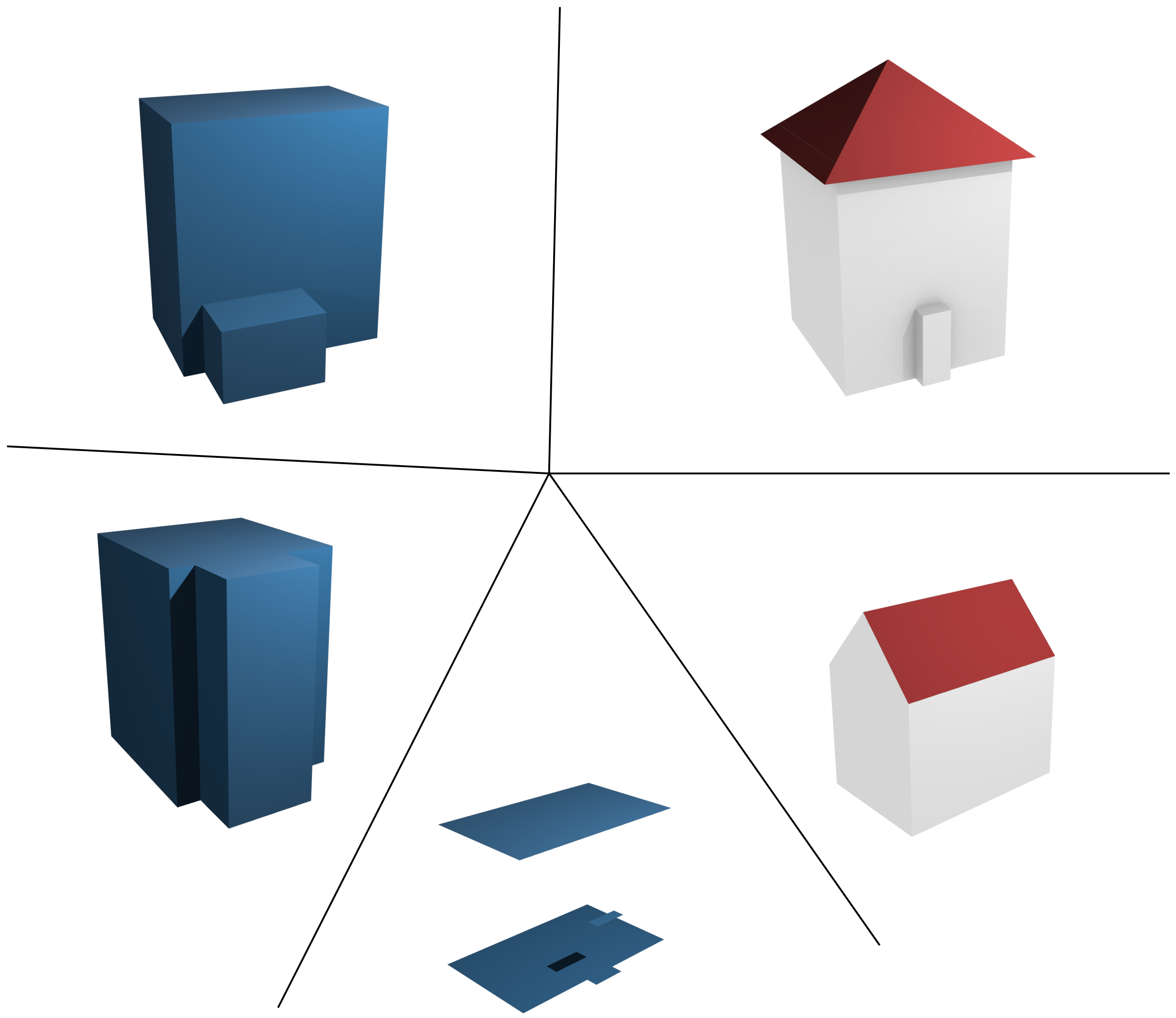
MSc thesis in Geomatics for the Built Environment

Exploring the automatic Level of Detail inference for the validation of buildings in 3D city models

Balázs Dukai

2018



EXPLORING THE AUTOMATIC LEVEL OF DETAIL INFERENCE FOR THE VALIDATION OF BUILDINGS IN 3D CITY MODELS

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of

Master of Science in Geomatics for the Built Environment

by

Balázs Dukai

January 2018

The work in this thesis was made in the:

3D geoinformation group
Department of Urbanism
Faculty of Architecture & the Built Environment
Delft University of Technology

Supervisors:	Dr. Filip Biljecki
	Dr. Hugo Ledoux
	Anna Labetski
Co-reader:	Dr. Jorge Lopes Gil

ABSTRACT

There are several 3D city models available openly, worldwide. These models are used in various applications, from which many expects a homogeneous Level of Detail (LoD). Validating the accuracy of the LoD of a model requires the inference its LoD class and its conformance to the real-world object. This process quickly becomes infeasible for large models when done manually. Yet there is no automatic method for LoD inference and validation. Therefore the thesis proposes a method to automatically infer the geometric LoD (LoD0-2.3) in 3D city models.

A central aspect of this work is the use of machine-learning to classify building models based on their LoD. It follows the assumption that a process is possible where a classifier trained in a synthetic 3D city model containing all LoD classes, and applied in *real* city model. Therefore ten geometry measures (features) are computed from the objects and tested with six classification algorithms. The six experiments the transferability of a classifier from the synthetic city model to the real one, multi-class (LoD0-2.3) and binary (LoD2 or not) classification, and the effect of LoD class imbalance by introducing various amounts of LoD1 objects into the LoD2 model. Furthermore, by using a point cloud as ground truth, this explored the possibility of validating the inferred LoD classes.

The results indicate that the classifier is not transferable to the real data set when trained on the synthetic city model, which is probably due to the significant difference in object shapes between the two models. Binary classification outperforms the multi-class case and it is favourable for LoD validation where the main question is whether the model conforms the stated LoD or not. Finally, class-imbalance can reduce the classification with as much as 20%.

ACKNOWLEDGEMENTS

Thanks to everyone directly or indirectly involved throughout the research. Particularly to my supervisors Dr. Filip Biljecki, Anna Labetski, Dr. Hugo Ledoux for their patience, ideas and support all along the process. A thorough review takes time and expertise and for these I'm grateful for Dr. Jorge Lopez Gil. My gratitude extends to virtualcitySystems for providing the Amsterdam data set that I extensively used for this research. Finally, I would like to thank my family and friends for supporting me in starting and finishing the Geomatics programme.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

1.1 MOTIVATION

3D city models of several cites are openly available in different detail and quality [Cit, nd]. All of the twenty-six listed data sets are less than ten years old, and more than half of them are less than five years old. The tender for Rotterdam 3D 2.0 (2016) [Rot, 2016] asked for offers for the generation of a 3D city model consisting of 205.000 buildings. The model will also be openly available, and is meant to be used in the design processes for urban and regional development, communication and public participation in construction projects, management of built assets and optimization of energy management [GIM, 2016], [rot, nd]. This shows the current interest in 3D city models and will from some cities to invest in their acquisition and open dissemination in order to facilitate well informed decisions about the built environment.

Biljecki et al. [2015] lists twenty-nine use cases where 3D city models are applied, among them are visibility analysis, routing and change detection in urban inventory. Many of these applications analyse a phenomenon in relation to the shape of the buildings (e.g.wind flow analysis) [Biljecki et al., 2015]. However, the accuracy of many analysis relies on the assumption that the level of geometric abstraction in the city model is consistent in each and every building. In other words, the Level of Detail (LoD) is homogeneous in the city model. For example if the roof structure of the buildings in the city model are assumed to be fully reconstructed, then a building model with a flat top surface has a flat roof in reality as well. For example, in case the roof reconstruction failed for some buildings, usually the model is reconstructed with a flat top surface (e.g.Figure 2.5). Then these incorrect models are considered by the analysis as buildings with flat roofs. Such discrepancy in the abstraction from reality can cause inaccurate analysis results as it was shown by Biljecki et al. [2016c].

The concept of Level of Detail (LoD) for 3D city models was borrowed from computer graphics, where it was pioneered by Clark [1976] and it describes the complexity of the representation of a geographic object [Biljecki et al., 2013]. The City Geography Mark-up Language (CityGML) standard prescribes how to model various parts of a city (e.g.buildings, furnitures, vegetation, terrain), and it also categorises objects in the city model based on their level of geometric and semantic abstraction [Kolbe, 2009]. CityGML describes five LoD categories (0-4), ranging from a 2D building footprint (LoD0) to a complete building model, including its interior (LoD4), see Figure 1.2 [Gröger et al., 2012]. Biljecki et al. [2016b] removed ambiguity from the LoD categories by providing more well defined requirements for each. This improvement resulted in an increase from five to sixteen classes (Figure 1.1).

The LoD plays an important role in data maintenance and conversion. Change detection methods for CityGML use the LoD of the building and its features in order to prevent the matching of surfaces of different LoDs and thereby creating objects with inconsistent detail [Redweik and Becker, 2015]. In case of solar irradiation studies Wieland et al. [2015] states that buildings modelled without the roof structure introduce a bias to the results due to incorrect solar incidence angles. Also Wieland et al. [2015] underline that building models without roof structure are questionable for estimating solar irradiation potential. In case of noise pollution

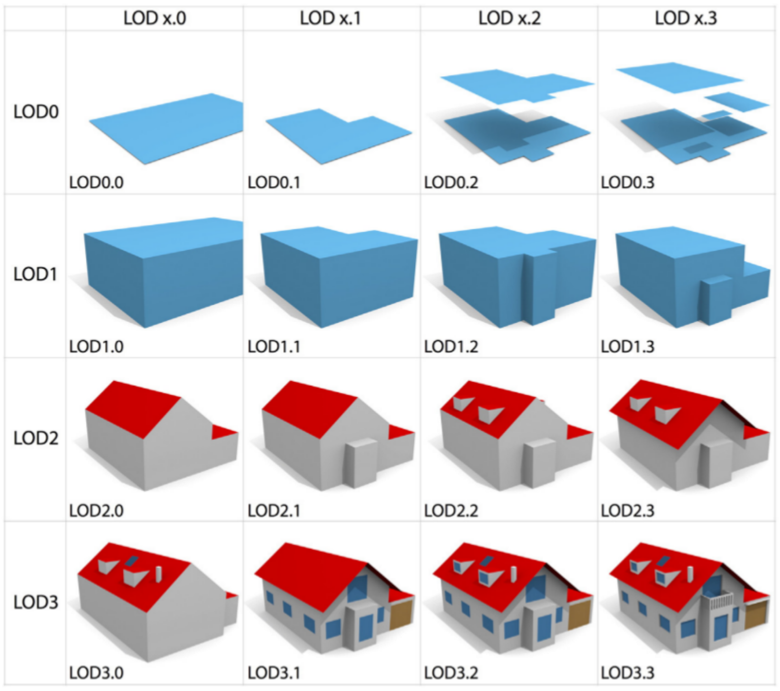


Figure 1.1: The refined LoD specification by 'Biljecki et al. [2016b]'

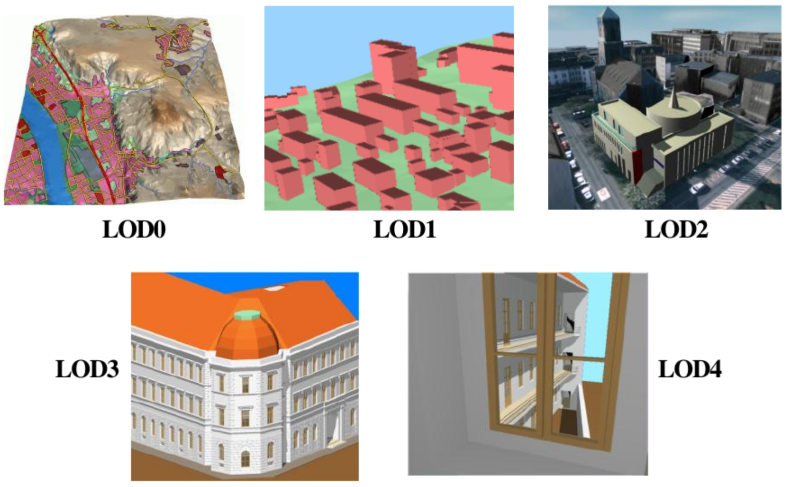


Figure 1.2: Examples of LoD0-4 models [Gröger et al., 2012]

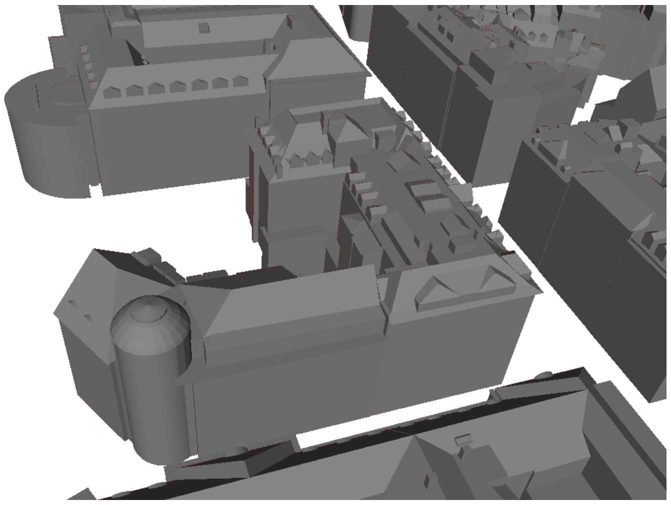


Figure 1.3: Zürich old town in LoD2, with dormers and roof overhangs. Source: reference Zürich data

modelling, Van Renterghem and Botteldooren [2010] show that the roof shape is a very important aspect. For example gabled roofs reflect the sound waves in a significantly different pattern than flat roofs.

1.2 PROBLEM STATEMENT

Considering building models, the current LoD requirements as they are prescribed by Gröger et al. [2012] leave room for personal interpretation [Biljecki, 2017], in aspects that are relevant in evaluating the suitability of a data set. These aspects include the presence of roof overhangs (in LoD2), the size and presence of roof super-structures, or in LoD1 models the height of the top surface in relation to the real hight of the roof. The improved LoD specification by Biljecki et al. [2016b] clarifies many of the ambiguous cases, however it is a recent study and not part of the standard, thus its widespread adoption is still expected. Ambiguity in the LoD specification leads to 3D city models that are considered equal in their LoD class, but differ in their actual level of approximation, see Figures 1.3, 1.4.

For reconstructing building models in LoD2, common approaches are based on on parametric modelling, segmentation or Digital Surface Model simplification [Haala and Kada, 2010]. In parametric modelling, the most suitable roof form is picked from a set of roof types and fitted to the building model [OSM, 2017], [vir, nd]. However, usually there are a limited number of available roof types and a vast diversity of buildings. Thus the roof fitting process fails occasionally, resulting in either a model with a flat top (LoD1 instead of LoD2, e.g.Figure 2.5), or model which roof significantly differs from its real-world counterpart e.g.Figures 2.4, A.3, A.4. In either case, the failure is not reflected in the meta-data.

In some data sets a few characteristic buildings, landmarks are intentionally modelled in higher detail than the rest, see Figure 1.5. It is usually the case with data sets that are intended for navigation, or that were generated by volunteers, such as OpenStreetMap.

For these reasons 3D city models are often heterogeneous in their LoD and contain invalid models. There are building models which validity cannot be immediately determined, but requires manual work. Thus it can be seen that manual validation is not feasible for large city models, such as Rotterdam 3D 2.0.

Although there are existing solutions for the geometric validation of 3D city models such as val3dity [Ledoux, 2013], and citydoctor [cit, nd], there are no existing solutions or research about the LoD validation of 3D city models. In the first com-

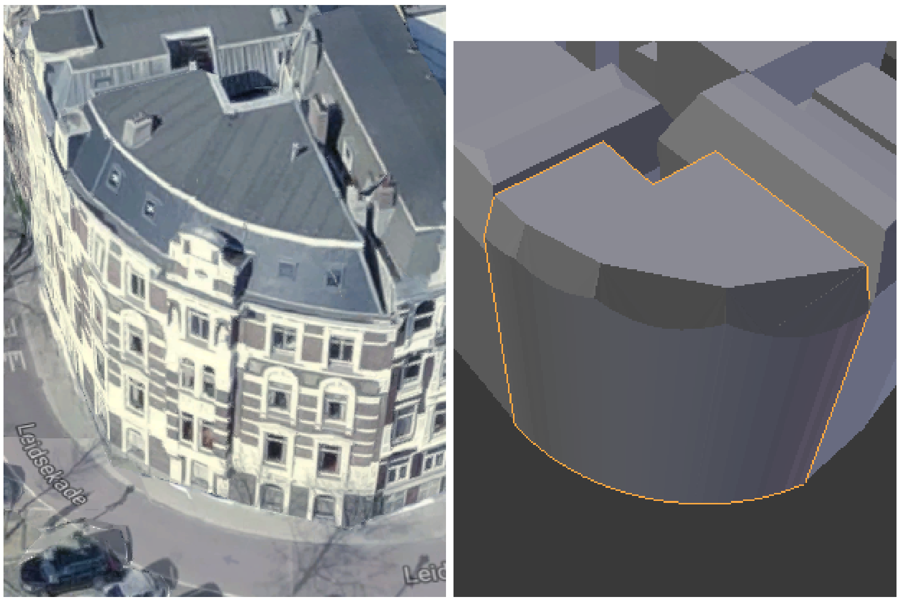


Figure 1.4: Building model in Amsterdam city center in LoD2, without dormers and with simplified roof geometry. Source: Google Earth for the aerial image; VirtualCitySystems for the 3D model

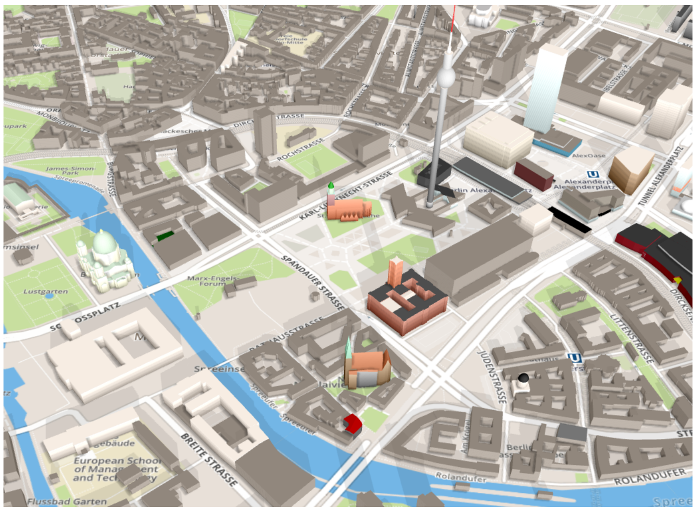


Figure 1.5: Landmarks are modelled in a higher LoD than the rest of the buildings in Berlin. Source: https://osmbuildings.org/

prehensive study on the LoD in 3D city models Biljecki [2017] argues for the importance of LoD in the context of data quality. While Biljecki [2017] sets the foundations for further research, there was no work carried out about inferring the LoD in 3D city models. Wong and Ellul [2016] researched the use of geometry-based metrics for evaluating the fitness-for-purpose of a 3D city model, but they did not relate the metrics to the LoD classes.

1.3 RESEARCH QUESTIONS AND METHOD

As stated in the previous sections, the central problem in validating the LoD is determining the LoD class of the building model at hand. Once the actual LoD class of the model is known, it can be compared to the stated LoD. Therefore the primary focus of the current work is answering the question of *How can the geometric Level-of-Detail be inferred automatically in 3D city models?*. In order to answer the research question, there are several sub-questions to be answered as well:

- To what extent is it possible to automatically validate the LoD of a 3D city model?
  - Without using reference data?
  - With the help of reference data?
- What is a suitable method to classify 3D building models in terms of their LoD?
  - Which building geometry measures can be used to uniquely describe each LoD class?

This work focuses on the automatic inference of the LoD in 3D city models, while considering only buildings of LoD0-LoD2 Furthermore, emphasis is given to data sets with heterogeneous LoD, as these data sets are challenging to validate manually. The thesis strives to provide a generic method of geometric LoD validation, therefore semantic information is not considered by the method. The thesis does not provide an extensive evaluation of learning algorithms for LoD inference, however it will test a set of algorithms which are selected based on theoretical considerations.

The main contribution of this work is the theoretical development and testing of an automatic LoD inference and validation method. To the extent of my knowledge this is the first study that applies machine learning techniques on 3D city models in order to infer and validate their LoD. In this process the concept of LoD and its validity is explored (Section 2.2), the LoD requirements [Gröger et al., 2012], [Biljecki et al., 2016b] are translated into features suitable for machine learning (Section 3.1.1, 3.1.2), and their performance is tested in several experiments (Section 4).

Therefore the thesis follows the research process illustrated in Figure 1.6. After introducing and defining the problem (Section 1.1, 1.2), the goals of the research are formulated (Section 1.3). In Section 2, the key aspects of the required background knowledge are reviewed, together with the work that has been done in, or closely related to the topic under analysis. Following, Section 3 develops the method for automatic LoD inference, while Section 4 implements and tests the theory. Finally the thesis is concluded by reviewing the outcomes and lessons of the experiments (Section 5).

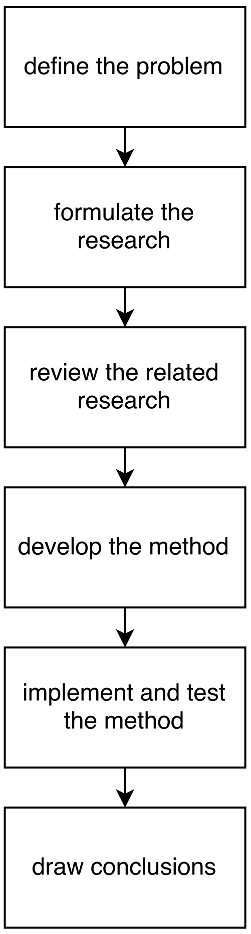


Figure 1.6: Research method of this thesis

2 THEORETICAL BACKGROUND

To my knowledge there is no research published or tool developed that allows the classification of city models according to the CityGML LoD categories. There is however work done on determining the scale of 2D objects in geographical data and also on measuring the geometric complexity of 3D objects in geographical data. Both are relevant for my work.

