). *Overfitting, model tuning, and evaluation of prediction performance.* Multivariate Statistical Machine Learning Methods for Genomic Prediction. https://www.ncbi.nlm.nih.gov/books/NBK583970/

Madakam, S., Uchiya, T., Mark, S. & Lurie, Y. (2022). *Artificial Intelligence, Machine Learning and Deep Learning.* Management Research and Innovation, 18(1-2), 7-23. https://doi.org/10.1177/2319510X221136682

Meng, S., Soleimani-Babakamali, M. H., & Taciroglu, E. (2023). *Automatic Roof Type Classification Through Machine Learning for Regional Wind Risk Assessment.* https://doi.org/10.48550/arXiv.2305.17315

Microsoft. (n.d.). *What is data integration?* https://azure.microsoft.com/sv-se/resources/cloud-computing-dictionary/what-is-data-integration

Mutreja, G. & Bittner, K. (2025). *Efficient Building Roof Type Classification: A Domain-Specific Self-Supervised Approach.* https://doi.org/10.48550/arXiv.2503.22251

OpenStreetMap contributors. (2026). *OpenStreetMap.* *https://www.openstreetmap.org/#map=13/57.70602/11.96068*

Pantoja-Rosero, B. G., Achanta, R., Kozinski, M., Fua, P., Perez-Cruz, F., & Beyer, K. (2022). *Generating LOD3 building models from structure-from-motion and semantic segmentation*. Automation in Construction, 141, 104430. https://doi.org/10.1016/j.autcon.2022.104430

Park, Y. & Guldmann, J-M. (2019) *Creating 3D city models with building footprints and LIDAR point cloud classification: A machine learning approach.* Computers, Environment and Urban Systems, 75, 76-89. https://doi.org/10.1016/j.compenvurbsys.2019.01.004

Pytorch. (n.d.). *Pytorch*. https://pytorch.org/projects/pytorch/

Rane, N. L., Mallick, S. K., Kaya, Ö., & Rane, J. (2024). *Applications of machine learning in healthcare, finance, agriculture, retail, manufacturing, energy, and transportation: A review.* Deep Science Publishing. https://doi.org/10.70593/978-81-981271-4-3_6

Sadeghi, F., Arefi, H., Fallah, A. & Hanh, M. (2015). *3D Building Façade Reconstruction Using Handheld Laser Scanning Data.* The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XL-1/W5, 625–630. https://doi.org/10.5194/isprsarchives-XL-1-W5-625-2015

Safe Software. (n.d.-a). *Transforming Data*. https://docs.safe.com/fme/html/FME-Form-Documentation/FME-Form/Workbench/transforming-data.htm

Safe Software. (n.d.-b). *What is FME?*. https://docs.safe.com/fme/html/FME-Form-Documentation/FME-Form/Workbench/What_is_FME.htm#WhatisETL

Sajadian, M. & Arefi, H. (2014) *A DATA DRIVEN METHOD FOR BUILDING RECONSTRUCTION FROM LiDAR POINT CLOUDS.* https://isprs-archives.copernicus.org/articles/XL-2-W3/225/2014/

Shen, N., Wang, B., Ma, H., Zhao, X., Zhou, Y., Zhang, Z., & Xu, J. (2023). *A review of terrestrial laser scanning (TLS)-based technologies for deformation monitoring in engineering.* Measurement, 223, 113684. https://doi.org/10.1016/j.measurement.2023.113684

Shorten, C. & Koshgoftaar, T. M. (2019). *A survey on Image Data Augmentation for Deep Learning.* Journal of Big Data, 6. https://doi.org/10.1186/s40537-019-0197-0

Sokolova, M., & Lapalme, G. (2009). *A systematic analysis of performance measures for classification tasks*. Information Processing & Management, 45(4), 427–437. https://doi.org/10.1016/j.ipm.2009.03.002

Stockholms stad. (2024). *Kartor och geodata*. https://kartor.stockholm/kartor-geodata/

Syed Abdul Rahman, S. A. F., Abdul Maulud, K. N., Ujang, U., Wan Mohd Jaafar, W. S., Shaharuddin, S., & Ab Rahman, A. A. (2024). *The Digital Landscape of Smart Cities and Digital Twins: A Systematic Literature Review of Digital Terrain and 3D City Models in Enhancing Decision-Making. SAGE Open, 14*(1). https://doi.org/10.1177/21582440231220768