2.1 THE LEVEL OF DETAIL OF 3D CITY MODELS

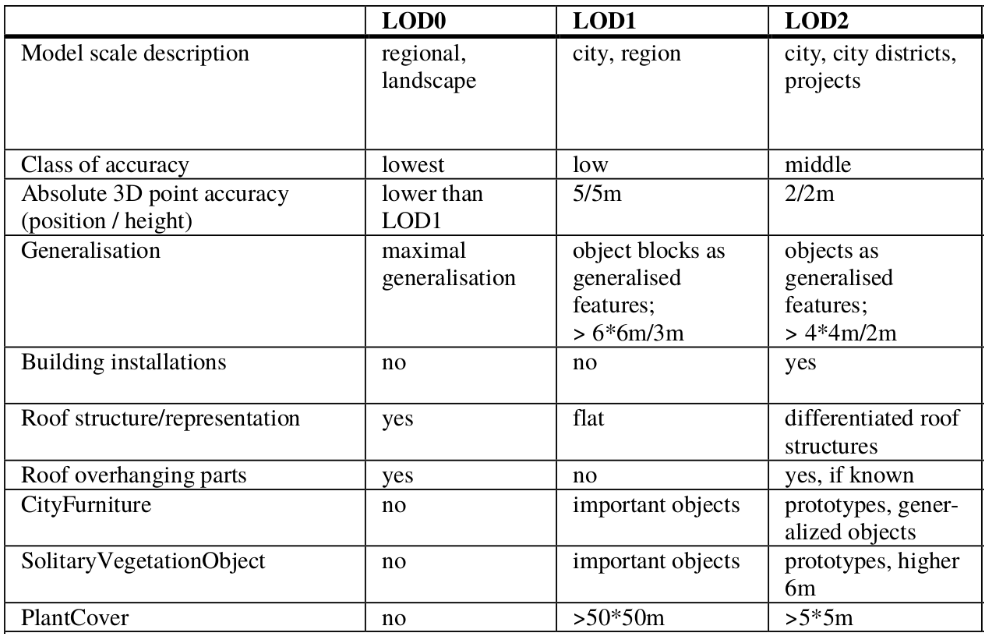
The Open Geospatial Consortium (OGC) developed the City Geography Mark-up Language (CityGML) standard to formalise city modelling and facilitate the sharing and widespread use of these models across many domains. The standard prescribes how to model cities in terms of the required object and their geometric and semantic qualities.

In order to facilitate multi-scale modelling, CityGML introduced the concept of Level of Detail (LoD) for digital city models. The LoD categorises the level of geometric and semantic abstraction of objects in the city model [Kolbe, 2009]. However, the level of detail in digital models in not a new concept, it is widely used in computer graphics to measure the efficiency of the representation. Biljecki [2017] points out that the same measures are not suitable to refer to the detail of city models. While computer graphics use an ordinal scale to refer to more or less efficient models, the LoD classes are categorical, the higher LoD does not necessarily mean a "better" model.

According to Gröger and Plümer [2012] the coarsest level (LoD0) is a two and a half dimensional Digital Terrain Model with the buildings represented by their footprint polygons. The level LoD1 is probably the most commonly seen in city models, where buildings are represented by their prismatic approximations, with flat top surfaces. The level LoD2 adds generalised roof structures to the LoD1 and in addition, boundary surfaces can be represented as thematic features (e.g.*GroundSurface*, *WallSurface*, *RoofSurface*). The class LoD3 extends LoD2 by openings (windows, doors), detailed roof structures (dormers, chimneys) and detailed façade structures, while the class LoD4 extends LoD3 with interior structures. However, LoD3 and LOD4 are not explored by the present study. The positional and height accuracy is increasing with the LoD class. Figure 1.2 illustrates the five LoD classes, and Figure 2.1 summarises the LoD requirements by CityGML.

According to Biljecki et al. [2016b] there are a few issues with the LoD specification as they are described by CityGML:

- In the geo-domain, the LoD concept is already used prior 3D model acquisition to define the desired quality, while the LoD was invented for computer graphics to describe geometry abstractions;
- The number of primitives cannot be considered as an unambiguous differentiator (as in computer graphics);
- The LoD cannot be ordered in importance or fit-for-purpose, eg. LOD(i+1) > LODi is not true;



	LOD0	LOD1	LOD2
Model scale description	regional, landscape	city, region	city, city districts, projects
Class of accuracy	lowest	low	middle
Absolute 3D point accuracy (position / height)	lower than LOD1	5/5m	2/2m
Generalisation	maximal generalisation	object blocks as generalised features; > 6*6m/3m	objects as generalised features; > 4*4m/2m
Building installations	no	no	yes
Roof structure/representation	yes	flat	differentiated roof structures
Roof overhanging parts	yes	no	yes, if known
CityFurniture	no	important objects	prototypes, generalized objects
SolitaryVegetationObject	no	important objects	prototypes, higher 6m
PlantCover	no	>50*50m	>5*5m

Figure 2.1: The LoD requirements by CityGML 2.0 [Gröger et al., 2012]

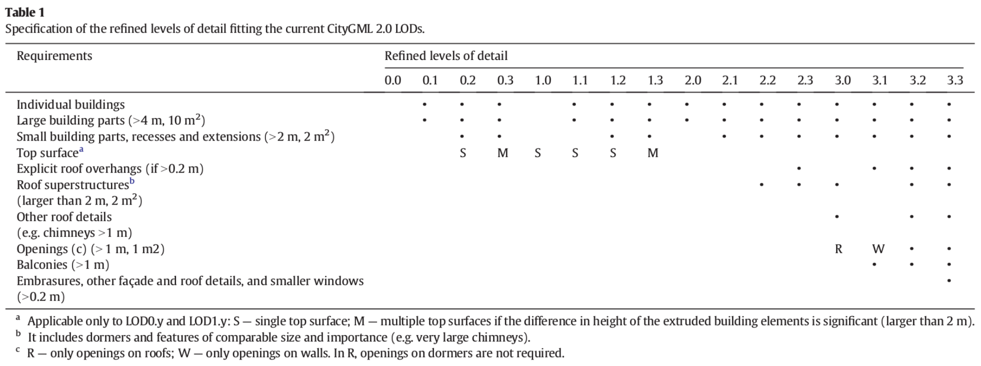


Table 1
Specification of the refined levels of detail fitting the current CityGML 2.0 LODs.

Requirements	Refined levels of detail															
	0.0	0.1	0.2	0.3	1.0	1.1	1.2	1.3	2.0	2.1	2.2	2.3	3.0	3.1	3.2	3.3
Individual buildings		•	•	•		•	•	•	•	•	•	•	•	•	•	•
Large building parts (>4 m, 10 m^2)		•	•	•		•	•	•	•	•	•	•	•	•	•	•
Small building parts, recesses and extensions (>2 m, 2 m^2)			•	•			•	•		•	•	•	•	•	•	•
Top surface[a]			S	M	S	S	S	M								
Explicit roof overhangs (if >0.2 m)												•		•	•	•
Roof superstructures[b] (larger than 2 m, 2 m^2)											•	•	•		•	•
Other roof details (e.g. chimneys >1 m)													•		•	•
Openings (c) (>1 m, 1 m2)													R	W	•	•
Balconies (>1 m)														•	•	•
Embrasures, other façade and roof details, and smaller windows (>0.2 m)																•

[a] Applicable only to LOD0.y and LOD1.y: S — single top surface; M — multiple top surfaces if the difference in height of the extruded building elements is significant (larger than 2 m).
[b] It includes dormers and features of comparable size and importance (e.g. very large chimneys).
[c] R — only openings on roofs; W — only openings on walls. In R, openings on dormers are not required.

Figure 2.2: The refined LoD requirements by Biljecki et al. [2016b]

- There is no strict specification in CityGML 2.0 on how detailed each LoD should be;
- Lack of tools and methods for LoD validation, similarly for example to geometry or schema validation (e.g.Wagner and Ledoux [2016]);
- Often 3D city models are created by combining data from different sources [Biljecki et al., 2016b] . CityGML 2.0 LoD specification does not consider the LoD of the combined data, where some parts of the building might be acquired in finer or coarser detail than others. For example an LoD1 building model that has a very detailed footprint, or a data set that was generated by combining multiple data sets, such as the 3D city model of Berlin[1].

Therefore Biljecki et al. [2016b] refined the CityGML 2.0 LoD specification, adding three subcategories to each category from LoD0-LoD3, see Figure 2.2, Figure 1.1. The refined specification does not prescribe the semantic and texture requirements, but only addresses the amount of geometric detail that has to be acquired.

While the CityGML standard [Gröger et al., 2012] is considered to be the authoritative reference on the LoD definition, it leaves room for interpretation which often lead to 3D city models with heterogeneous LoD. The refined LoD specification by

1 http://www.businesslocationcenter.de/berlin3d-downloadportal/?lang=en

Biljecki et al. [2016b] provides a clear distinction between each sixteen class, therefore it is a key reference used in this thesis.

Löwner et al. [2016] discuss the proposed improvements of the LoD concept for the upcoming third version of the CityGML standard. They highlight the need that the LoD concept complies with the requirements of certain application. For example that the combination of a rough LoD1 or LoD2 model of the exterior shell with a detailed interior model would be beneficial and notably cost-effective, because some applications require it (eg. fire fighting, emergency operation, indoor navigation [Boeters et al., 2015]).

2.2 THE VALIDITY AND LOD OF A 3D CITY MODEL

Wagner and Ledoux [2016] conducted an extensive experiment on the validity of 3D city models, and among others, it developed guidelines for the definition of data quality and quality checking process of CityGML data. They defined five different themes for validation:

- XML-Schema validation;
- Conformance based on formal and non-formal requirements in the CityGML standard document;
- Referential integrity (within a CityGML document as well as to external data sources);
- Geometry; and
- Semantics / attribute constraints such as deviation of attribute "measured height" and the height of the LoD2 geometry.

Even though the LoD is one of the most important aspects of 3D city models [Biljecki et al., 2016b], it was not part of the study. It is important to mention that the validation of geometry and validation of geometrical LoD are two distinct processes. The former is described by Wagner and Ledoux [2016], while the latter is one of the subjects of this thesis.

Accuracy measures the difference between reality and a representation of reality. Biljecki et al. [2018] argues that the *accuracy* and the *LoD* of a 3D city model should be clearly distinguished. However, it is not immediately clear how to separate the accuracy from LoD of a city model. This is, because 3D city models are primarily intended for representing real-world scenarios, therefore the CityGML data model is driven by the characteristics of real-world objects. Consequently many of the LoD requirements relate to the true size of objects. Note that the present work studies only the geometric aspects of 3D city models, therefore other aspects, such as temporal or thematic, are ignored.

In order to validate the geometric LoD of a city model, two distinct questions need to be answered, related to *accuracy* and *LoD*:

1. What is the amount of geometric detail in the building model, expressed in LoD?
2. At the given LoD, does the geometry of the building model correspond to that of its real-world counterpart (considering the accuracy requirements)? For example, does the roof shape of the LoD2 model follow the geometry of the real roof? More specifically, an LoD2 model with a flat top surface, representing a building with a gabled roof is invalid.

It is relatively simple to establish the validity of a model in the simplified case above, but there are several ambiguous cases in data sets. Although, the CityGML

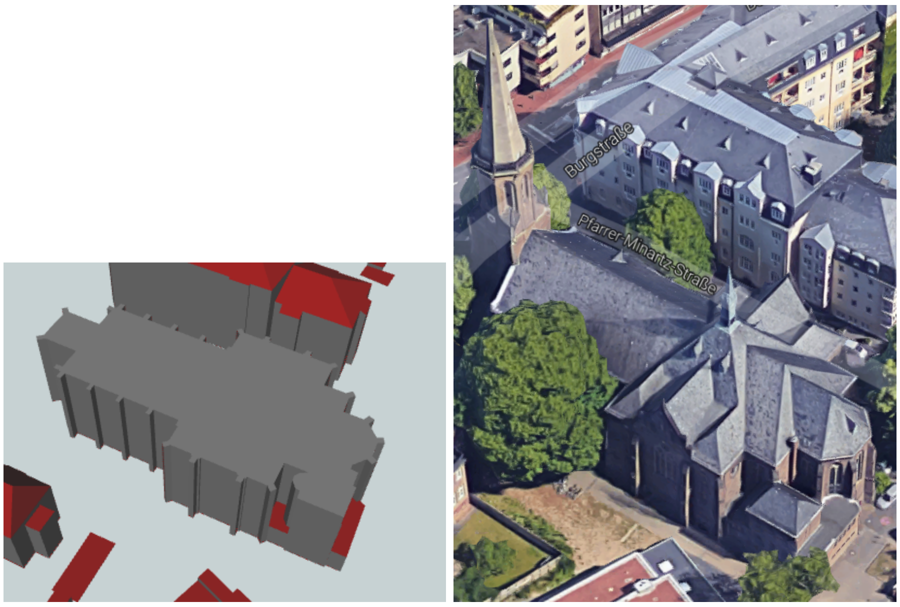


Figure 2.3: LoD1 model of a church and an image of the real building. Source: Google Earth for the aerial image; CityGML data set of Bonn Bad Godesberg, Germany

standard proposes certain absolute 3D point accuracy requirements (e.g.5/5m position/height in LoD1), examples can be found where this falls short in determining the validity of the LoD.

In case of Figure 2.3, the LoD1 model does not require the representation of the roof structure, and the model was probably obtained by extruding the footprint to the height of the eves. Even though the height accuracy of the model is probably less than 5m (as proposed by CityGML) due to the distance between the top of the tower and the top surface of the model, the model can be still considered valid.

In case of Figure 2.4, the 3D point accuracy of the roof might be under 2m (as proposed by CityGML), but the roof shape is not correct, therefore the model is invalid.

The case of Figure 2.5 is more ambiguous. Due to the dormers, it is not immediately clear where does the wall stop and the roof begin. Only a closer inspection of the Google Streetview reveals the true height of the walls and the angle of the roofs. In this case the validity of the model depends on the interpretation of the building and model's roof structure.

The three cases above illustrate that the validity of the LoD (or accuracy) can be only determined with respect to the LoD class in question (also see Figure 2.6). Although the accuracy and LoD of a model is assessed in separate processes, both are required to assess whether a model conforms to the stated LoD.

2.3 A BRIEF OVERVIEW ON MACHINE LEARNING

Machine learning is a complex process with many variables and combinations. As there are no hard rules or recipes for solving a given problem, the machine learning process is iterative and relies on data exploration, modeling and experimentation. A generalised overview of the process is provided by 2.7.

The present section is an excerpt from Bishop [2006], Theodoridis and Koutroumbas [2009], Han and Kamber [2011], Pedregosa et al. [2011], Duin and Pekalska [2015], Müller [2016].

DATA PREPROCESSING in case of spatial data usually consists of format conversion, geometry validation and reparation (e.g.closing rings and shells, harmonis-

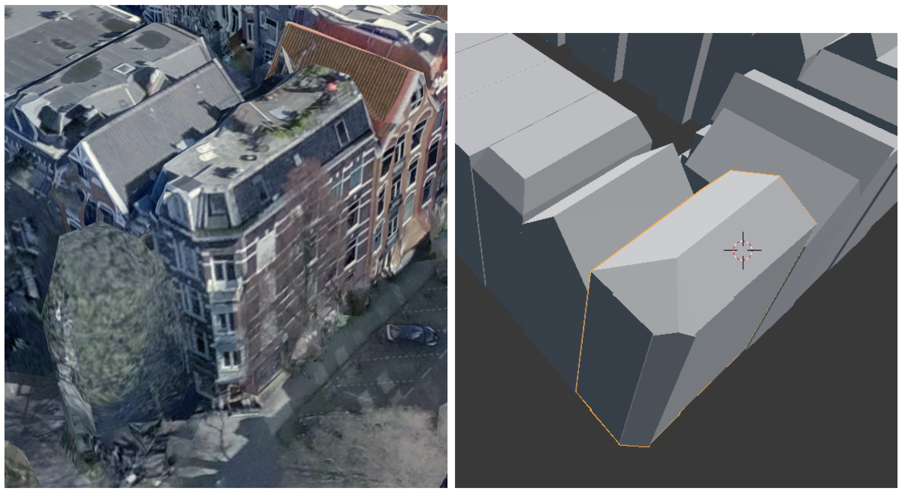


Figure 2.4: LoD2 model of a house and an image of the real building. Source: Google Earth for the aerial image; CityGML data set of Amsterdam, The Netherlands, by VirtualCitySystems

ing face orientation), extraction of geometry parts (e.g.vertices, edges or faces) and transforming the data into a structure that is easy to process in the following steps.

FEATURE GENERATION or representation is the process of making real world objects numerically comparable. In GIS *features* represent real-world entities of interest for geographic information (e.g.buildings, windows) [Kresse and Danko, 2012] and *feature, object* are often used interchangeably. In the field of machine learning a *feature* is any measured property of an *object* and a collection of features (*feature vector*) is used to uniquely identify an object Theodoridis and Koutroumbas [2009]. To resolve this conflict in this work the digital representation of real-world entities are referred to as *objects*, while their measured properties as *features*. In the present work, the feature generation step deals with computing several numerical descriptors of the shape of a 3D building model (e.g.the number of shape characterizing points).

Duin and Pekalska [2015] describes that in a machine learning system the first source of knowledge is the expertise of the analyst or application expert, the second source of knowledge is the *design set*. The design set is a set of objects drawn from the same source as the future objects to be classified. In case of the present work, this means a 3D city model that is representative of the city models in which the LoD inference method will be applied. All of the objects in the design set should have a correct class label assigned. Thus in the city model, each building needs a valid and accurate LoD label. Then the design set is split into at least the *training set* and *test set*. Where the training set is used for training the classifier, the test set is used for evaluating it.

FEATURE EXPLORATION is the step responsible for getting familiar with the statistical properties of the generated features, as the performance of a classifier heavily depends on them. Common techniques include scatter and box plots, first order statistics [Duin and Pekalska, 2015], [Han and Kamber, 2011]. It is important to identify and correct null, or other abnormal values that might bias the classification [Müller, 2016].

FEATURE PREPROCESSING is the step of scaling and/or encoding the features so that a classifier can use them. Some classifiers are more sensitive to the scale of features than others. For example many elements used in a learning algorithm (e.g.Support Vector Machines) assume that all features have a mean around 0 and

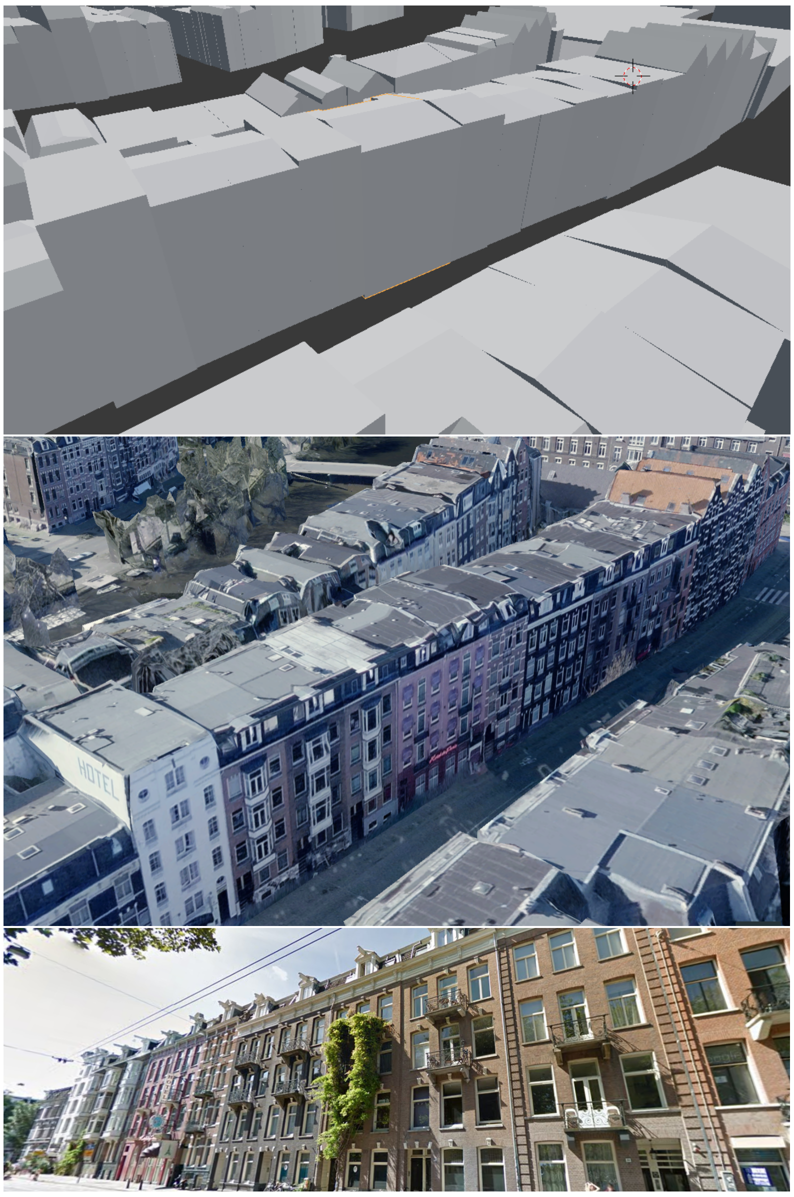


Figure 2.5: LoD2 model of a house (on the corner) and an image of the real building. Source: Google Earth for the aerial image; Google StreetView of the image from the street; CityGML data set of Amsterdam, The Netherlands, by VirtualCitySystems

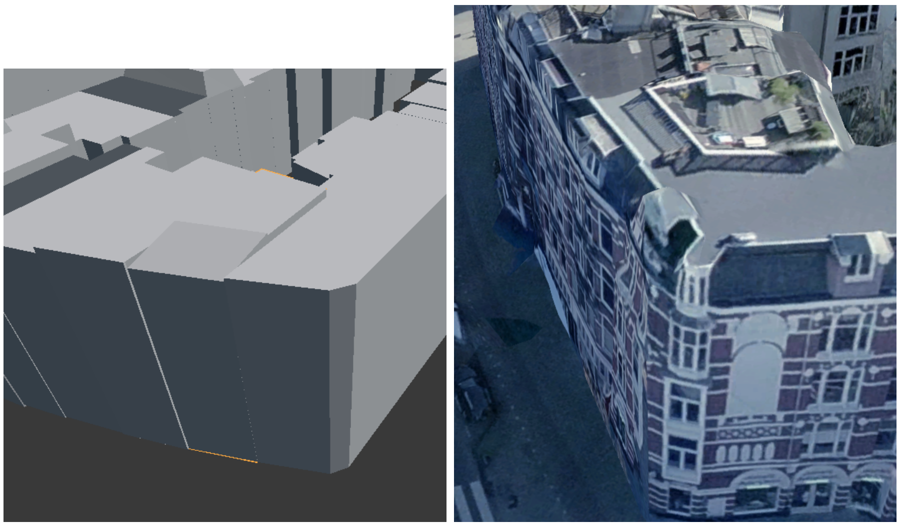


Figure 2.6: LoD2 model of a house on the corner where the roof reconstruction failed. It would be a valid LoD1 model. And an image of the real building. Source: Google Earth for the aerial image; CityGML data set of Amsterdam, The Netherlands, by VirtualCitySystems

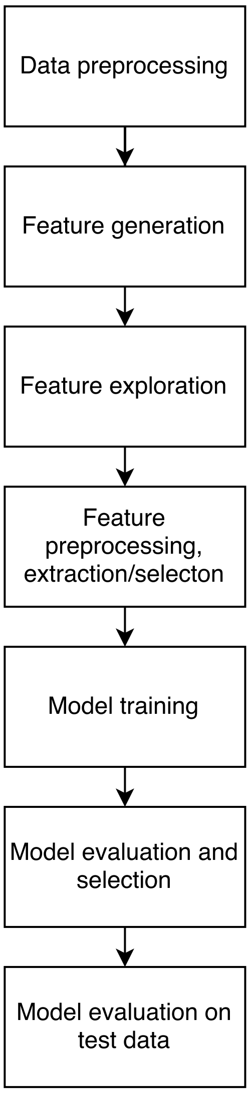

Data preprocessing
Feature generation
Feature exploration
Feature preprocessing, extraction/selecton
Model training
Model evaluation and selection
Model evaluation on test data
Figure 2.7: A generalised overview of the machine learning process. After Theodoridis and Koutroumbas [2009] and Müller [2016]

Table 2.1: Overview of selected classifiers in scikit-learn [Pedregosa et al., 2011]

Classifier	Multiclass	Standardizing necessary	Linear
LR	OneVsRest(solver=liblinear)	No	Yes
LDA	Yes	Yes	Yes
kNN	Yes	Yes	No
DTree	needs balanced classes	No	No
GaussianNB	Yes	No	No
SVM	OneVsRest(kernel=linear)	Yes	Yes

have variance in the same order [Theodoridis and Koutroumbas, 2009]. If a feature has a variance that is orders of magnitude larger than others, it might dominate the objective function and make the estimator unable to learn from other features correctly as expected [Pedregosa et al., 2011]. Centring a feature to 0 mean and scaling to unit variance is called *standardisation*. Many classifiers (e.g.Logistic Regression) can only process numerical data, therefore categorical variables need to be encoded. One of the most common ways to represent categorical variables is *one-hot-encoding* [Müller, 2016]. This method replaces a categorical variable with one or more new binary features.

If the training set is small in relation to the number of features, the classifier might adapt more to the local noise than to the global class differences (overtraining). To avoid this, dimension reduction procedures are used, such as *feature selection* or *feature extraction* [Duin and Pekalska, 2015]. However in case of the LoD inference, each LoD requirement may need at least one feature in order to unambiguously distinguish between the classes and removing a feature might result in a higher accuracy in the training data, but decrease the reliability in a city model where additional LoD classes are present.

MODEL TRAINING is the step when a classifier algorithm learns from the training set and results into a trained classifier, which is a function of the features [Duin and Pekalska, 2015]. Usually more than one classifier is trained in order to find the ones that perform the best for the given problem. Theodoridis and Koutroumbas [2009] groups classification algorithms into three groups, *classifiers based on Bayes decision theory*, *linear classifiers* and *non-linear classifiers*. Where *linear* and *non-linear* refers to the decision boundary that is used for separating the classes. Table 2.1 gives an overview of some of the most common classifiers described below.

Classifiers that are based on Bayes decision theory exploit the statistical nature of the generated features. They classify unknown objects or patterns in the most probable of the classes. In order to do so, they estimate the underlying distribution that describes the data. Classifiers such as the *Gaussian Naive Bayes (GaussianNB)* and *Linear Discriminant Analysis (LDA)* belong to this group and assume that the data follows a Gaussian distribution. The *GaussianNB* classifier is relatively fast to train, because it learns parameters by looking at each feature individually and collect per-class statistics from each feature. Therefore it is often used in very high-dimensional data. To make a prediction, a data point is compared to the statistics for each of the classes, and the best matching class is predicted [Müller, 2016]. The *LDA* classifier has a linear decision boundary and besides the Gaussian distribution, it assumes that each class share the same covariance matrix. The LDA can also be used for dimensionality reduction, e.g.[Wurm et al., 2016].

Linear classifiers assume that the classes can be separated by planes that are described by linear functions, regardless of the distribution of the data. The advantage of linear classifiers lies in their simplicity and that they are fast to train and predict, and work well on large, sparse data sets. Classifiers such as *Logistic Regression (LR)* and *linear Support Vector Machines (SVM)* belong to this group. Their main parameter is the regularization parameter, called C [Pedregosa et al., 2011]. In case of LR and SVM, a low value for the parameter C will cause the algorithm to try to fit the majority of data points, while a higher value of C will cause the algorithm to try to correctly classify each data point but therefore might not capture the overall layout of the classes well [Müller, 2016].

Non-linear classifiers are fit for problems when the classes are not linearly separable in the feature space. Classifiers such as *Decision Tree* and *k-Nearest Neighbours (kNN)* belong to this group. The *Decision Tree (DTree)* classifier essentially learns a hierarchy of if/else questions that lead to a decision, and they can be unstable as small variations in the data can result in a completely different tree. They are also sensitive to class imbalance. However, they are flexible in accepting data, as they do not require normalization or encoding, and can handle multiple classes [Müller, 2016], [Pedregosa et al., 2011]. The *kNN* algorithm can be used for clustering as well as classification. When used for classification, the data point in question is assigned the majority label among its k nearest neighbours. Where the *nearest* neighbours are determined by some distance metric, such as Euclidean or Minkowski distance. Both the kNN and DTree classifiers are non-parametric, thus the model estimation is based on a non-parametric function of all observation (and not on optimizing a parametric model e.g.normal distribution), therefore they are generally well suited for problems where the decision boundary is very irregular [Duin and Pekalska, 2015], [Pedregosa et al., 2011], [Theodoridis and Koutroumbas, 2009].

Classifiers such as LDA and SVM are binary classifiers, but can be extended to a multinomial classifier using a *one-versus-the-rest* strategy [Pedregosa et al., 2011]. In this way, elements of one group are separated from all other elements belonging to the other groups.

MODEL EVALUATION is the process of selecting the best performing classifiers which will be used in the final classification system. A common method for obtaining reliable classifier accuracy estimates is *cross-validation*. In *k-fold cross-validation* the initial data are randomly partitioned into k mutually exclusive subsets or "folds", each of them approximately equal size. Then training and testing is performed k times. In each iteration, one (different) partition serves as a test set, while the all other partitions are used for training. The accuracy estimate is the overall number of correct classifications from k iterations, divided by the total number of objects in the initial data. Common values for k are 5 or 10 [Duin and Pekalska, 2015]. An alternative method is the *leave-one-out* cross-validation, which is similar to k-fold cross-validation but each fold consist only of a single object. The *leave-one-out* method often provides more reliable estimates on small data sets [Müller, 2016].

The *accuracy* of a classifier in a test data set is the percentage of objects that are correctly classified. *Precision* is a measure of exactness, or what percentage of objects labelled as positive are actually such. While *recall* is a measure of completeness, or what percentage of positive objects are labelled as such. The *F measure (or F_1 score)* is the harmonic mean of precision and recall, thus is gives equal weight to each [Han and Kamber, 2011].

2.4 SHAPE DESCRIPTORS IN COMPUTER GRAPHICS

Shape descriptors are extensively studied in field of computer graphics, as it is illustrated by Zhang and Lu [2004], Yang et al. [2008], Costa et al. [2009] and Kazmi et al. [2013]. The current section is a very brief outlook this field. Shape descriptors are measures of a shape, comprising its geometry or topology into set of numerical values or graph like structure [Kazmi et al., 2013]. Thus in machine learning terminology, these shape measures serve as features in shape classification, and good features are essential for a good classification outcome.

According to Yang et al. [2008] and Kazmi et al. [2013], effective shape descriptors have the following properties:

Table 2.2: Categorisation shape descriptors by Kazmi et al. [2013]

Dimension	Category	Name
2D	Contour Based	Fourier Descriptors (FD)
		Wavelet Descriptors (WD)
		Curvature Scale Space (CSS)
		Shape Context (SCD)
	Region Based	Zernike Moment Descriptors (ZMD)
		Scale Invariant Feature Transform (SIFT)
		Angular Radial Transform (ART)
	Hybrid	FD + ART
		FD + ZMD
3D	View Based	Adaptive Views Clustering
		Compact Multi-View Descriptor
		LightField Descriptor (LFD)
	Histogram Based	Shape Spectrum
		Generalized shape distributions
		Bag-of-Features (BoF)
	Transform Based	Spherical Harmonics Descriptor
		PCA Spherical Harmonics Transform
		Spherical Trace Transform
	Graph Based	Skeletal Graph Based
		Reeb Graph Based
	Hybrid 3D Descriptors	CMVD + STT
		Depth-Buffer + Silhouette + REXT
		SIFT + Bag of Features
		Depth-Buffer + Spherical Harmonics

- Discriminative accuracy and completeness: To represent the information content of in the shape and accurately distinguish one shape from another based on subtle differences;
- Transformation (translation, scaling, and rotation) invariance: Also known as pose normalization;
- Robustness against model degeneracies / roughness;
- Uniqueness: Each shape descriptor must be uniquely coupled with a unique shape;
- Performance and memory efficient, thus the computation of distance between descriptors should be simple;
- Partial matching: robust against incomplete shapes;
- Insensitive to noise: Small changes in the shape to lead to small changes in the shape descriptor;

For the purpose of describing building models, the categorisation of shape descriptors by Kazmi et al. [2013] is fitting, see Table 2.2. But Zhang and Lu [2004], Yang et al. [2008] provide different categories, while Costa et al. [2009] also describes several general descriptors such as *diameter, major and minor axes, mean distance to boundary*.

The approach of describing 3D shapes by the sampled distribution of simple shape functions (e.g.distance between two random points on a surface) [Osada et al., 2001], or by means of 3D shape histograms [Ankerst et al., 1999], seems particularly fitting for setups where the city model is stored in a database (e.g.by 3DCityDB

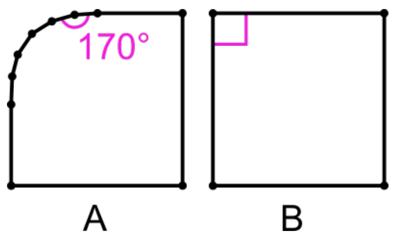


Figure 2.8: Vertices along the curve do not count as SCP, thus (A) has 3 SCPs. The rectangle (B) has 4 SCPs (B)

[2]). A similar setup could make it possible to efficiently retrieve and/or analyse the geometry of individual buildings.

2.5 SHAPE DESCRIPTORS AND CLASSIFICATION IN GEOGRAPHIC INFORMATION

In section 2.4 provided a brief overview on shape measures in the field of computer graphics. Researchers of geographic information developed their own, application specific shape measures for analysing and classifying geographic objects. Sections 2.5.1 and 2.5.2 review those 2D and 3D measures that to the extent of my knowledge relevant for describing building models. While Section 2.5.3 describes two cases such measures were applied for building shape classification.

2.5.1 Geometric complexity of 2D objects in geographic information

Moser et al. [2002] give an overview of the various polygon shape descriptors. They describe the *Number of shape characterising points (NSCP)*. It is obtained by checking which vertices in a polygon boundary have a turning angle less than 160 degrees. Vertices with an angle above 160 and until 180 degrees are not considered as *shape characterising points (SCP)*. Consequently, the higher the NSCP, the more complex the polygon boundary. However, one weakness of the method is that the angle threshold need to be set manually. When curved objects are modelled with straight line segments, the segments along the curve can have a low turning angle. These point would not be considered as SCP, see 2.8.

Touya and Brando-Escobar [2013] describes a method for assessing the LoD of geographical features. Their method includes a set of criteria from which some are relevant for my work:

- Vertex density (resolution): number of vertices compared to feature length described with $vertex\_density\_value = (1 - vertexDensity)$
- Median edge length (resolution)
- Shortest edge (granularity): the length of the shortest edge
- Size (granularity): area of the feature

They also analyse the spatial relations between features (e.g. road crossing a lake) to fine tune and filter the results. This idea can be easily applied to buildings as well, for example a window must be part of a wall or roof but cannot be part of a floor, or roofs must be above floors.

Touya and Reimer [2015] compares and combines the methods of Touya and Brando-Escobar [2013] and Reimer et al. [2014] for inferring the scale or LoD of

2 https://www.3dcitydb.org

OpenStreetMap objects, with the conclusion that in case of buildings the combination of methods does not result in significant increase in classification accuracy. The empirical method of Reimer et al. [2014] requires a manual determination of a constant, furthermore Touya and Reimer [2015] could not detect significant numerical difference in vertex frequency in polygons where curvy rivers were part of the polygons and when not. In case of buildings, the higher number of vertex per object is likely to indicate a higher LoD, due to the additional feature parts (chimney, window etc.).

2.5.2 Geometric complexity of 3D objects in geographic information

Wong and Ellul [2016] evaluates simple geometry metrics that are designed to give an indication of the geometric detail of the data set as a whole and not each individual building. To the extent of my knowledge, this is the only work in this topic. These metrics are influenced by the architecture of the buildings. Their findings are that:

- The mean number of vertices / edges / faces per building reveal the same pattern in the data sets.
- The mean number of vertices per face measures the efficiency and detail of the model, where the lower the ratio the less efficient the model.
- In case of the minimum footprint area and minimum feature length often very small values are found eg. $< 0.001m$ or NULL therefore these measures cannot be used directly but their frequency distribution is more suitable. The authors recommend using the minimum dimensions of the buildings if the data is relatively clean, and the frequency distribution of the minimum areas if the data is noisy. The frequency distribution could also provide insight into inconsistencies or errors in the data set.

2.5.3 Geometry-based classification of building types

When using shape-based features, building type classification is similar to LoD inference if we consider the various building types as LoD subclasses. Both Henn et al. [2012] and Wurm et al. [2016] worked out a method to classify building types in LoD1 3D city models.

Henn et al. [2012] uses machine learning to classify seven building types in 3D city models, see Figure A.1. Their approach is similar to that of the present work, as in they use statistical learning methods to categorise 3D building models based on their geometry. Henn et al. [2012] found out that incorporating the spatial context of the building improves the classification accuracy greatly. Therefore they compute the distance between neighbouring buildings, distance from infrastructures. However, such metrics are only possible when infrastructural buildings are also included in the city model, which might no be the case when only parts of a city is analysed.

Henn et al. [2012] uses a training data set of 1227 building and also tests a Decision Tree and Naive Bayes classifier, however they achieve the best overall accuracy of 90.79% with an SVM with linear kernel. The accuracy was determined with 10-fold cross-validation. In case of a building type (villas) that is highly similar in LoD1 to another building type (detached buildings) they achieved only 54.84% recall.

Additionally, Henn et al. [2012] developed an efficient, semi-automatic method to collect training data for the Support Vector Machine classifier that they used. In this method they used outlier detection and clustering to reliable differentiate between building types. Their method is based on the cluster assumption of semi-supervised learning which states that, if objects are in the same cluster, they will likely be of the same class [Chapelle et al., 2006]. The Local Outlier Factor (LOF) compares local densities and it rates how isolated a point in relation to its nearest neighbours. This is an advantageous property as the underlying distribution of the clusters is often not known.

Similarly, Wurm et al. [2016] used Linear Discriminant Analysis to distinguish between five different building types in LoD1 (see Figure A.2) and evaluate the discriminatory power of the computed features. They focused solely on shape-based features and computed 26 1D, 2D and 3D features, while ignoring the spatial-relational features that for instance Henn et al. [2012] used. Using a total of 9557 building from in two cities (Berlin, Munich), they discovered that training sample sizes above 10% do not significantly improve the classification accuracy. Furthermore, they tested the generalizability of trained classification functions for transfer between geographical regions by training the classifier in one city and applying it in the other, and vice versa. They achieved a kappa index between 0.91-0.94 in their classification.

3 RESEARCH METHOD

The goal of this thesis is to research a method for automatic LoD inference in non-semantic 3D city models, and use the inferred LoD classes to evaluate whether a 3D city model conforms the stated LoD. In order to determine the LoD, the shape-complexity of the model need to be measured and assigned to an LoD classes, according to the LoD requirements. However, a simple simple shape does not necessarily mean coarse LoD. Buildings, building attachments that have a simple shape (e.g.garages, tower blocks) can be modelled with the same geometry in LoD1 and LoD2, see 3.2. Therefore in case of a detailed city model (e.g.LoD2) if an object has coarse shape that would indicate LoD1, it is necessary to compare the object to ground truth in order to evaluate whether the model is indeed LoD1 or the building has a simple shape.

Consequently, the method comprise of two steps. In the first step, the shape complexity of each object is measured and the object's LoD class is inferred from the measured values. In the second step, the objects are compared to a reference data set, and the comparison results are included in the LoD classification, in order to distinguish between simple buildings in LoD2 and complex buildings in LoD1. Figure **??** gives an overview of the process.

3.1 STEP 1 — LOD INFERENCE WITHOUT REFERENCE DATA

In its core, the framework utilizes statistical learning methods to distinguish between buildings of shape complexity and infer the LoD of the city model. Therefore the proposed framework follows that of common with machine learning. Broadly this is an iteration of feature generation, application of learning methods, model evaluation.

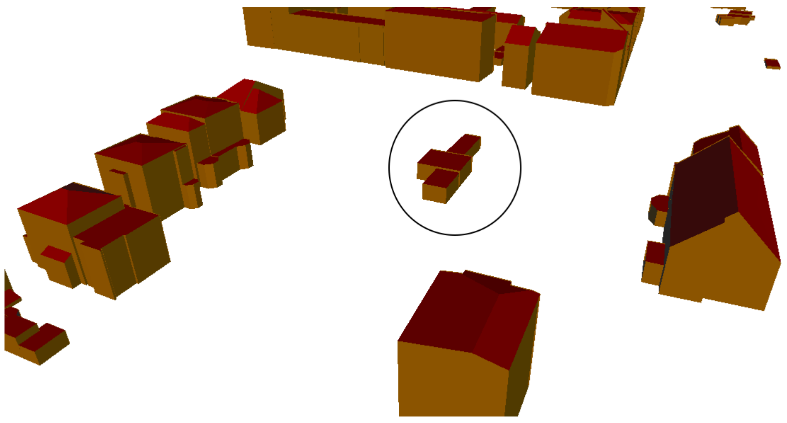


Figure 3.1: A building with the same geometry in LoD1 and LoD2 (garage in the center).

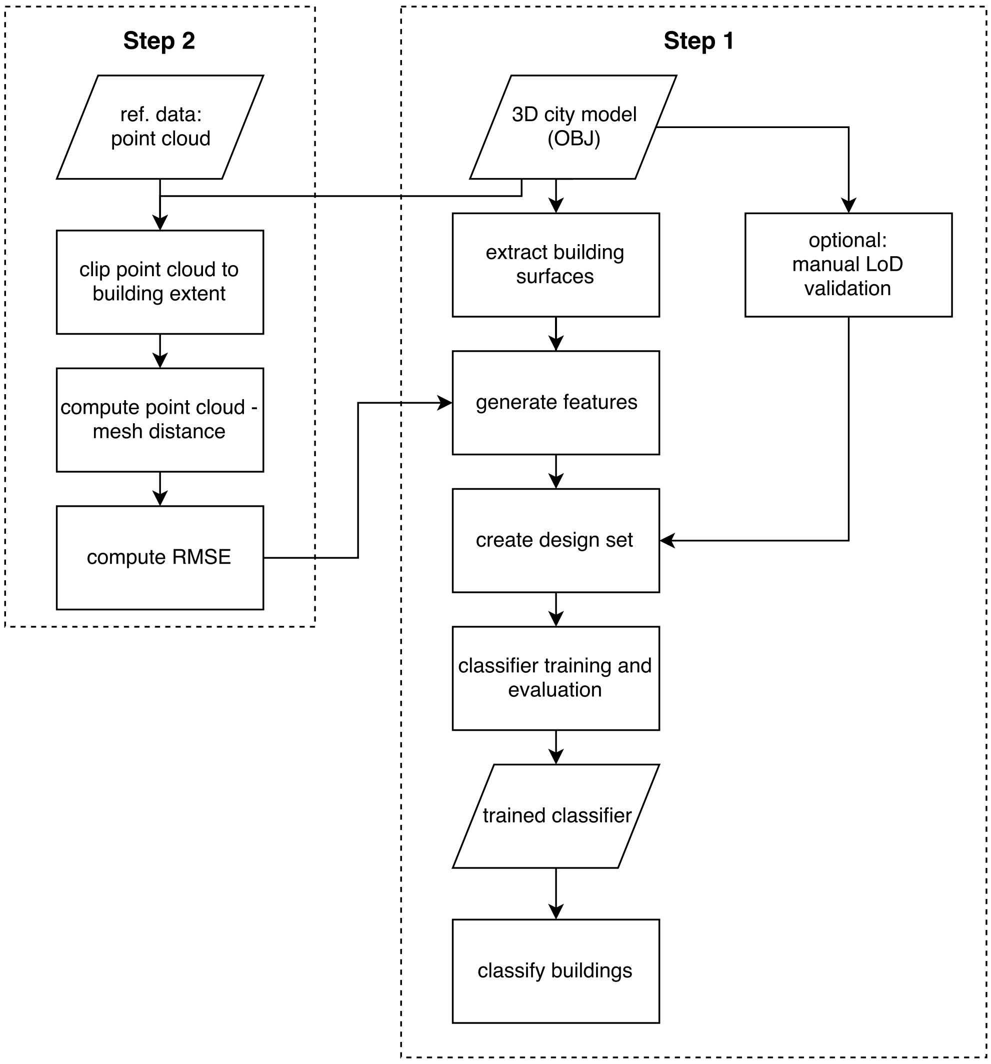

Step 2
ref. data: point cloud
clip point cloud to building extent
compute point cloud - mesh distance
compute RMSE
Step 1
3D city model (OBJ)
extract building surfaces
optional: manual LoD validation
generate features
create design set
classifier training and evaluation
trained classifier
classify buildings

Figure 3.2: The process of LoD inference

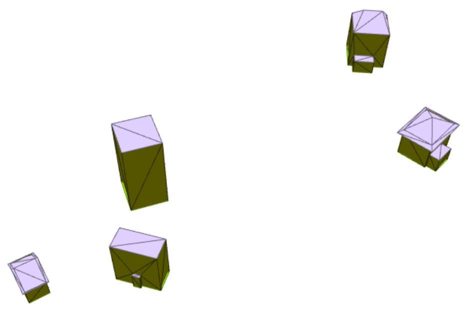


Figure 3.3: Roof (purple) and wall (green) triangles of models

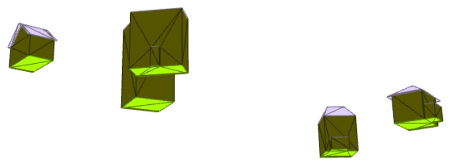


Figure 3.4: The footprint triangles (bright green) of models

3.1.1 Building surface extraction

The data sets used are converted from CityGML to OBJ format, which does not store semantic information. In this format each object is represented by its boundary surface, a closed mesh of triangles. This type of representation is referred to as Boundary Representation (BRep) [Mäntylä, 1987]. There is no distinction in which triangles represent which surface in the building. Therefore the triangles that make up the roof, walls and footprint need to be extracted from the model, see 3.3, 3.4.