Uggla, M., Olsson, P., Abdi, B., Axelsson, B., Calvert, M., Christensen, U., Gardevärn, D., Hirsch, G., Jeansson, E., Kadric, Z., Lord, J., Loreman, A., Persson, A., Setterby, O., Sjöberger, M., Stewart, P., Rudenå, A., Ahlström, A., Bauner, M., ... Harrie, L. (2023). *Future Swedish 3D City Models—Specifications, Test Data, and Evaluation.* ISPRS International Journal of Geo-Information, 12(2), 47. https://doi.org/10.3390/ijgi12020047

Wehr, A., & Lohr, U. (1999). *Airborne laser scanning—an introduction and overview.* ISPRS Journal of Photogrammetry and Remote Sensing, 54(2–3), 68–82. https://doi.org/10.1016/S0924-2716(99)00011-8

Woodbury, R. (2010). *Elements of parametric design*. Routledge.

Xiao, A., Zhang, X., Shao, L., & Lu, S. (2023). *A survey of label-efficient deep learning for 3D point clouds*. arXiv. https://doi.org/10.48550/arXiv.2305.19812

Yarroudh, A., Kharroubi, A., Jeddoub, I. & Billen, R. (2024). *Automatic upgrade of 3D building models to LoD3 using 3D Point Clouds and Grounding DINO.* https://isprs-archives.copernicus.org/articles/XLVIII-2-W8-2024/471/2024/

Zhao, T., Xiong, T., Li, M. & Li, Z. (2025). *Automatic Reconstruction of 3D Building Models from ALS Point Clouds Based on Façade Geometry.* ISPRS International Journal of Geo-Information, 14(12), 462. https://doi.org/10.3390/ijgi14120462

Zeng, H., Wu, J. & Furukawa, Y. (2018) *Neural Procedural Reconstruction for Residential Buildings.* Computer Vision - ECCV 2018, 759-775. https://doi.org/10.1007/978-3-030-01219-9_45

Zhu, Z., Lin, K., Jain. A. K., & Zhou, J. (2023). *Transfer Learning in Deep Reinforcement Learning: A Survey.* IEEE Transactions on Pattern Analysis and Machine Intelligence, 45(11), 13344-13362. https://doi.org/10.48550/arXiv.2009.07888

Miljö- och geovetenskapliga institutionen, Lunds Universitet

Student-examensarbete (seminarieuppsatser) i geografisk informationsteknik. Uppsatserna finns tillgängliga på LUP student papers.

Serie examensarbete i geografisk informationsteknik

1. Patrik Carlsson och Ulrik Nilsson (2010) Tredimensionella GIS vid fastighetsförvaltning

2. Karin Ekman och Anna Felleson (2010) Att välja grundläggande karttjänst - Utveckling av jämförelsemodell och testverktyg för utvärdering

3. Jakob Mattsson (2011) Synkronisering av vägdatabaser med KML och GeoRSS - En fallstudie i Trafikverkets verksamhet

4. Patrik Andersson and Anders Jürisoo (2011) Effective use of open source GIS in rural planning in South Africa

5. Nariman Emamian och Martin Fredriksson (2012) Visualiserig av bygglovsärenden med hjälp av Open Source-verktyg - En undersökning kring hur man kan effektivisera ärendehantering med hjälp av en webbapplikation

6. Gustav Ekstedt and Torkel Endoff (2012) Design and Development of a Mobile GIS Application for Municipal FieldWork

7. Karl Söderberg (2012) Smartphones and 3D Augmented Reality for disaster management - A study of smartphones ability to visualise 3D objects in augmented reality to aid emergency workers in disaster management

8. Viktoria Strömberg (2012) Volymberäkning i samhällsbyggnadsprojekt

9. Daniel Persson (2013) Lagring och webbaserad visualisering av 3D stadsmodeller - En pilotstudie i Kristianstad kommun

10. Lisette Danebjer och Magdalena Nyberg (2013) Utbyte av geodata - studie av leveransstrukturer enligt Sveriges kommuner och landstings objekttypskatalog