Roof, wall and footprint triangles are identified according to the rules below. Given that horizontal triangle has a slope of 0°, a vertical triangle has a slope of 90°. In order to account for irregularities in the surfaces, a threshold of 5° is used when differentiating between horizontal, vertical and sloped triangles, and a threshold of 10 cm is used when differentiating between footprint and non-footprint triangles. The presented method assumes that the surface normals are pointing outwards from the model.

```
slope = slope of a triangle
z_mean = mean z coordinate of a triangle

for each object:
     z_mean_min <- min(mean z coordinate of each triangle)
    for each triangle:
        if slope < 5 and (z_mean < z_mean_min + 10cm):
            footprint_triangles <- triangle
        else if slope < 85 and (z_mean > z_mean_min + 10cm):
            roof_triangles <- triangle
        else:
            wall_triangles <- triangle
```

Table 3.1: Summary of the computed features and their relation to the LoD requirements

Geometry	Feature	Related LoD requirement	Relevant LoD
2D footprint	Number of Shape Characterising Points (NSCP)	none	all
	Shape Characterising Lengths (SCL)	Size of building parts	≥ 0.1
	Footprint Area	Size of building parts	≥ 0.1
	Building Part Footprint Area	Size of building parts	≥ 0.1
3D solid	Building Volume	none	all
3D surface	Roof Type	Roof representation	≥ 1
	Median Roof Gap	Top surface (Single / Multi)	0.2-1.3
	Roof Overhangs	Explicit roof overhangs (if 0.2m)	≥ 2.3
	Footprint-Roof Triangle Ratio	Roof superstructures	≥ 2.2
	Walls	Presence of walls	0
3D solid, Point Cloud	RMSE of PC-Model distance	(LoD validity)	all

3.1.2 Describing building geometry

Sections 2.4, 2.5 illustrate that there are numerous possible measures to describe the geometry of an object and that the appropriate descriptor depends on their application. Although some of the geometry descriptors in this section were influenced by Moser et al. [2002], Wong and Ellul [2016], Henn et al. [2012] and Wurm et al. [2016], one of the primary considerations was to closely relate the descriptors to the LoD requirements by Gröger et al. [2012], Biljecki et al. [2016b]. Table 3.1 summarises the descriptors and their relation to the LoD.

NUMBER OF SHAPE CHARACTERISING POINTS Moser et al. [2002] describes the Number of Shape Characterizing Points (NSCP). The NSCP is obtained by checking which vertices in a polygon boundary have an inner-angle less than 160 degrees. Vertices with an angle above 160 and until 180 degrees are not considered characteristic for the shape. The higher the NSCP, the more complex the polygon boundary, see **??**. Theodoridis and Koutroumbas [2009] also report this measure as so-called key points. However, in round object or object with curved parts this measure can fall short. Arcs are often discretised by a number of straight line segments, and depending on the amount of segments representing the arc, the vertices can have a large inner angle. In this case none of the vertices along the arc would count as Shape Characterizing Points, see Figure 2.8. The NSCP is determined from the total footprint of the object, thus including extensions, attachments.

SHAPE CHARACTERISING LENGTHS The Shape Characterising Lengths (SCL) are the distances between the Shape Characterising Points, see Figure 3.5. Therefore the SCL exclude collinear vertices and vertices with a low turning angle. As both Gröger et al. [2012] and Biljecki et al. [2016b] define the object generalisation requirements in terms of minimal object length and/or area, SCL is interpreted in a sense that no edge in the footprint can be shorter than 4m or 2m respectively.

However, as Biljecki et al. [2016b] states, *"The minimum size can be expressed as the minimum length and/or width of an object, and/or the minimum footprint area."*. Thus there is no guarantee that an object adheres to the minimum length requirement in its level of detail. Possibly, an edge is shorter than the stated minimal length, but the area of the building part is larger than the required minimal area.

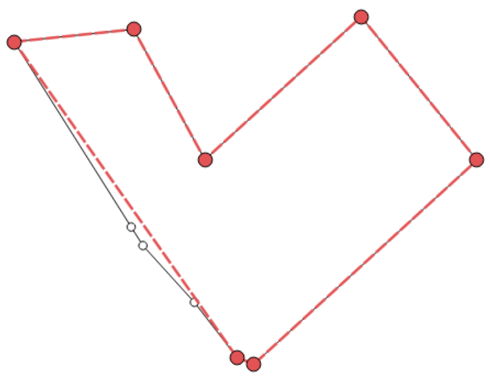


Figure 3.5: Shape Characterising Points (red circle), Shape Characterising Lengths (red dashed line) and non-characteristic points (white circle).

As Wong and Ellul [2016] recommend for edge lengths, the SCL are not used directly for classification due to potential outliers (very short edges) that would skew the values. Instead, the frequency distribution of the SCL is computed, with bins according to the LoD requirements. As an additional feature, the minimal SCL is used.

The SCL are grouped into six bins and the frequency of the bin is computed. It is necessary to use the frequency distribution instead of simple counts, because the the counts change according to the total number of edges in the footprint.

The bins are:

```
>6m
6-4m
4-2m
2-1m
1-0.2m
0.2m>
```

Frequency of elements in each bin is computed as:

$$frequency\_of\_bin = \frac{count\_edges\_in\_the\_bin}{count\_total\_edges}$$

FOOTPRINT AREA Similarly to the SCL, the footprint area relates directly to the generalisation requirements of the LoD specifications. Gröger et al. [2012] prescribes the minimal size of the object that should be considered in each LoD. For example in LoD1, only those objects should be represented that are larger than $6 * 6m$ in the footprint and $3m$ in the height. The footprint area values are not used directly, but they are categorised in four categories according to Gröger et al. [2012]:

```
1. <- area > 36m^2
2. <- area > 16m^2
3. <- area > 4m^2
4. <- area <= 4m^2
```

BUILDING PART FOOTPRINT AREA Biljecki et al. [2016b] defines building parts as objects that are part of the building, but their geometry is perceptually distinguishable. Thus this work interprets building parts as objects that have their own, or separate roof structure, see Figure 3.6. Following, a building always consists of at least one building part (the main part). As a building can have arbitrary many parts, it is only necessary to measure the footprint area of the smallest part, because the extent of the minimal area is prescribed by the LoD requirements.

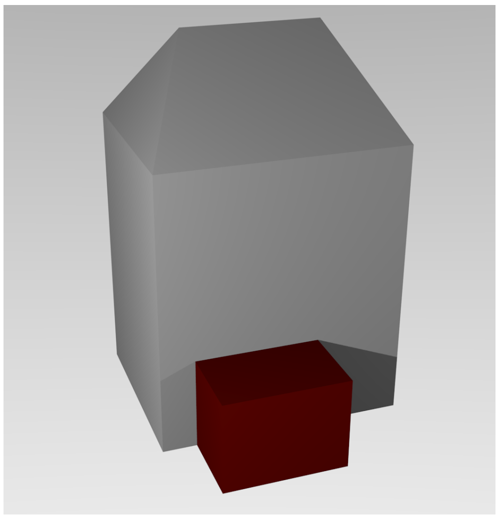


Figure 3.6: An example of main building part (grey) and attached building part (red) with a separate roof structure.

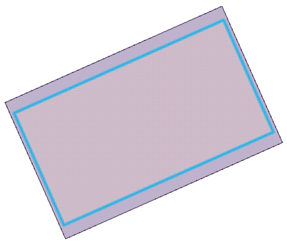


Figure 3.7: Footprint polygon (blue outline), roof polygon (black outline). Each part of the building has an overhanging roof.

Building part footprint area is computed as the area of the roof's projection on the ground. Therefore in case the model has explicit roof overhangs, the computed building part area is slightly larger than the actual building part footprint area.

BUILDING VOLUME The volume of the BRep is computed to give an indication of the size of the object. The volume is computed using FME's *VolumeCalculator* transformer. This measure does not relate directly to any LoD requirement.

ROOF OVERHANGS According to Biljecki et al. [2016b], models in LoD2.3, 3.1 and above can have explicit roof overhangs. This feature indicates the presence of roof overhangs.

The shortest distance between the projected polygon of the roof and the footprint polygon is measured. It works in situations where each part of the building has an overhanging roof (see Figure 3.7). But if a part of the building does not have overhanging roof, the minimum distance between the two polygons is zero (see Figure 3.8).

Therefore an area-based comparison is more robust. In order to make it invariant to the size of the object, the footprint area is divided by the roof area. If the ratio is lower than 1 (or lower than 0.999 exactly, to account for floating point errors), there is an overhanging roof.

Then the feature is encoded as:

```
0 - no overhang
1 - overhang
```

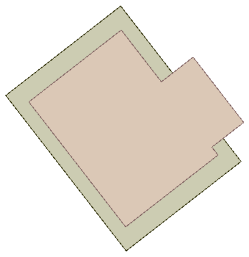


Figure 3.8: Footprint polygon (brown) and the projection of the roof (green). If a part of the building does not have overhanging roof, the minimum distance between the two polygons is zero

FOOTPRINT-ROOF TRIANGLE RATIO Models of LoD0.2, 1.0-1.2 will typically have the same amount of triangles in their roof surface as in their footprint. If the roof has multiple levels or has extensions, the amount of triangles in the roof will be higher than in the footprint. Therefore the *footprint-roof triangle ratio*, combined with the *roof type*, can indicate complex roof structures, see Figure 3.9.

The footprint-roof triangle ratio is computed as below, and therefore a ratio lower than 1 indicates a complex roof structure.

$$footprint\_roof\_ratio = \frac{number\_of\_triangles\_in\_footprint}{number\_of\_triangles\_in\_roof}$$

MEDIAN ROOF GAP *Include pictures of the referenced building models*

According to Biljecki et al. [2016b], models of LoD0.3, LoD1.3, LoD2.0 and above, can represent multiple levels in their roof structure. Therefore the *median roof gap* indicates whether the roof has multiple levels.

The mean z coordinate cannot be used to reliably identify the roof levels, because in case of non-planar roofs, the triangle's mean z varies, depending on the orientation of the triangle. This is illustrated in 3.10.

Also, see that there is some variation in the mean z coordinates of the non-planar roof part of the building "BID_363100012169940" (also see Figure A.7):

```
7.219487 7.236606 7.289978 7.302748 7.310786 7.956606 7.956907
7.958656 8.024201 8.024612 9.379174 9.379585
```

However, it is clear from looking at the shape of the roof that the these triangles belong one roof level, even though the variation. The building "BID_363100012168777" (see Figure A.6) has four distinct roof levels, including the top level. The mean z coordinates support this observation, in the values we can identify these clusters centred around `4.8, 13.6, 15.9, 18.5`.

```
4.81000  4.81000  4.81000  4.81000  4.81000 13.62514 13.62514
13.63000 13.63000 15.96000 15.96000 15.96000 15.96000 15.97527
15.97527 18.53000 18.53000 18.53000
```

A method is needed that clusters the first case in a single cluster, and the second case in four clusters. As clustering is mainly used in multi-dimensional data, a better term for this is to identify breaks in the data. Two of the common methods to identify breaks in 1D data are:

1. natural breaks estimation (e.g.. Jenks natural breaks optimization),
2. finding local minima in kernel density estimates.

Natural breaks estimation requires a fixed number of breaks (clusters) to set, similarly to *k-means* clustering. Because buildings can have various roof levels, this is not a suitable method. Furthermore, the optimization process for finding the

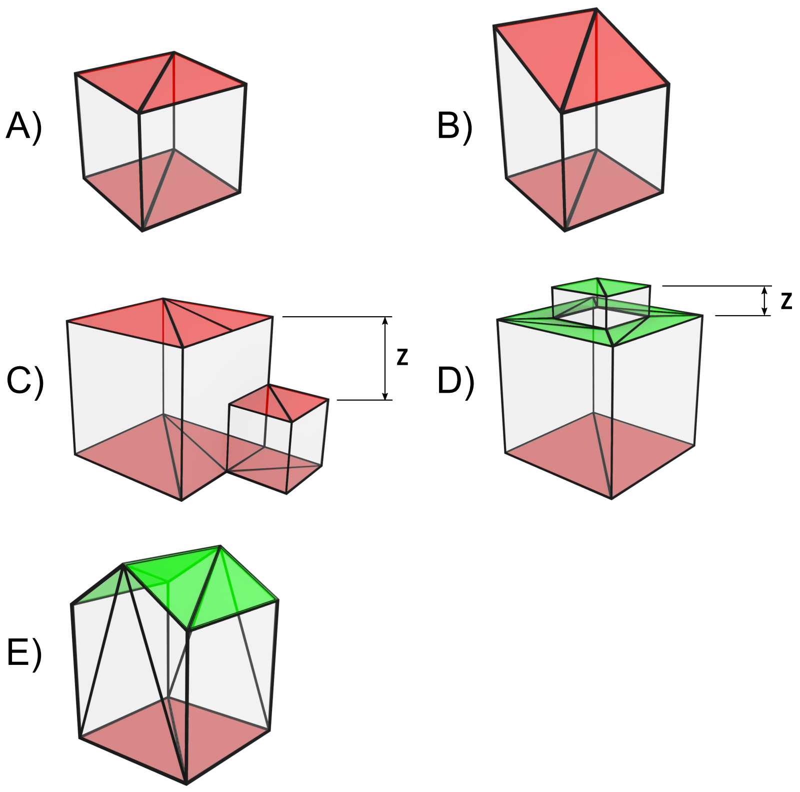


Figure 3.9: Five roof structure types, with equal amount of triangles in footprint and roof (both are red), and with more triangles in the roof (green triangles in roof)

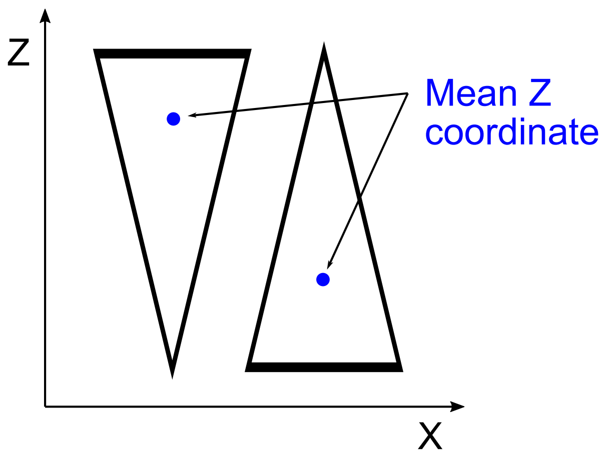


Figure 3.10: The value of the mean of the Z coordinates depends on the orientation of the triangle

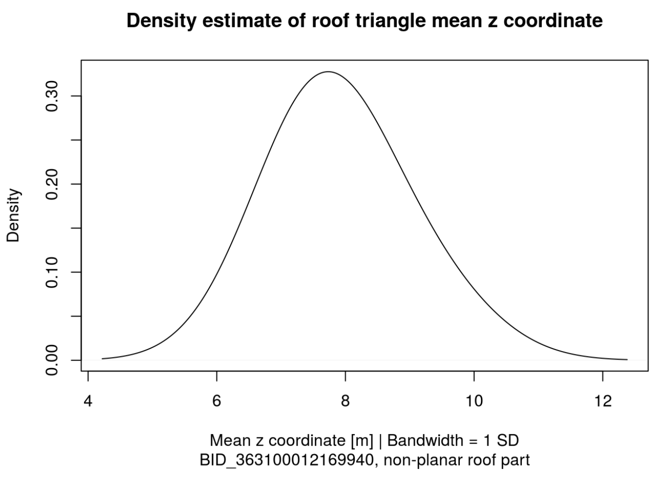


Figure 3.11: Density estimate of roof triangles mean z coordinate. Object BID_363100012169940

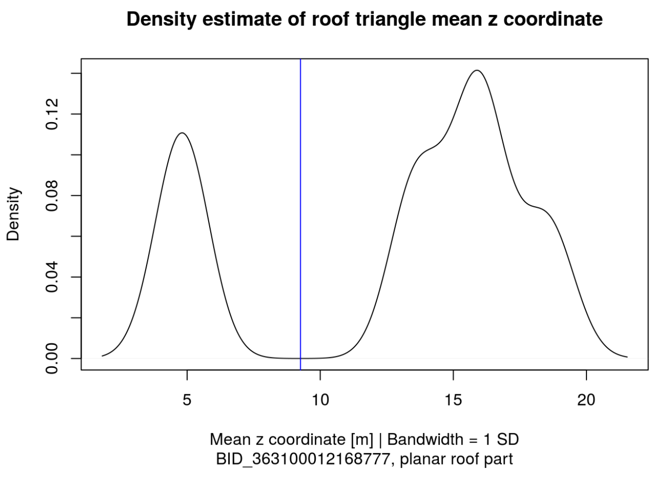


Figure 3.12: Density estimate of roof triangles mean z coordinate. Object BID_363100012168777

"natural number of breaks" with this method would put an unnecessary cost on the process.

Kernel density estimation does not require specifying the number of breaks in the data, thus it seems more suitable. Density estimation with a Gaussian kernel on the previous roofs reveals the modality in the mean z values. Then the local minima of the density estimate indicates the breaks in the mean values (blue lines), thus the presence of multiple roof levels. The single roof level is clearly illustrated by the uni modal density of the mean z values of the non-planar roof part of BID_363100012169940 (see Figure 3.11). Also, the multiple roof levels are displayed by the multiple modes of the density estimates of BID_363100012168777 (see Figure 3.12) and BID_363100012130645 (see Figure 3.13, A.5).

Although there is no local minimum in case of BID_363100012169940, which is correct, based on its roof shape, four local minimum points are expected for BID_363100012168777. But the density estimate only reaches a local minimum at a single location. A solution for this problem would be to adjust the smoothing bandwidth of the kernel.

In the plots above it is set to one standard deviation. In order to get a better fit of the density estimate, the smoothing bandwidth need to be reduced, for example from 1 standard deviation to half. But although this yields a more ac-

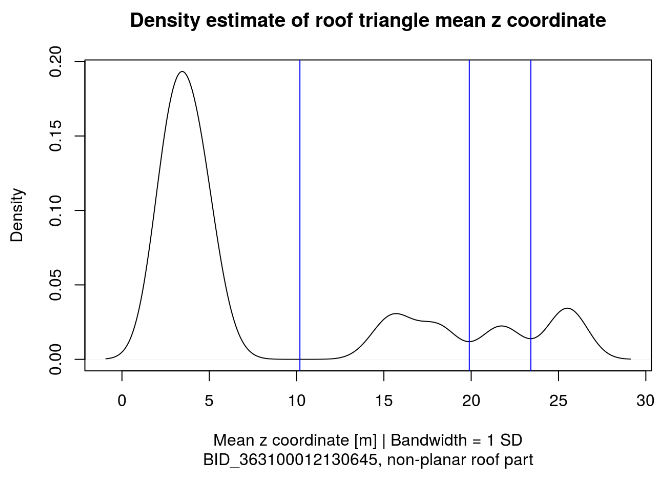


Figure 3.13: Density estimate of roof triangles mean z coordinate. Object BID_363100012130645

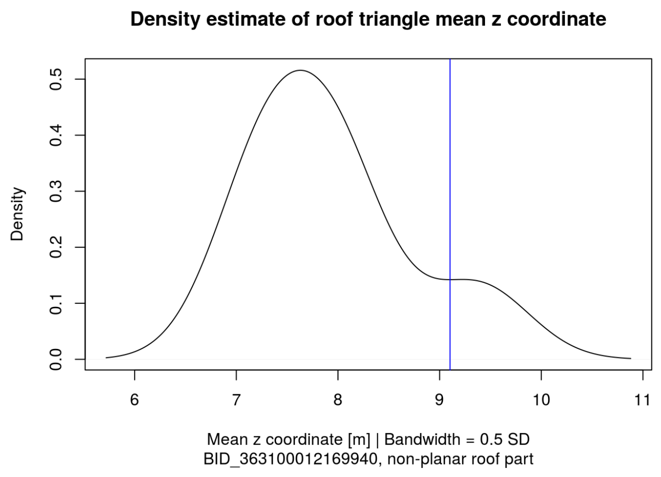


Figure 3.14: Density estimate of roof triangles mean z coordinate, bandwidth 0.5 std. Object BID_363100012169940

curate estimate for BID_363100012168777 (see Figure 3.14), it clearly over-fits for BID_363100012169940 (see Figure 3.15).

A third approach of finding breaks in mean z values is more simplistic, but directly related to the LoD specification. This is a desirable property, because it makes it easy to reason about the results. The LoD specification defines that the minimal vertical "jumps" that need to be identifiable are >`2m` or >`1m`.

Then it suffice to identify the presence of gaps of >`2m` or >`1m` in the sorted mean values. This method might work well in case of planar roofs because mean z values close to each other in one level. However, in non-planar roofs kernel density estimation might yield better results due to the variation in the mean values. For example in case of a high, gabled roof. The gaps are computed as:

```
gap <- 2 meters
    y <- sort(mean z values)
    if length(y) < 2:
        gaps <- 0
    else:
        for each in y:
            if abs(y[i] - y[i+1]) > gap:
```

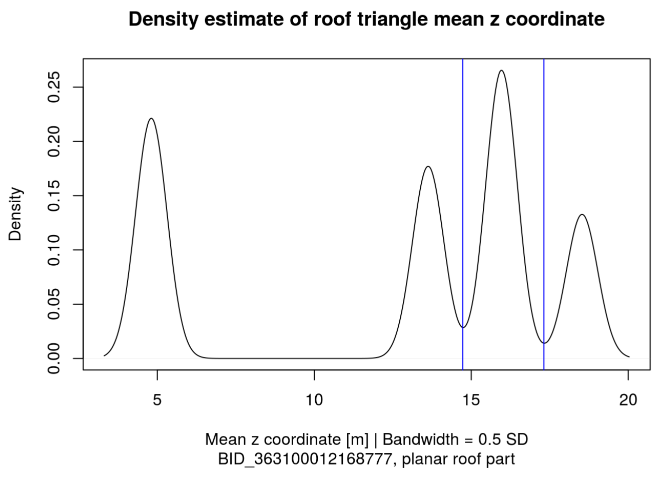


Figure 3.15: Density estimate of roof triangles mean z coordinate, bandwidth 0.5 std. Object BID_363100012168777

```
            gaps <- c(gaps, y[i])
```

The median distance between the roof levels gives an indication of whether the building has a single top surface, or its top surface has multiple levels. Multiple roof levels are allowed in LoD0.3, LoD1.3 and >LoD2.0. In case there is a single roof level, the median distance is 0.

ROOF TYPE According to Biljecki et al. [2016b] LoD0.0, LoD0.1 models have no roof surface. LoD0.2-LoD1.3 models have horizontal roof surface. Models of LoD2.0 and above have more detailed roof structure. Therefore the roof type feature can have the following values:

- *0* - no roof surface
- *1* - horizontal roof surface
- *2* - non-horizontal roof surface
- *3* - a mix of horizontal and non-horizontal roof surfaces

PRESENCE OF WALLS The LoD0 family (0.0-0.3) does not allow representing walls, while models of LoD1.0 or higher can represent walls. Therefore this binary feature indicates the presence of walls in the model.

3.1.3 Building classification

The central problem of this thesis is to match buildings from the input data set to specific LoD classes. Using machine learning terminology, it is called supervised classification. Supervised, because the class labels (LoD classes) are provided for the classification algorithm and it needs to assign each building to an LoD class, based on some distance metric or probability. The choice of the appropriate algorithm depends on the statistical distribution of the features and there are no hard rules on algorithm selection, thus to certain extent the algorithm selection relies on experimentation.

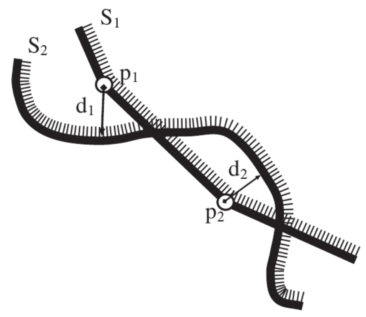


Figure 2: *Signed distance evaluation; distance is positive in p_1 and negative in p_2 (S_1 is the sampled curve).*

Figure 3.16: Signed distance evaluation; distance is positive in p1 and negative in p2 (S1 is the sampled curve). [Cignoni et al., 1998]

3.1.4 Classification accuracy and generalization performace

In order to evaluate the accuracy of the classification, the data needs to be labelled correctly. After classification, the inferred classes are compared to the correct labels, and the fraction of correctly classified objects is computed.

Furthermore, it is necessary to get an estimate on how a given algorithm performs on unseen data. In other terms, how well does the algorithm generalize what it learned from the training data to a different data set (e.g.different city model). To evaluate the generalization performance, the method *k-fold cross-validation* is used. In this method the data is split repeatedly in *k* parts, holding out one part as a test set and training the classifier on the rest. For each part, the classification accuracy is computed and the average accuracy is reported at the end Müller [2016].

3.2 STEP 2 — LOD INFERENCE WITH REFERENCE DATA

The goal of the second step is to use a reference data set to inform the LoD classification, particularly in cases where an object's geometry does not change between level of details (e.g.a simple detached garage). Therefore the object is compared to the reference and the similarity is used as an additional feature in the classification.

As reference this work uses the same point cloud that was used for generating the city model. Using the same data set for validation that was used for generating the data set can introduce the same errors in the validation as they are in the data, making the validation unreliable. However, to my knowledge there is no other open, 3D data set available in the same resolution for the area as the mentioned point cloud. After visually comparing both the AHN2 and AHN3 to the city model, it was identified that the AHN2 was used for generating the city model.

The *Metro* algorithm [Cignoni et al., 1998] which evaluates the difference between two meshes, by computing the signed, shortest (orthogonal) Euclidean distance between a point sample and the mesh. In case of a point cloud - mesh comparison, the algorithm computes the distance from a point to the nearest triangle plane in case the if the orthogonal projection of the point on this plane falls inside the triangle, otherwise it's the distance to the nearest edge that is taken. Therefore if the extent of the point cloud is larger than that of the mesh, the computed distances will be positive (see Figure 3.16).

The Root Mean Square Error (RMSE) is sensitive to outliers as it is proportional to the size of the squared errors, therefore it is potentially a more appropriate choice to detect if a part of a building is not modelled, but the point cloud contains it. The median distance would hide these cases if the missing part is small related to the

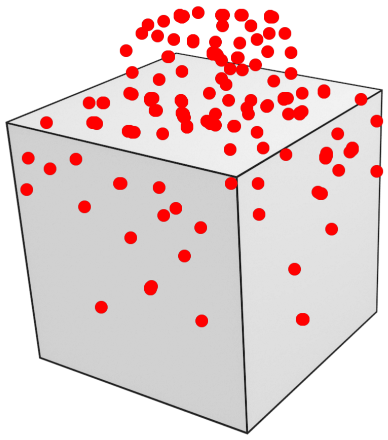


Figure 3.17: Building model (black) and its point cloud (red). A small part of the building is not modelled, but it is represented by the point cloud, causing larger than point cloud - mesh distances.

size of the building, thus only a small amount of distances are considered outliers, see Figure 3.17.

4 IMPLEMENTATION AND RESULTS

4.1 DATA PREPARATION

The thesis uses two 3D city models to test the developed method. One is randomly generated by procedural modelling, the other was created by combining building footprints and point cloud, and represents a part of Amsterdam city centre. A valid synthetic 3D city model can be generated in any of the sixteen LoD classes, with any amount of buildings [Biljecki et al., 2016a], therefore it was used for to test the method with the LoD classes that are not present in the Amsterdam data. One of the central concepts of the developed method is to train a LoD classifier on a 3D city model (design set) and apply it on a new model (target). Therefore the design set needs to contain all the possible LoD classes, thus it can recognise them in the target data. Among the city models that I explored, all of them contain models in LoD1 or LoD2 family, in various amounts from each class, and varying validity. Only Random3Dcity [Biljecki et al., 2016a] could provide an arbitrary amount of valid building models in each class. The Amsterdam data set is then used for testing the generalisation performance and transferability of the classifier. Both data sets are originally stored in (City)GML format.

OBJ is a file format that represents 3D geometry without semantics [OBJ, 2018] and is adopted by the GIS community to store 3D city models (e.g.Biljecki and Arroyo Ohori [2015]). Being a simple format, it is relatively easy to parse and write (as opposed to GML), which enables rapid development. As this thesis is concerned with the geometric LoD only, there is no need to use CityGML as input. Therefore all analysed data sets are translated from CityGML to OBJ by the CityGML2OBJs tool developed by Biljecki and Arroyo Ohori [2015].

SYNTHETIC CITY MODEL was generated with Random3DCity, in LoD 0.1-2.3 (except 1.0) and it contains 1000 buildings, hundred buildings per class (see Figure 4.1). The model is converted to OBJ format with CityGML2OBJs.

THE AMSTERDAM CITY MODEL was created in LoD1 and LoD2. The LoD2 model was provided by the company VirtualCitySystems[1] (see Figure 4.3). The LoD1 model was generated with *3dfier* (see Figure 4.2). For the generation of both LoD1 and LoD2 models, the building footprints of the Dutch *Basisregistraties Adressen en Gebouwen (BAG)*[2] are used in combination with the point cloud of the *Actueel Hoogtebestand Nederland (AHN)*[3]. To the extent of my knowledge, the LoD data set was created with the second version of AHN (AHN2), therefore this was used for the creation of the LoD1 model as well. However, a disadvantage of AHN2 compared to AHN3 is that in the former, the vegetation points are included in the same class as the building points.

The data set contains 656 building footprints, but due to removing invalid geometry, 482 remained.

The data set was prepared with FME Desktop[4] and converted into OBJ with the *CityGML2OBJs* tool. The OBJ writer of FME dumps all buildings into one group,

1 http://www.virtualcitysystems.de
2 https://www.kadaster.nl/bag
3 http://www.ahn.nl
4 https://www.safe.com/fme/fme-desktop/

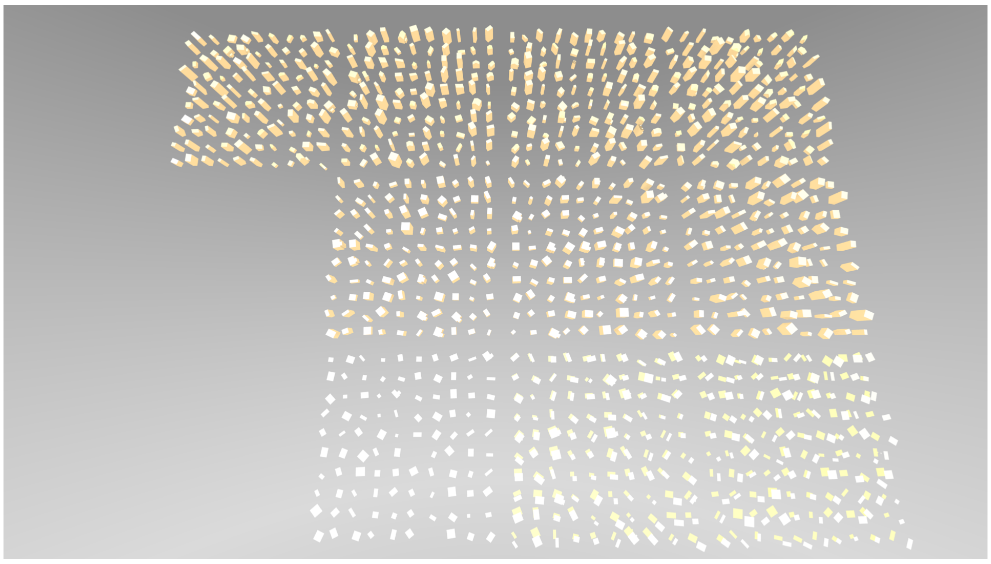


Figure 4.1: The generated synthetic data set of 1000 buildings, 10 LoD classes

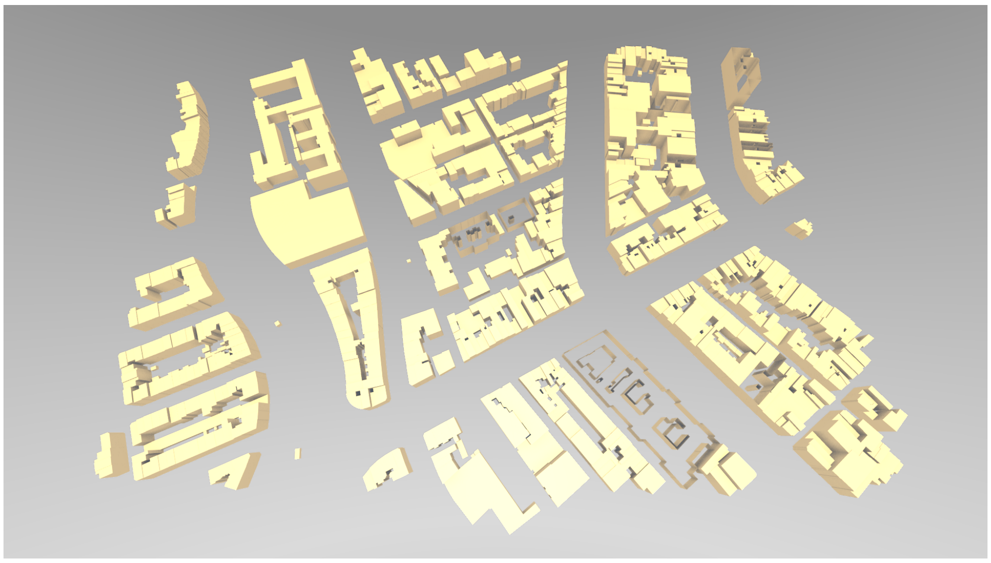


Figure 4.2: LoD1 version of the Amsterdam data set

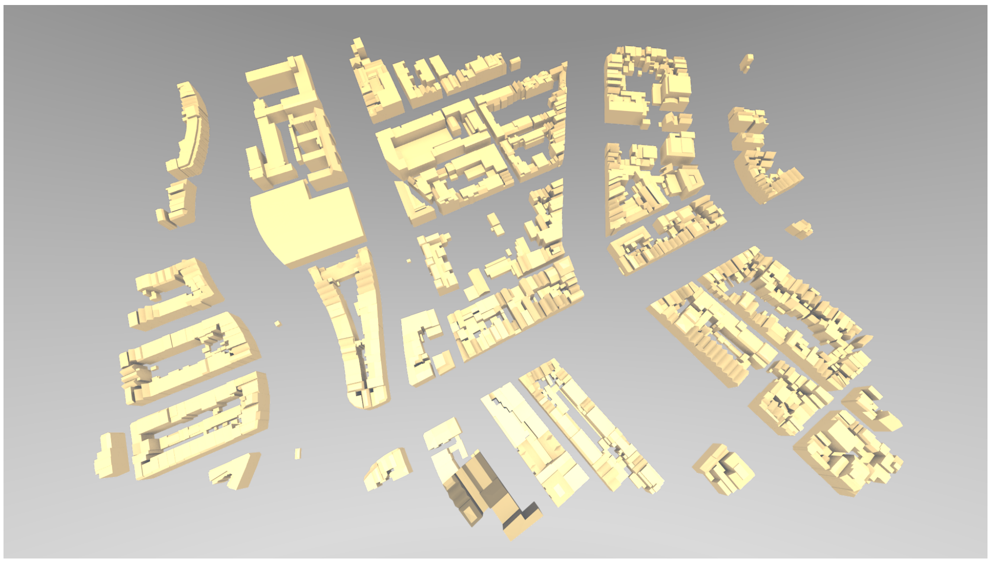


Figure 4.3: LoD2 version of the Amsterdam data set

Table 4.1: Data set statistics

Data set	Property	Mean	Standard deviation
Synthetic LoD2	NSCP	4,26	0,99
	Nr. vertices	60	23,34
	Nr. triangles	15	5,83
Amsterdam LoD2	NSCP	6,37	4,28
	Nr. vertices	212,83	228,74
	Nr. triangles	53,21	57,19

which makes it tedious to split them for the point cloud comparison. Therefore *CityGML2OBJs* is more suitable as it can write one object per building.

The original LoD2 model contains semantic surfaces which are not required for the analysis. Therefore the semantic surfaces are removed and the buildings are modelled in LoD2 with one `gml:Solid`. Furthermore, geometric errors such as invalid surface orientation, non-planar surfaces, degenerate or corrupt geometries, self-intersection in 2D are repaired in the conversion process, using FME's *GeometryValidator* transformer.

Table4.1 shows that the average number and standard deviation of NCSP, vertices and triangles are lower in the synthetic data (LoD2) than in the Amsterdam data set which is due to the simpler shapes in the former.

4.1.1 Validating the LoD of the Amsterdam data set

Finally, the Level of Detail was manually validated against the digital surface model provided by Google Earth. This is aimed to identify models with invalid LoD, for example where the roof reconstruction failed and an LoD1 model is generated instead of an LoD2 model. The Google Earth mesh was the only available ground truth that I could use to compare the 3D city model against.

However, there are a few problems with using Google Earth as a 3D ground truth:

- The positional and temporal accuracy of the model is unknown.
- It is not possible to take 3D measurements on the surface model, thus comparison is only possible by the eye.

Therefore the LoD validation itself is an ambiguous process (see Section 2.2). Also, considering that there are certain roof shapes that are in between of the classic gabled and flat shapes and include dormers. These roofs could be modelled partially flat or totally flat. In these cases the decision whether a model is valid or not might be subjective.

Several models are in fact composed of a group of buildings. Further investigation revealed that this is due the geometry of BAG footprint and the the 3D modelling process, thus these cases are considered valid if the roof was valid.

The outcome of data preparation is 621 remaining LoD2 building models, from which 55 are with invalid LoD.

4.2 GENERATING FEATURES

Features are generated in an FME workflow that takes a triangulated city model in OBJ format as an input and outputs a CSV file with the features per object. The building models must be stored as named objects in the OBJ file, and have a unique identifier in order to match the computed features to the objects.

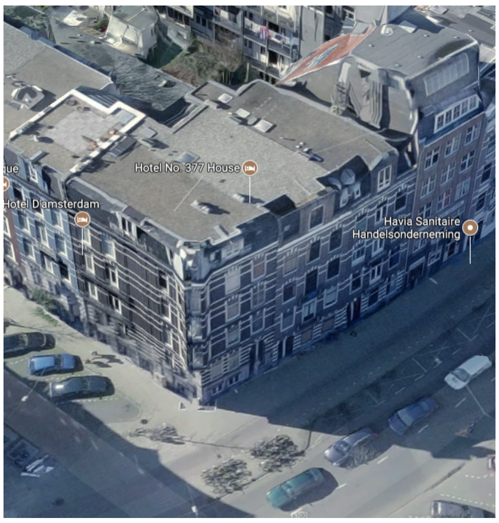


Figure 4.4: Googe Earth mesh

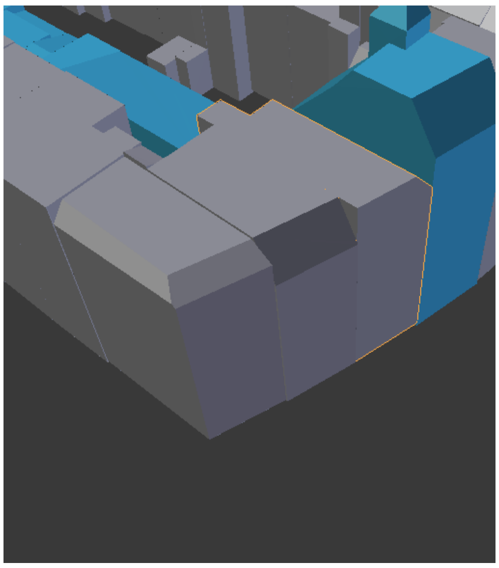


Figure 4.5: LoD2 model

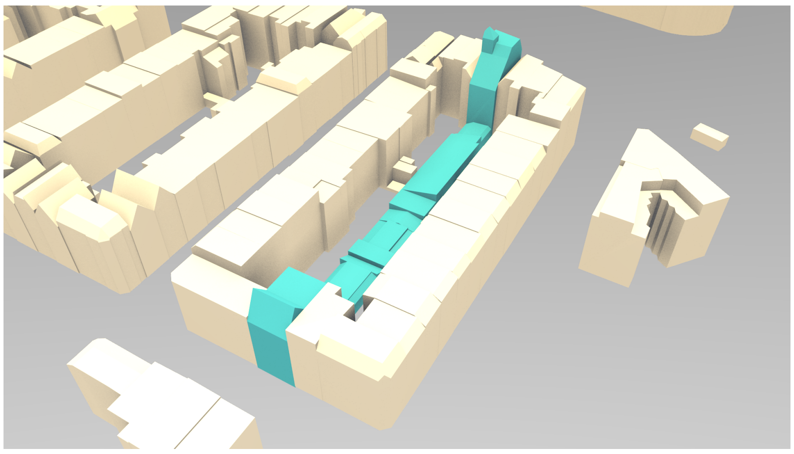


Figure 4.6: Combined buildings into one footprint and model (blue)

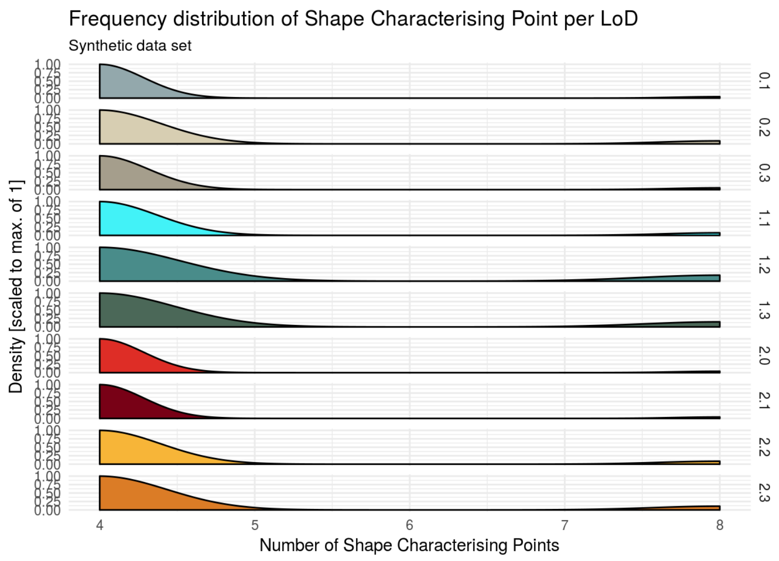


Figure 4.7: Distribution of NSCP in the Synthetic data

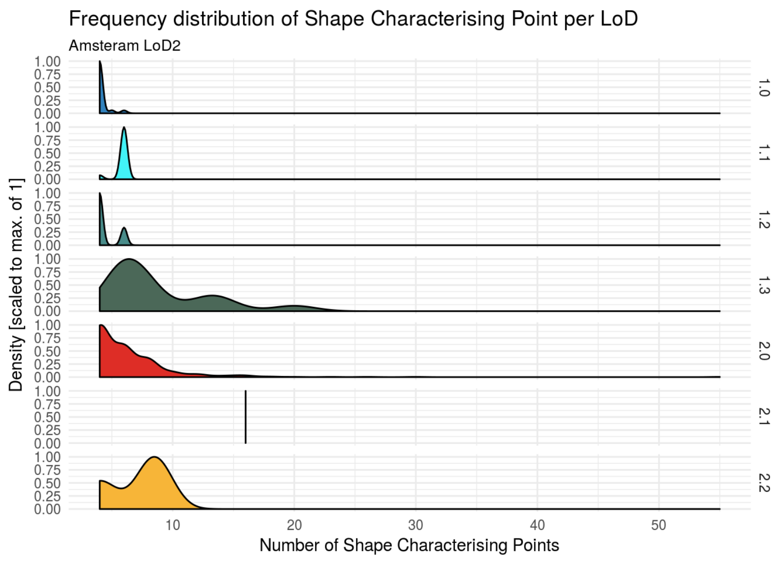


Figure 4.8: Distribution of NSCP in the Amsterdam data

4.2.1 Step 1

The distributions of the generated features might reveal how similar are the LoD classes in the features space. Comparing the Gaussian kernel density estimates of *NSCP* (Figure 4.7, 4.8), the *SCL* distributions (Figure 4.9, 4.10, 4.11, 4.11), the distributions of *footprint and building part area* (Figure 4.13, 4.14, 4.15, 4.16) and the distributions of the *footprint-roof triangle ratio* (Figure 4.17, 4.18) reveal that in case of the synthetic data the distribution of features are very similar across all LoD classes, while in case of the Amsterdam data there are considerable differences.

In case of *SCL* the plot 4.9 show the presence of edges that are shorter than 4m in the LoD classes *0.1*, *2.0*, which should not be the case according to the interpretation. Thus the SCL alone is insufficient to reason about the size of the building parts.

In case of *footprint area* he synthetic data set does not seem to respect the LoD requirements, as objects of all *1,2,3* categories are present in similar quantities across all LoD classes (Figure 4.13, 4.14).