11. Alexander Quist (2013) Undersökning och utveckling av ett mobilt GISsystem för kommunal verksamhet

12. Nariman Emamian (2014) Visning av geotekniska provborrningar i en webbmiljö

13. Martin Fredriksson (2014) Integrering av BIM och GIS med spatiala databaser – En prestandaanalys

14. Niklas Krave (2014) Utveckling av en visualiseringsapplikation för solinstrålningsdata

15. Magdalena Nyberg (2015) Designing a generic user interface for distribution of open geodata: based on FME server technology

16. Anna Larsson (2015) Samredovisning av BIM- och GIS-data

17. Anton Lundkvist (2015) Development of a WEB GI System for Disaster Management

18. Ellen Walleij (2015) Mapping in Agricultural Development – Introducing GIS at a smallholders farmers' cooperative in Malawi

19. Frida Christiansson (2016) Lagring av 3D - geodata - en fallstudie i Malmö Stad

20. Lisette Danebjer (2016) Methodology for creating and modifying distributed topologically structured geographical datasets

21. Jeanette Dunn Ekelund (2016) En jämförelse av algoritmer och resultat för flödesberäkning i QGIS/GRASS och ArcGIS

22. Ebba Gröndahl och Frida Thorman (2016) Verksamheters optimala läge i staden och hur de är lokaliserade idag

23. Gunnar Rolander (2017) Data transformation using linked data ontologies

24. Måns Andersson och Moa Eklöf (2017) Stilsättning av geografiska data

25. Josefine Axelsson (2018) Automatisering av bygglovsansökningsprocessen med stöd av BIM och GIS

26. Leonard B. O. Berge (2018) Uppdatering och visualisering av stadsmodell med stöd av konverterade BIM-modeller

27. Rickard Ingesson & Gabriella Olsson (2019) Publicering av geografiska data på webben : En utvärdering av programsystem med fokus på öppen källkod

28. Alfred Hildingson & Patrik Sylve (2020) Visualisering av stadsmodeller på webben : Jämförande studie mellan CityGML och CityJSON

29. Isabelle Andersson (2020) Indoor positioning systems in office environments : a study of standards, techniques and implementation processes for indoor maps

30. Sebastian Roos & August Cnattingius (2021) Covid-19-pandemins konsekvenser på svenskt näringsliv - en Space Syntax analys : Hur har konkursutsattheten och arbetslösheten förändrats för detaljhandel, hotell och restauranger till följd av restriktioner som begränsar besöksnäring?

31. Emelie Ulin (2021) Simuleringar i planeringsprocessen med 3D-stadsmodeller

32. Alfred Hirschfeld & Christoffer Karlsson (2022) Designing and implementing a geospatial mobile application

33. Andreas Ahlström (2022) 3D-stadsmodeller för mindre kommuner - vad är behoven och vilka standarder krävs?

34. Jorun Westman (2022) Looking for shrubs in an alvar - Investigating classification of orthophotos as a way of mapping shrub species Juniperus communis and Dasiphora fruticose on Stora Alvaret, Öland

35. Sophia Bladh & Ylva Kjellberg (2023) Enhetlig visualisering av geotekniska data och geokonstruktioner - En intervju- och implementeringsstudie

36. David Andersson & Beatrice Ekström (2023) Scenariohantering med parametrisk design i planeringsprocessen

37. Alexander Brynolf (2023) Undersökning av tillgänglighetsmått

38. Axel Andersson (2024) Kvalitetsbeskrivning och kvalitetspåverkande faktorer vid produktion av TIN-modeller

39. Hanna Lundström (2024) Gränssnittsstandarden ”OGC API - Environmental Data Retrieval” och dess användning för nedladdning av väderdata

40. Erik Lökholm (2025) 3DCityDB som leveransdatabas - En fallstudie för Malmö stad och 3CIM

41. Matilda Bengtsson & Alba Björkman (2025) Auto-Calibration of Geolocation & Heading Direction: Segmentation based Matching between Network Camera & Satellite Images

42. Justin Hellsten (2025) Automatisk generering av transportytor för 3D-stadsmodeller enligt 3CIM-specifikationen

43. Isabelle Hilding (2025) Automatisk identifiering av outnyttjade byggrätter – En studie om digital detaljplaneinformation och utveckling av ett digitalt sökverktyg

44. Didrick Kananen (2025) Monitoring tree harvest in selection forestry with UAV data and machine learning

45. Iris Åberg & Thelma Åsvärd (2026) Parametric Roof Modelling Using a Machine Learning Approach