The utility of the *footprint-roof triangles ratio* measure it highly dependent on the local architecture. While in case of Amsterdam old town, were most of the buildings

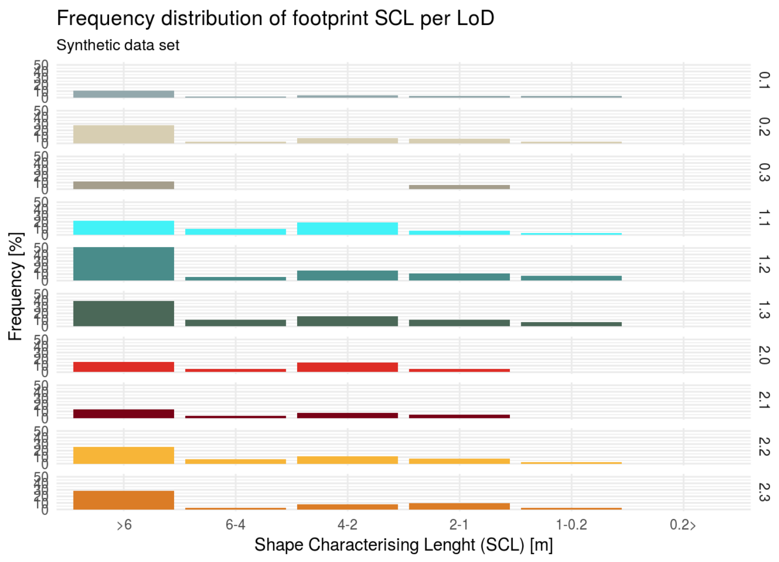


Figure 4.9: Shape Characterising Lengths in the Synthetic data

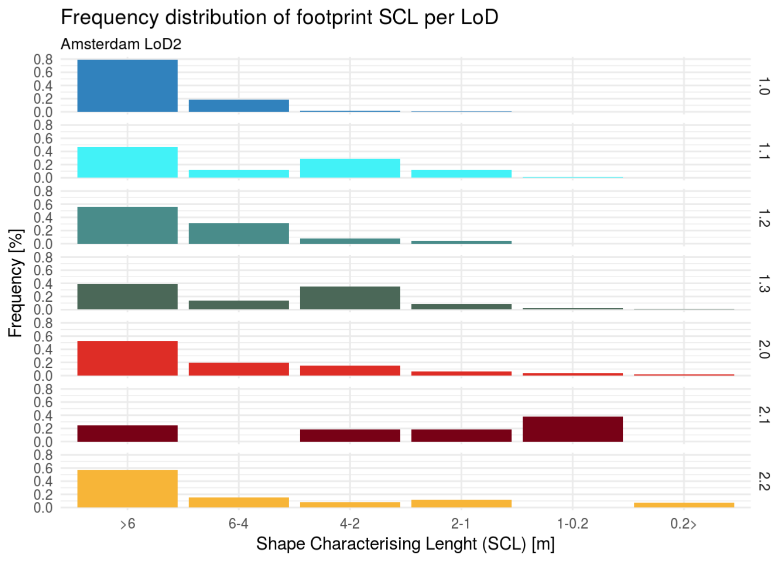


Figure 4.10: Shape Characterising Lengths in the Amsterdam data

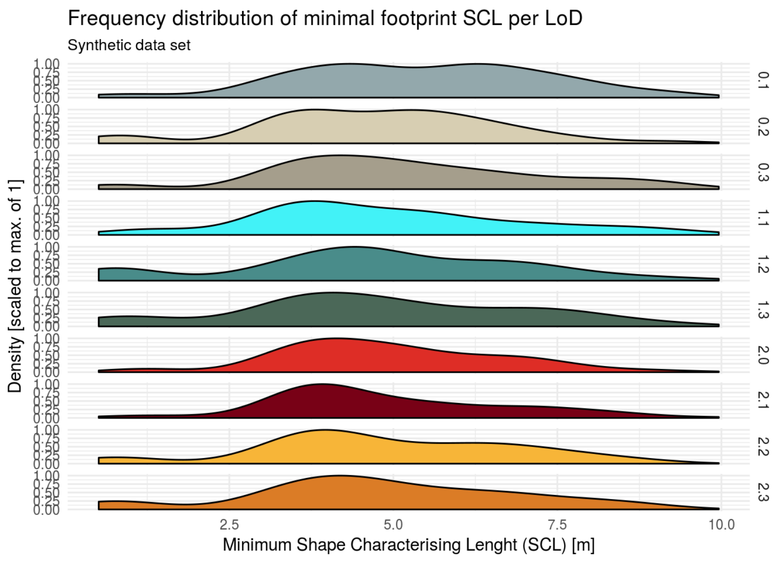


Figure 4.11: Minimal Shape Characterising Lengths in the Synthetic data

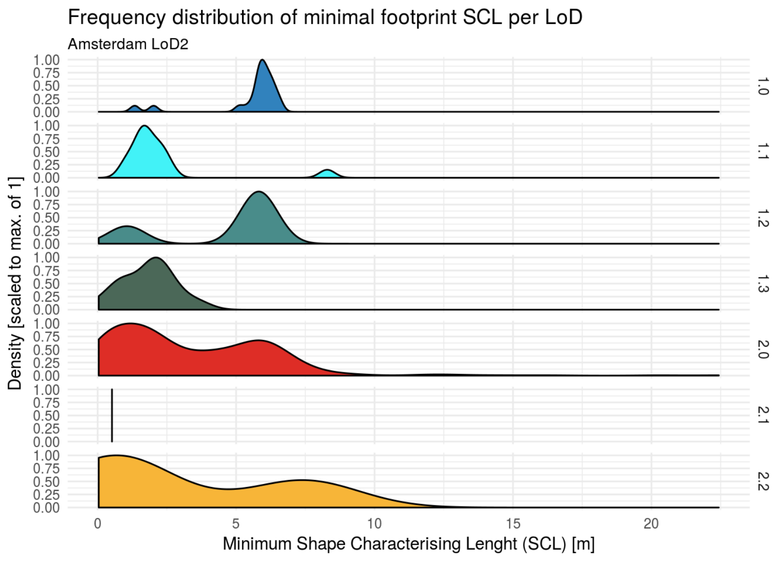


Figure 4.12: Minimal Shape Characterising Lengths in the Amsterdam data

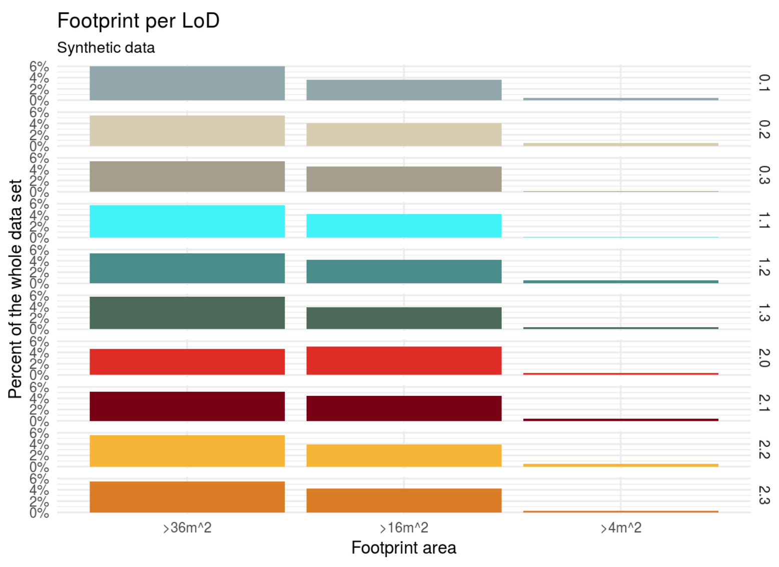


Figure 4.13: Footprint area in the Synthetic data

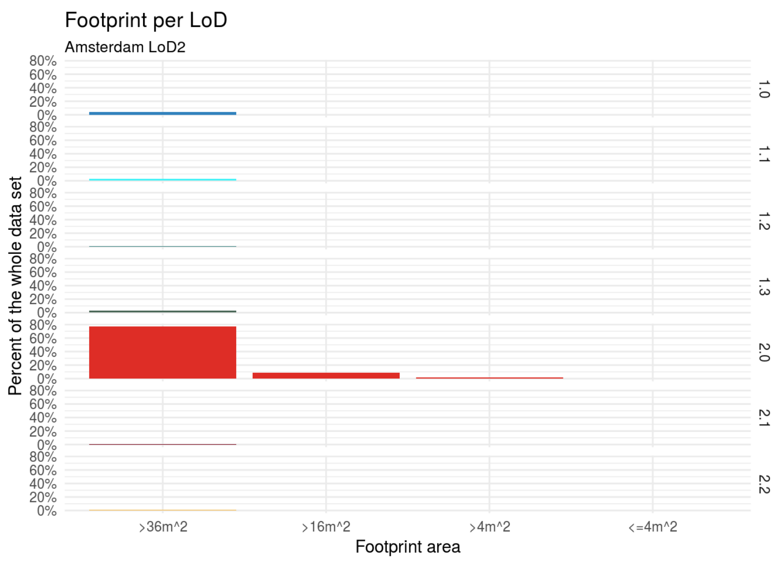


Figure 4.14: Footprint area in the Amsterdam data

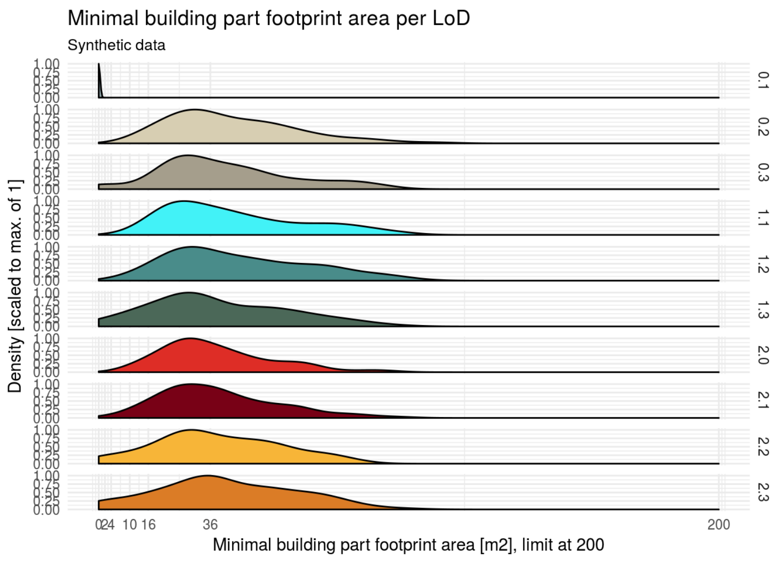


Figure 4.15: Minimal building part area in the Synthetic data

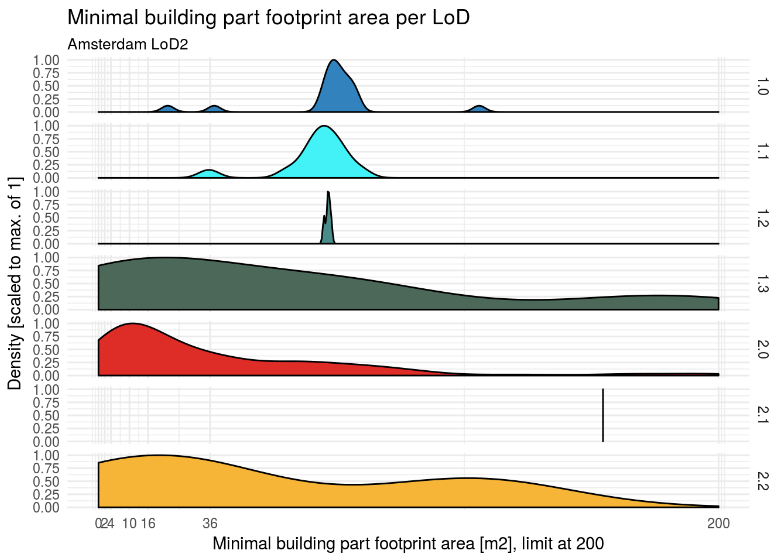


Figure 4.16: Minimal building part area in the Amsterdam data

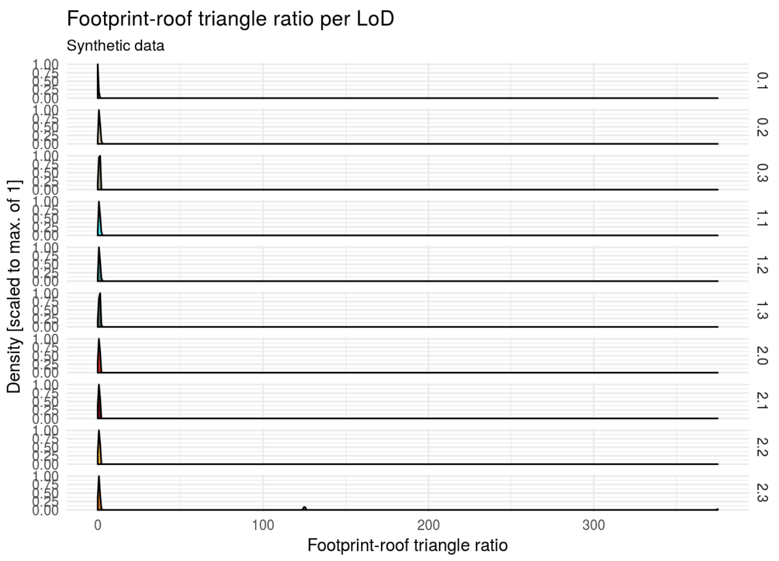


Figure 4.17: Footprint-roof triangle ratio in the Synthetic data

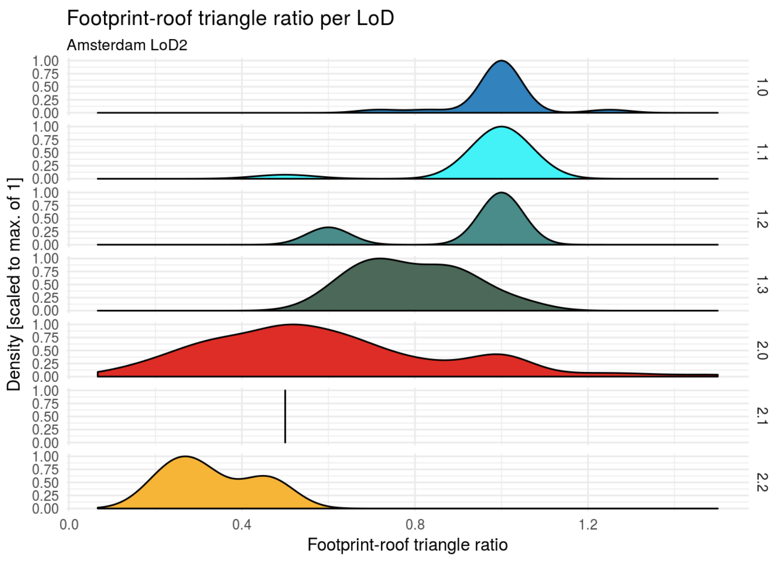


Figure 4.18: Footprint-roof triangle ratio in the Amsterdam data

have a complex, gabled roof a ratio close to 1 could indicate a failed roof reconstruction (Figure 4.18). In case of the synthetic data that mainly contains buildings with simple geometry across all LoD, the utility of this measure is questionable as the majority of the buildings have a ratio close to 1 in all LoDs. The exception is LoD0.1 which does not have roof surface at all, thus has a ratio of 0 (Figure 4.17).

4.2.2 Step 2

The distance between the point cloud and the model is computed in the following procedure:

0. The OBJ file and the point cloud need to be in the same coordinate reference system.

1. In the OBJ file, each building need to be a separate object. This allows to split the file into a separate file per building.

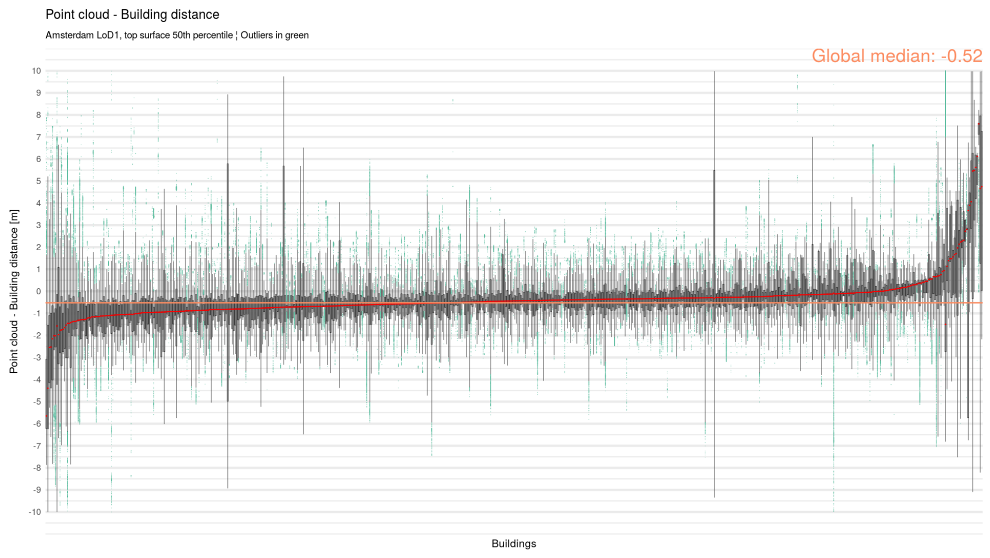


Figure 4.19: Point cloud to model distances in the LoD1 model

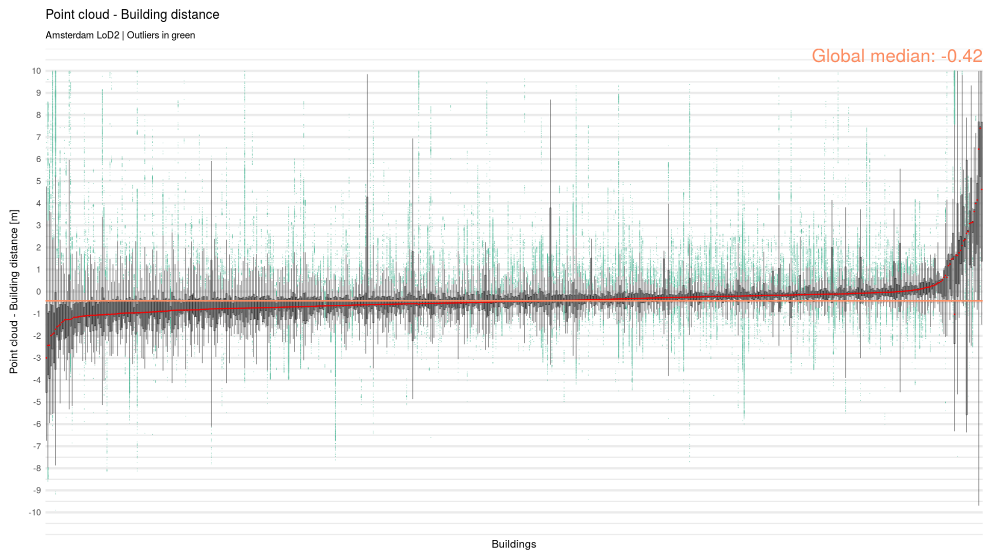


Figure 4.20: Point cloud to model distances in the LoD2 model

2. The building footprint polygons are precomputed in order to speed up the comparison process.

3. If the point cloud is stored with offset coordinates in order to reduce the size of the coordinates, store the offset coordinates.

For each building:

```
1. Crop the point cloud with the footprint polygon.

2. Match the centers of the point cloud and the buildings.

3. Compute the signed point cloud - mesh distances.

4. Export the result to a table.
```

The figures 4.19 and 4.20 do not indicate significant difference between the point cloud – mesh distances of LoD1 and LoD2 models. The median distance of the

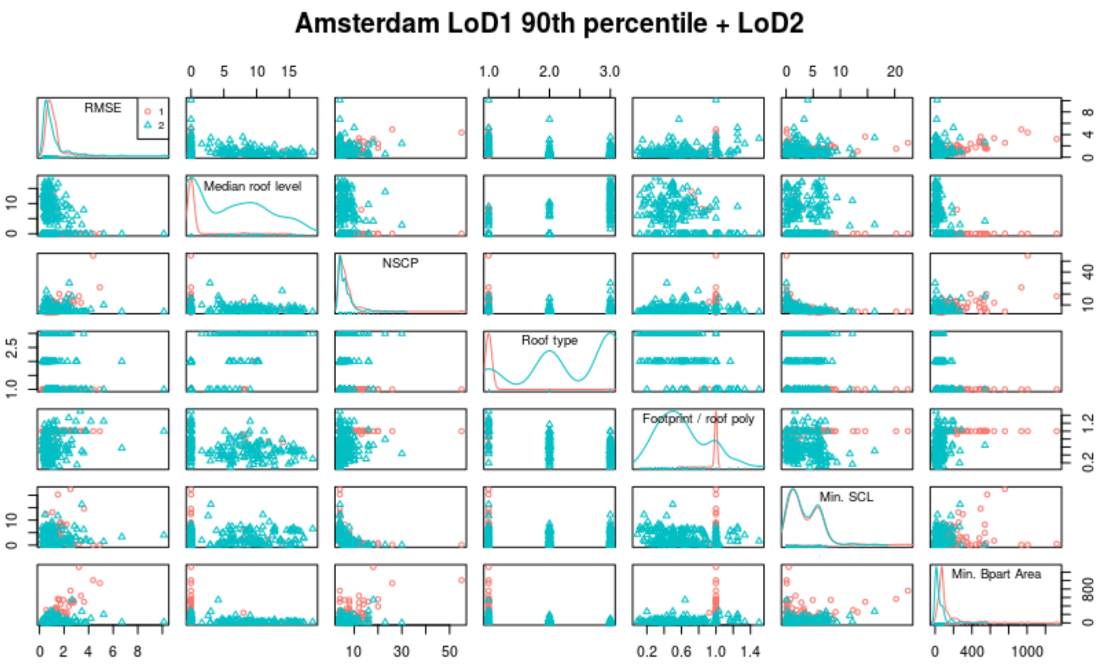


Figure 4.21: RMSE of point cloud to model distances in relation to the other features. Amsterdam data set, 50% LoD1 (red circle), 50% LoD2 (blue triangle).

whole city model is $-0.52m$ for the LoD1 model, and $-0.42m$ for the LoD2 model, which is very little difference. The box plots suggest that the variation of the distances (length of the black boxes) in LoD1 models are generally larger than those of LoD2 models, but there are more distances that are considered outliers (green dots) at the LoD2 models than at LoD1 models. This indicates that the LoD2 models are indeed closer to their point cloud then the LoD1, but there is a very high amount of noise in the measurements.

The scatter plot matrix (Figure 4.21) shows the Root Mean Square Error (RMSE) computed from the point cloud distances in relation to other features. The RMSE is in the leftmost column, and each row shows a scatter plot of RMSE and another feature. The diagonal of the plot shows the distribution of a feature. The distribution of RMSE in LoD1 (red) and LoD2 (blue) models overlap almost entirely. The long tail of both distributions is accountable for the noise (mainly vegetation, see below) in the measurements. Apart from the overlapping distributions, a closer look on the plots reveals that the RMSE does not separate the two classes in any of the feature combinations. The clearest separation is provided by the *RMSE - Min. bulding part area* combination, but even in this case the largest part of the two classes overlap.

In order to analyse the distances more in detail, five buildings were selected (see 4.22). By visual evaluation, each building but *. . . 68821* approximate their point cloud very closely. The building *. . . 68821* is included as a reference for an LoD2 model that is relatively distant from its point cloud.

The spikes in the positive range are probably caused by under-extrusion, overreaching vegetation or roof superstructures (Figure 4.31). This is the case for buildings *68818*, *69703* (see 4.23).

In 4.24 and in 4.25, the buildings with unusually high standard deviation of distances are marked red. The buildings with unusually high median distance are marked green. The buildings that have both usually high median distance and standard deviation, are marked blue.

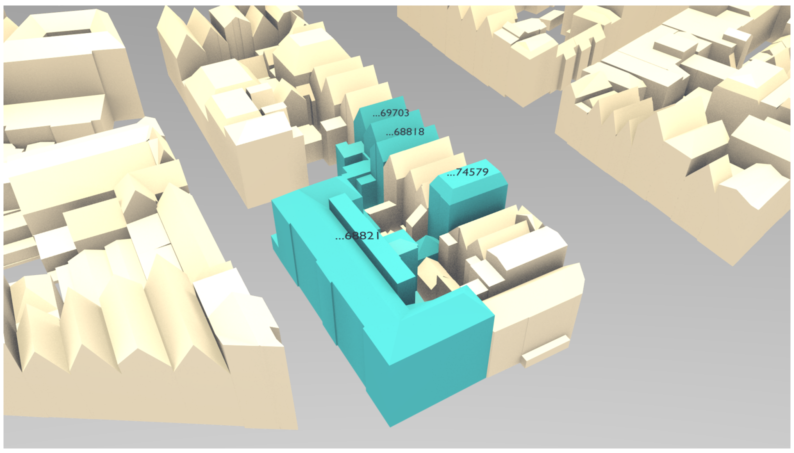


Figure 4.22: Selection

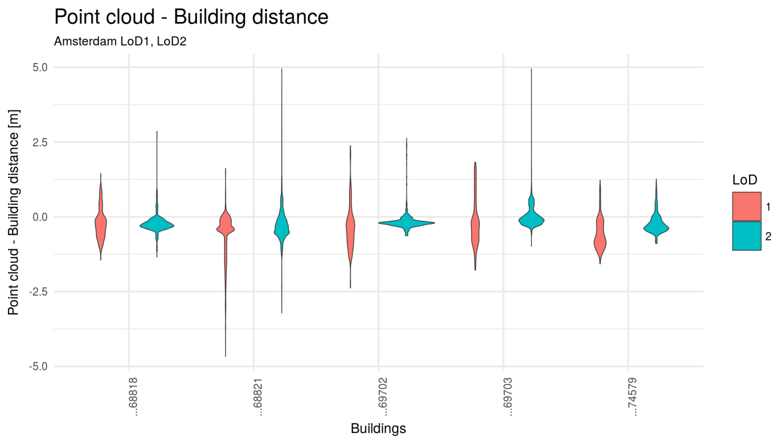


Figure 4.23: Point cloud - model distances of the five selected buildings (those in blue in Figure 4.22)

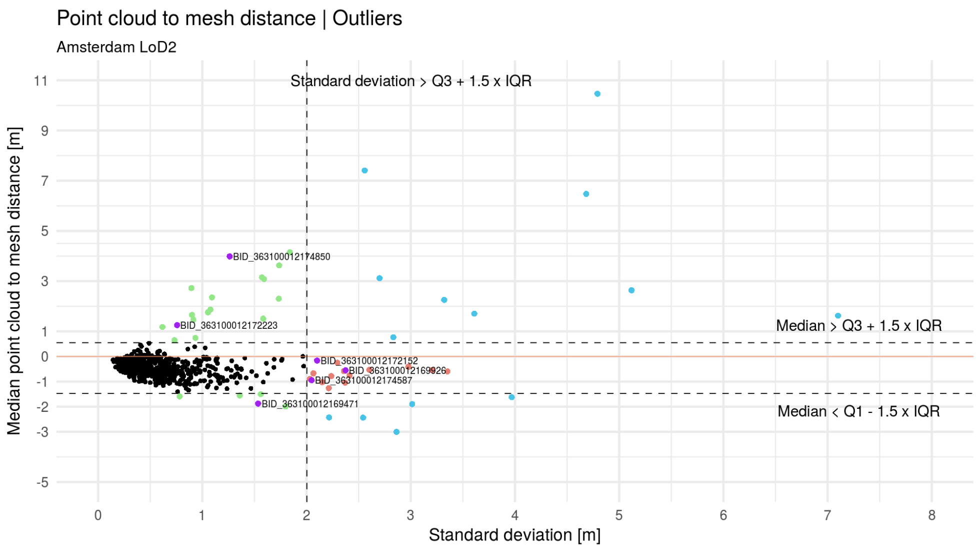


Figure 4.24: Outliers in the computed distances

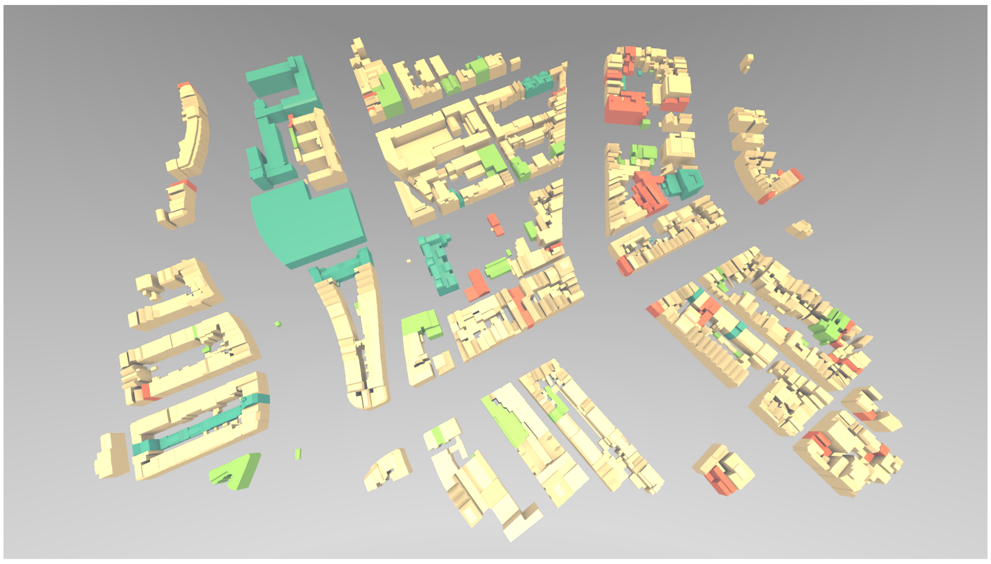


Figure 4.25: Outliers

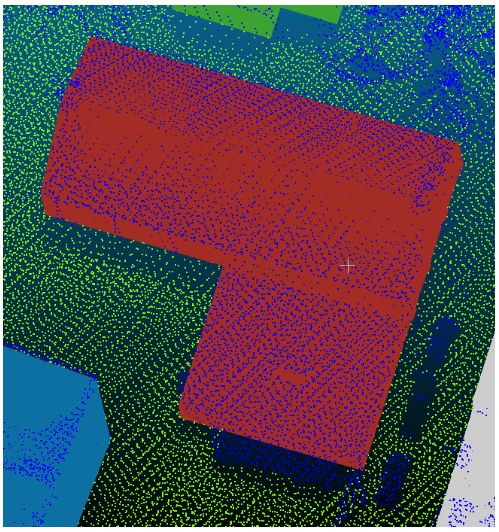


Figure 4.26: BID_363100012169926

OUTLIERS OF STANDARD DEVIATION These buildings have a normal median distance but the distances have an unusually high standard deviation. These outliers are usually due to a missing building part. See 4.26, 4.27.

OUTLIERS OF MEDIAN DISTANCE These buildings have a normal standard deviation of distances but unusually high median distance. These outliers are usually due to tall vegetation (trees) that reach above the building, thus their points are included when the point cloud is clipped with the footprint. This is the case with the building *BID_363100012169051* (see 4.28).

The other cause for these outliers is large, missing structures from the model. Opposed to the SD outliers, in this case a substantial part of the building or the roof is missing, thus most of the points have large distances within the building footprint. This is the case with the building *BID_363100012169471* (see 4.29).

OUTLIERS OF BOTH MEDIAN DISTANCE AND STANDARD DEVIATION These buildings have unusually high standard deviation of distances and median distance. These outliers are usually due to buildings that failed to reconstruct in LoD2, such

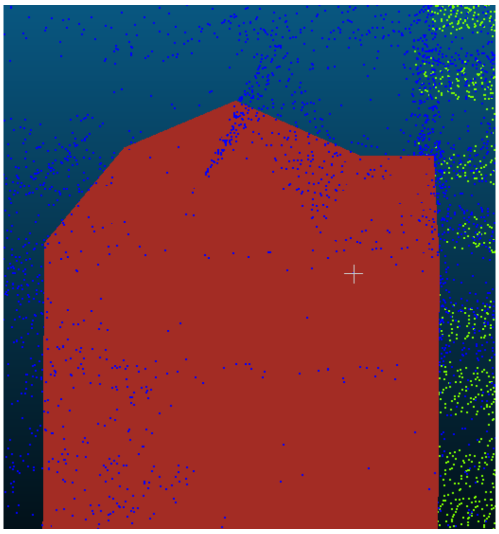


Figure 4.27: BID_363100012174587

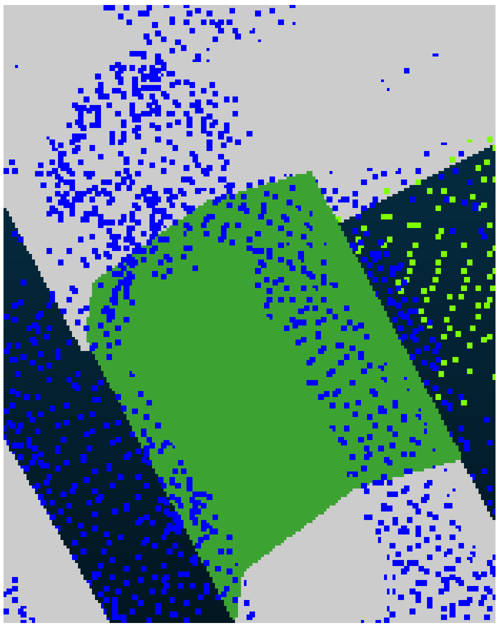


Figure 4.28: BID_363100012169051

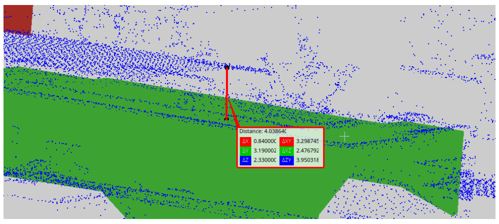


Figure 4.29: BID_363100012169471

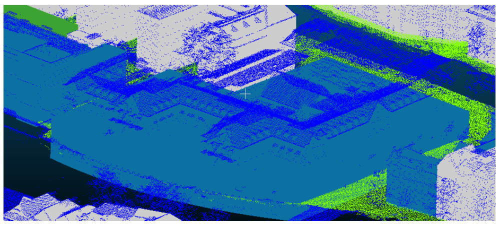


Figure 4.30: BID_363100012242951

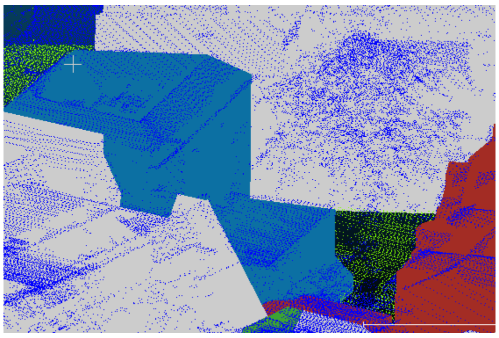


Figure 4.31: BID_363100012174769

as *BID_363100012242951* (see 4.30). Or they are largely covered by a tall tree, such as *BID_363100012174769* (see 4.31).

4.3 CLASSIFYING THE LOD WITH MACHINE-LEARNING

Six experiments were conducted in order to test the method in various scenarios (see 4.2). The features used for the classification are:

- Number of Shape Characterising Points,
- Minimal SCL,
- Footprint area,
- Minimal Building Part area,
- Roof overhang,
- Footprint-roof triangle ratio,
- Median roof level,
- Roof type,
- Walls,
- RMSE of point cloud-model distances.

The *roof type* is a categorical variable. It is encoded as integer values but such representation is not directly usable with scikit-learn estimators, because these expect continuous values. Simply using the integers in learning, scikit-learn would interpret the values as they are ordered [Pedregosa et al., 2011]. Therefore the *roof type* is encoded into binary variables, one variable per class, with *one-hot encoding*.

The features *walls* and *roof overhang* are binary, they do not need to be standardized.

The generalization performance of the learning methods is estimated with k-fold cross-validation. Both the synthetic and Amsterdam data is split into 80% training

Table 4.2: Overview of experiments

#	Goal	Features and Classes	Data
1.	Test classification performance on raw features	Not standardised; Multi-label	Synthetic, Amsterdam LoD2
2.	Test classification performance on standardised features	Standardised; Multi-label	Synthetic, Amsterdam LoD2
3.	Test geographical transfer of classifiers	Standardised; Multi-label	Train on synthetic, test on Amsterdam LoD2
4.	Test binary classification when incl. Point cloud distances	Standardised, incl. RMSE; Binary	Amsterdam LoD2
5.	Test the effect of LoD class imbalance	Standardised, incl. RMSE; Binary and Multi-label	Amsterdam LoD2
6.	Test the effect of LoD class imbalance and geographical transfer	Standardised; Multi-label	Train on synthetic, test on Amsterdam LoD2

and 20% test data set, and using 10-fold cross-validation. The random number seed of the cross-validation is reset before each run to ensure that the evaluation of each algorithm is performed using exactly the same data splits. It ensures the results are directly comparable.

For evaluating the performance of the classifiers, both the mean accuracy and its standard deviation are considered. A classifier that achieves a certain accuracy with a low standard deviation is considered more reliable than one with the same accuracy but higher standard deviation.

The synthetic data set poses a balanced classification problem, as each of the 10 classes contain 100 objects, thus contains 1000 objects in total. The Amsterdam data set poses an imbalanced classification problem [Han and Kamber, 2011] as the number of objects in each class differ largely, with LoD2.0 being the dominant class, and contains 482 objects in total.

```
1.0  1.1  1.2  1.3  2.0  2.1  2.2
 20   14    4   15  425    1    3
```

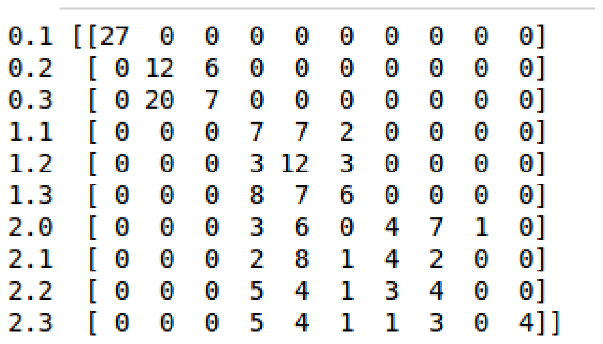
4.3.1 Experiment 1 – Raw features

All features but *roof type* are left as they are, without standardization. Table A.1 and figure 4.34 summarises the classification results on the synthetic data set. Table A.2 and figure 4.35 summarises the classification results on the Amsterdam data set. In the synthetic data the Logistic Regression (LR) achieves the best cross-validation score, due to its relatively high average accuracy and relatively low standard deviation. In the Amsterdam data the Decision Tree (DTree) performs the best.

Predicting LoD classes in the synthetic data with LR achieves an accuracy of 40.5%. The confusion matrix 4.32 show that most the 0.3 class is predicted as 0.2, none of the 1.1 or higher classes are predicted as an LoD0 family. However, in general objects are classified into a lower class than they belong to.

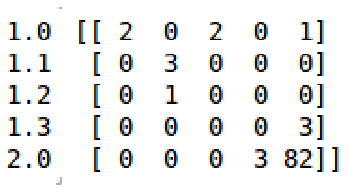
Prediction with Decision Tree in the Amsterdam data achieves an accuracy of 89.6%. The confusion matrix 4.32 show that the 2.1 and 2.2 classes are missing from the objects, and this is due to the low amount of objects in the test set. Most of the LoD2.0 classes are predicted correctly, however none of 1.2 or 1.3 are predicted correctly. There is a clear confusion between the LoD1.3 and 2.0 classes.

Prediction accuracy in the synthetic data is order of magnitudes lower than in the Amsterdam data. This is likely due to very similar distribution of many features



```
0.1 [[27  0  0  0  0  0  0  0  0  0]
0.2  [ 0 12  6  0  0  0  0  0  0  0]
0.3  [ 0 20  7  0  0  0  0  0  0  0]
1.1  [ 0  0  0  7  7  2  0  0  0  0]
1.2  [ 0  0  0  3 12  3  0  0  0  0]
1.3  [ 0  0  0  8  7  6  0  0  0  0]
2.0  [ 0  0  0  3  6  0  4  7  1  0]
2.1  [ 0  0  0  2  8  1  4  2  0  0]
2.2  [ 0  0  0  5  4  1  3  4  0  0]
2.3  [ 0  0  0  5  4  1  1  3  0  4]]
```

Figure 4.32: Confusion matrix of the LoD prediction in the synthetic data, raw features. Correct classes are in the rows, predicted classes in columns.



```
1.0 [[ 2  0  2  0  1]
1.1  [ 0  3  0  0  0]
1.2  [ 0  1  0  0  0]
1.3  [ 0  0  0  0  3]
2.0  [ 0  0  0  3 82]]
```

Figure 4.33: Confusion matrix of the LoD prediction in the Amsterdam data, raw features. Correct classes are in the rows, predicted classes in columns.

(4.7, 4.11, 4.15, 4.17) across all the LoDs in the synthetic data, thus the classifiers cannot distinguish between the classes. Features in the Amsterdam data have clear distinction between most of the LoD classes, however 4.33 shows that the LoD1.3 and LoD2.0 are confused in all cases.

4.3.2 Experiment 2 – Standardized features

Features *Number of Shape Characterising Points, Minimal SCL, Footprint area, Minimal Building Part area, Footprint-roof triangle ratio, Median roof level* are standardized. Table A.3 and figure 4.38 summarises the classification results on the synthetic data set. Table A.4 and figure 4.39 summarises the classification results on the Amsterdam data set. The results show a slight increase in classification accuracy and decrease in its standard deviation, in case of the Logistic Regression, k-NN, Decision Tree and SVM classifiers. Naive Bayes and Linear Discriminant Analysis are seemingly unaffected by standardization. Still the LR is the best performing classifier in case of the synthetic data, and DTree in case of the Amsterdam data.

Predicting LoD classes in the standardized synthetic data with LR achieves an accuracy of 42.5%. Predicting LoD classes in the standardized Amsterdam data with DTree achieves an accuracy of 88.6%. The confusion matrix of the synthetic data 4.36 and the Amsterdam data 4.37 show the same type of misclassification as in case of raw features.

4.3.3 Experiment 3 – Prediction in the Amsterdam data

In this case the best performing classifier (LR) in 4.3.2 is trained on the synthetic data and the LoD classes are predicted in the Amsterdam data. Standardized features were used. This experiment is meant to simulate a classifier that is trained on a generic and valid data set with all the LoD classes, and used for predicting the LoD classes in a novel data set.

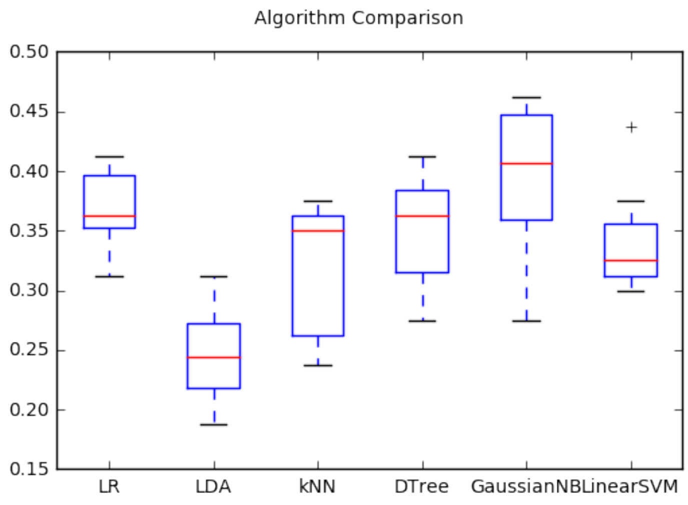


Figure 4.34: Classification accuracy and standard deviation on the synthetic data, raw features.

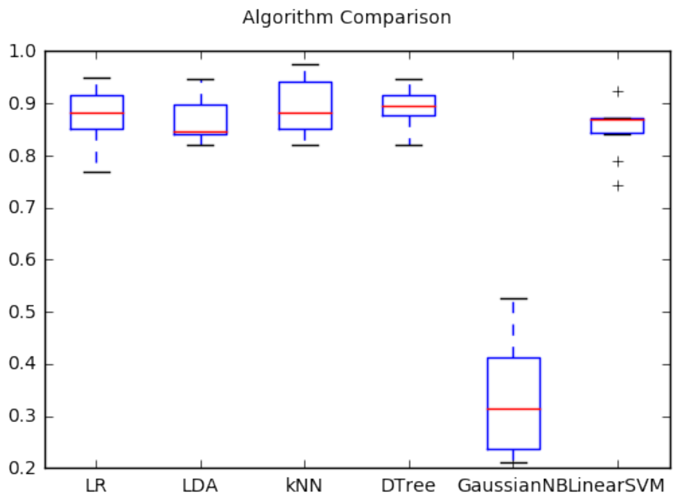
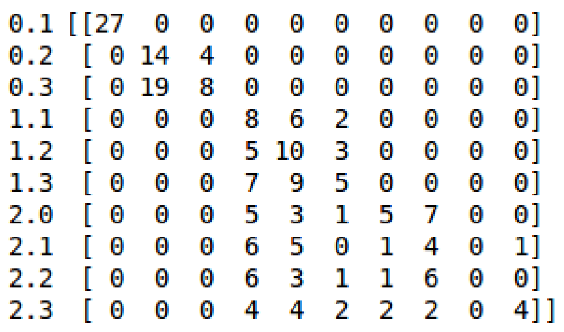


Figure 4.35: Classification accuracy and standard deviation on the Amsterdam data, raw features



```
0.1 [[27  0  0  0  0  0  0  0  0  0]
0.2  [ 0 14  4  0  0  0  0  0  0  0]
0.3  [ 0 19  8  0  0  0  0  0  0  0]
1.1  [ 0  0  0  8  6  2  0  0  0  0]
1.2  [ 0  0  0  5 10  3  0  0  0  0]
1.3  [ 0  0  0  7  9  5  0  0  0  0]
2.0  [ 0  0  0  5  3  1  5  7  0  0]
2.1  [ 0  0  0  6  5  0  1  4  0  1]
2.2  [ 0  0  0  6  3  1  1  6  0  0]
2.3  [ 0  0  0  4  4  2  2  2  0  4]]
```

Figure 4.36: Confusion matrix of the LoD prediction in the synthetic data, standardized features. Correct classes are in the rows, predicted classes in columns.

```
1.0 [[ 3  0  1  0  1]
1.1  [ 0  3  0  0  0]
1.2  [ 0  1  0  0  0]
1.3  [ 0  0  0  0  3]
2.0  [ 0  0  0  3 82]]
```

Figure 4.37: Confusion matrix of the LoD prediction in the Amsterdam data, standardized features. Correct classes are in the rows, predicted classes in columns.

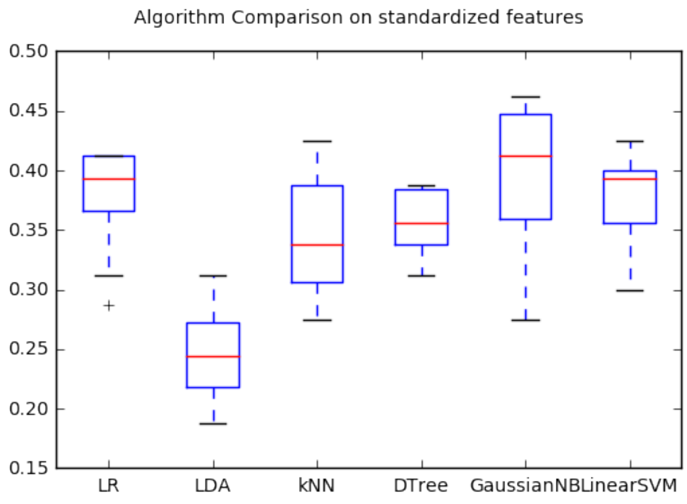


Figure 4.38: Classification accuracy and standard deviation on the synthetic data, standardized features

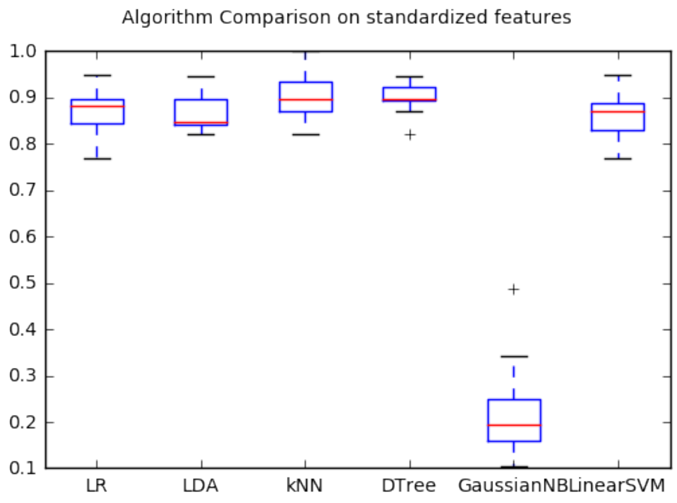


Figure 4.39: Classification accuracy and standard deviation on the Amsterdam data, standardized features

Table 4.3: Confusion matrix

	not LoD2	LoD2
not LoD2	7	5
LoD2	2	83

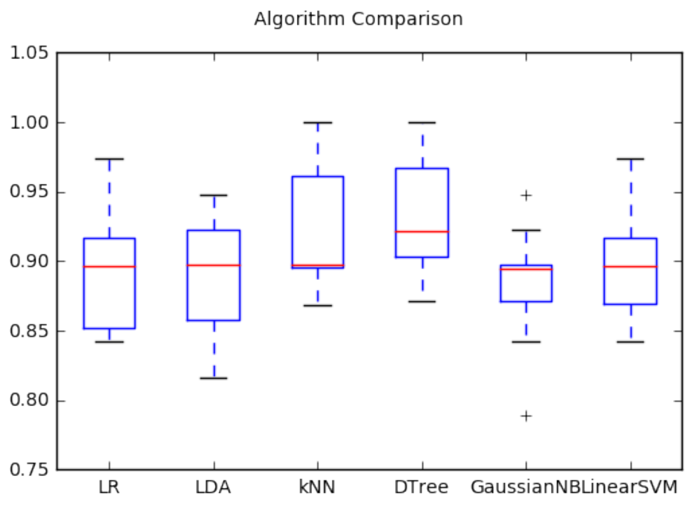


Figure 4.40: Binary classification accuracy and standard deviation on the Amsterdam data, standardized features

Logistic Regression achieves an accuracy of 7.4%, Decision Tree 3.7%, Naive Bayes 0.0%. There scores are extremely low and indicate the the current set up is completely unsuitable to predict LoD classes.

4.3.4 Experiment 4 – Predict LoD2 or not

In this case it is tested how accurately can the classifier distinguish between LoD2 and non-Lod2 objects in the Amsterdam data, with including the *point cloud - mesh* distances. The features are standardized in the same manner as in 4.3.2, including the *root mean square error* of the distances. The LoD classes are reduced to binary, *LoD2* and *not LoD2*. Due to the imbalanced classes (89% LoD2), the class weights are adjusted inversely proportional to class frequencies in the input data, in the classifiers LR, DTree and LinearSVM. Then the classifiers are trained and tested in the same setup as in 4.3.2.

Figure 4.40 and table A.5 summarises the cross-validation results. The Decision Tree classifier provides the best performance in terms of average accuracy and standard deviation. Predicting the LoD classes in the test set is achieved with 92.7% accuracy. This accounts for a 4% increase in classification accuracy compared to 4.3.2. However, the confusion matrix 4.3 shows that almost half of the LoD1 objects are classified as LoD2, while only 2 LoD2 objects are classified as LoD1.

4.3.5 Experiment 5 – Prediction in the Amsterdam data with mixed LoD1 and LoD2

The aim of this experiment is to observe how the classifiers behave when various amounts of LoD1 models are mixed into an LoD2 model. The Amsterdam data set is used. Therefore 10%, 25% and 50% of LoD2 objects were replaced by their LoD1.2 variant. The LoD1 models were created with *3dfier*, using the same footprints as the LoD2, the same point cloud (AHN2), and the top surface of the LoD1.2 models is

Table 4.4: Prediction accuracy of the kNN classifier with various amounts of LoD1 models

% of LoD1	Multi-label	Accuracy (%)
10	Yes	65.9
10	No	70.1
25	Yes	84.5
25	No	91.7
50	Yes	85.5
50	No	91.7

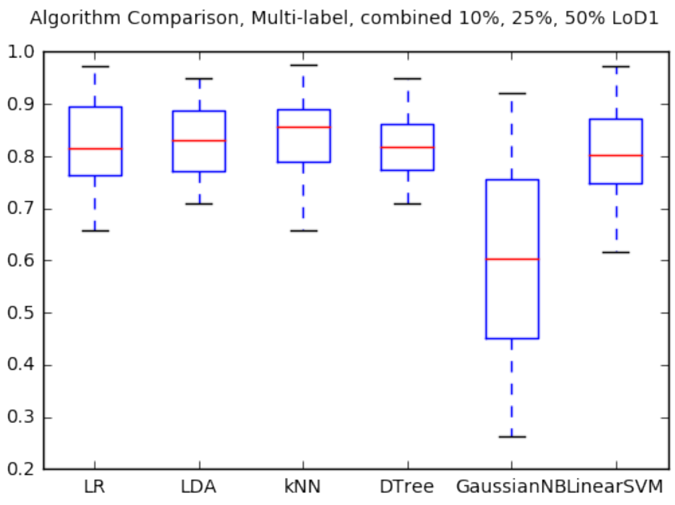


Figure 4.41: Multi-label classification accuracy, combined values.

set to the 90th percentile of the total height of the building. The cross-validation setup and feature transformation is identical to 4.3.4.

The classification was tested both in a multi-label and binary case. Figures 4.41 and 4.42 show combined cross-validation accuracies from the data sets with 10%, 25%, 50% of LoD1.2 models. In both cases the kNN classifers performs the best in terms of median accuracy and standard deviation.

The prediction performance 4.4 clearly show that the binary classification (LoD2 or not) outperform the multi-label classification.

4.3.6 Experiment 6 – Training in the synthetic, predicting in the Amsterdam data

This experiment combines 4.3.3 and 4.3.5, with the aim of observing the classifier performance when it is trained on the synthetic data (LoD1.1-2.3) and predicts classes in the Amsterdam data containing various amounts of LoD1.2 models. However, opposed to 4.3.3, in this case the LoD0 family was removed from the synthetic data set.

Figure 4.43 shows that the classifiers perform considerably better than in 4.3.3, with Logistic Regression providing the highest median accuracy and lowest standard deviation. Table shows the prediction accuracy of Logistic Regression in the Amsterdam data set with various amounts of LoD1.2 models. Similarly as observed in 4.3.5, accuracy increases with the amount of LoD1.2 models.

The classification accuracy is higher than it is observed in 4.3.3 and it is due to the removed LoD0 family. Interestingly, the prediction accuracy of the Logistic Regression in the 50% LoD1 Amsterdam data set (38.3%) is higher than the highest measured cross-validation accuracy in the synthetic data (33%). This shows the opposite behaviour as in 4.3.3, where the prediction accuracy in the Amsterdam data was orders of magnitudes lower than the cross-validation accuracy in the synthetic

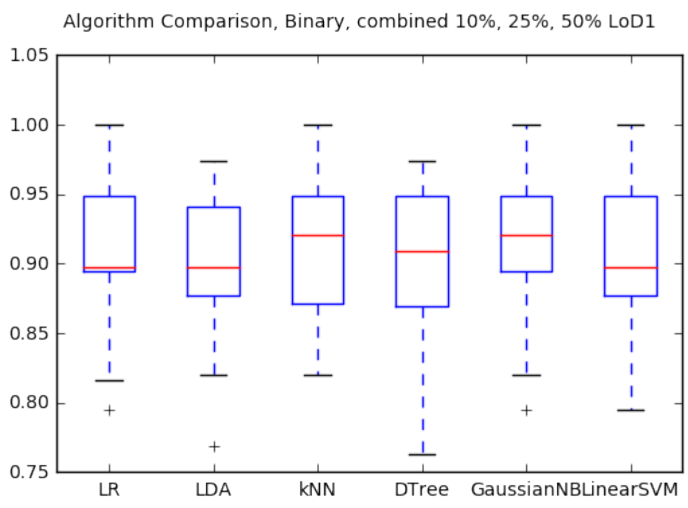


Figure 4.42: Binary classification accuracy, combined values.

Table 4.5: Classification accuracy of the Logistic Regression classifier

% of LoD1	Accuracy (%)
10	26.1
25	31.7
50	38.3

data in 4.3.2. However, even the highest measured accuracy, 38.3% is considered very low and shows that the current set up cannot be used to reliably classify the LoD.

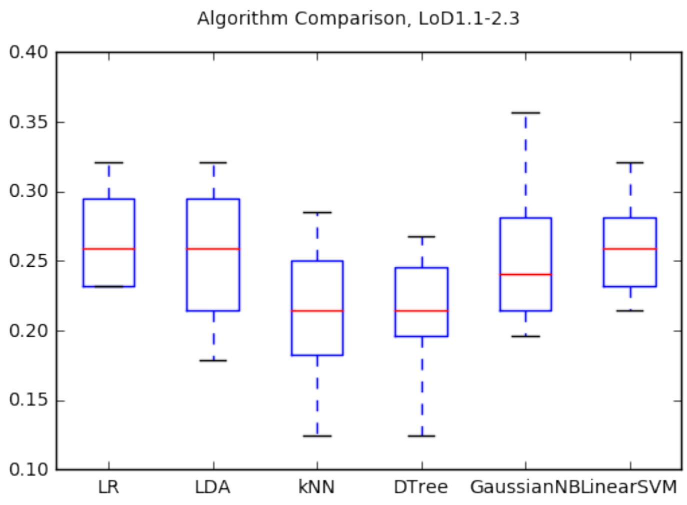


Figure 4.43: Cross-validation accuracy of classifiers in synthetic data.

5 CONCLUSIONS AND RECOMMENDATIONS

This thesis developed a method for automatically inferring the LoD in 3D city models. Testing shows that there are four main parameters that influence the LoD inference, *data set quality*, *feature quality*, *definition of the LoD classes* and *the classifiers*.

One of the main assumptions of the developed method is that a classifier is trained in a data set containing all LoD classes and then applied for the classification of arbitrary data sets. However, the differences in the classification performance in the synthetic and Amsterdam data, particularly as in Sections 4.3.3, 4.3.6, show that training a classifier on a data set with simple object shapes (Figure A.8) is not suitable to infer the LoD in a data set with geometrically more complex objects (Figure 4.22). Therefore the composition of the design set is a main limiting factor in the developed method, because the design set must be composed from objects that are comparable in their shape to those that are to be classified. For example a design set could be compiled from building models from various geographical locations and architectural styles in order to mitigate the bias towards a particular style or shape complexity.

This work explored 9 geometry measures (see Section 4.2) to infer the LoD, extended with the RMSE of the point cloud - mesh distances. In case of the synthetic city model, Figures 4.7, 4.11, 4.15, 4.17, provide an insight into the similarity in the shape of the objects. This data set contains models that share the same geometry across several LoD. As there is no reference data available, these models cannot be differentiated in their LoD, purely based on their shape.

The experiments did not provide conclusive evidence that the current LoD classification [Gröger et al., 2012], [Biljecki et al., 2016b] would hinder the automatic LoD *inference*. The detailed requirements by Biljecki et al. [2016b] are relatively straightforward to translate to features (see Table 3.1). However, as illustrated in Figure 2.3, 2.4 and 2.5, the proposed *absolute 3D point accuracy* requirement is not sufficient for determining the *validity* of the LoD class.

The synthetic data set simulates the scenario when many buildings have a simple shape and thus their models have the same geometry in multiple LoD. Sections 4.3.1, 4.3.2 and 4.3.6 show that the classifiers missed the correct label in more than half of the cases. The highest reported accuracy is 42.5%. Comparing Section 4.3.3 to 4.3.6 shows that removing the LoD0 family, that is not present in the Amsterdam data, improves the classification accuracy significantly.

The class frequencies in the Amsterdam data set suggest that real-world data sets can be highly imbalanced in their LoD, where the majority of the models having a valid LoD. Class imbalance proved to be one of the main reasons for low accuracy. Even though 4.3.4 achieves a relatively high accuracy in binary classification, it due to the low number of non-LoD2 objects in the data, as it is hinted by 4.3. 4.3.5 and 4.3.6 proves that the amount of class imbalance in the target data set decreases the prediction accuracy by orders of magnitudes.

Using a learning algorithm for LoD validation poses the challenge of uncertainty. In the binary classification cases (in 4.3.4, 4.3.5) it was tested that how well can a classifier distinguish between valid LoD models (LoD2) and invalid LoD models (not LoD2) in an LoD2 real-world data set. Even if an algorithm can accurately distinguish between valid and invalid in $> 92\%$ of the models in unseen data, it is not known which objects are those that belong to the 8% misclassified. Therefore the question of uncertainty in LoD validation requires further exploration. Reducing the classification into a binary case improves the accuracy, which is favourable in applications that seek to validate whether the data set conforms the stated LoD.

However, as described in Section 2.2, LoD inference in itself is not sufficient to determine the validity of the LoD and LoD validation is required to assess the quality of the city model. This thesis made an attempt to determine the validity of the LoD by using a point cloud as a reference data set, but it did not provide reliable results. The computed point cloud - mesh distances contain a large amount of noise, outliers, in many occasions due to overhanging vegetation (see 4.2.2). While working on the thesis, I did not find a reliable method to remove the vegetation from the AHN2 point cloud. However, using a point cloud with accurately classified vegetation (e.g.AHN3) would resolve this issue. Secondly, the presented method for comparing a building model to its point cloud might be overly simplistic, and the difference comprised into a single value (RMSE) is too coarse for differentiating between multiple LoD, even when related to other measures (see Figure 4.21). Therefore 4.2.2 and the explored distance computation method cannot be considered a reliable method for supporting the LoD inference.

Although this focused on the LoD in 3D city models, the explored method could be applicable for inferring the scale in 2D objects. Similarly Touya and Brando-Escobar [2013] and Reimer et al. [2014], the 2D features in this thesis can be used to determine the shape complexity of building footprints. Such method could be used for example to identify areas that are mapped in a coarse level of detail in OpenStreetMap.

For future work it is advisable to break down the problem into at least two groups, as LoD inference and validation are two distinct problems. Firstly, a mainly machine-learning based approach that could follow the outline of what is presented in Section 3.1. It requires the preparation of the appropriate amount and quality of 3D city models, in which all LoD classes are present, the models are sampled from various geographical locations and city types, and labelling and validating the models. In the feature generation and evaluation step also those geometry descriptors could be tested that are not directly related to the LoD requirements (see 2.4). Evaluating the features along with the classifiers (e.g.Wurm et al. [2016]) could help in pinpointing potential sources of errors and also the best performing features and classifiers. Secondly, the LoD validation in relation to the LoD classes requires a more thorough attention. In this work a point cloud was used as reference data set, but others could also work, for example a Digital Surface Model, or a combination of several data sets (e.g.high resolution footprints plus other attributes). As described in Section 2.2, LoD validation relates a city model to its real-world counterpart, therefore it should be integral in any complete LoD quality assessment. However, determining the validity of the LoD can be ambiguous, for example in case of Figure 2.5. It is for future research to assess whether such cases should be simply regarded in/valid, or provide a range a of accuracy? Also, following the idea of Biljecki et al. [2018], it is an open question whether it is worth the investment to accurately model the roof borderline cases (e.g.for 2.5), or the required effort does not justify the costs.

A ADDITIONAL TABLES AND FIGURES

Table A.1: Classification accuracy on the synthetic data, raw features

Classifier	Mean accuracy	Standard deviation
LR:	0.370000	0.031225
LDA:	0.245000	0.039211
kNN:	0.318750	0.054558
DTree:	0.350000	0.042573
GaussianNB:	0.397500	0.058843
LinearSVM:	0.340000	0.039449

Table A.2: Classification accuracy on the Amsterdam data, raw features

Classification	Mean accuracy	Standard deviation
LR:	0.877868	0.048930
LDA:	0.870175	0.040110
kNN:	0.896221	0.052898
DTree:	0.896289	0.036141
GaussianNB:	0.340486	0.111045
LinearSVM:	0.849325	0.047294

Table A.3: Classification accuracy on the synthetic data, standardized features

Classifier	Mean accuracy	Standard deviation
LR:	0.377500	0.042500
LDA:	0.245000	0.039211
kNN:	0.346250	0.049069
DTree:	0.356250	0.027528
GaussianNB:	0.398750	0.058750
LinearSVM:	0.377500	0.041382

Table A.4: Classification accuracy on the Amsterdam data, standardized features

Classifier	Mean accuracy	Standard deviation
LR:	0.872605	0.053614
LDA:	0.870175	0.040110
kNN:	0.901484	0.054115
DTree:	0.901552	0.035527
GaussianNB:	0.228475	0.106108
LinearSVM:	0.864777	0.050395

Table A.5: Binary classification accuracy on the Amsterdam data, standardized features.

Classifier	Mean accuracy	Standard deviation
LR:	0.890823	0.040365
LDA:	0.893455	0.043021
kNN:	0.922132	0.042030
DTree:	0.932591	0.038677
GaussianNB:	0.882996	0.041367
LinearSVM:	0.893387	0.038203

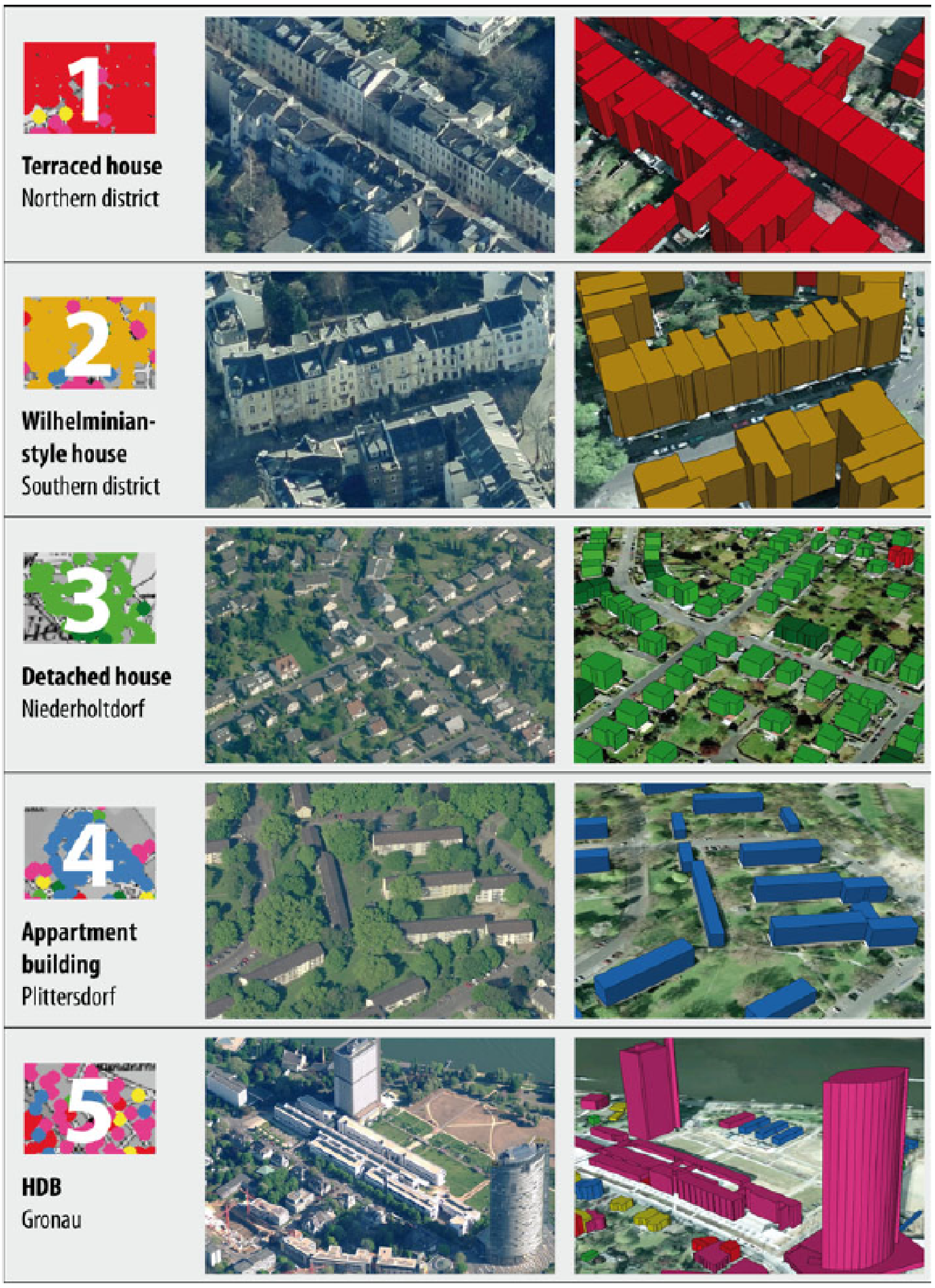


Figure A.1: Examples of building-type prediction in five exemplary districts of Bonn, Germany; source of left column: Microsoft Bing, of the aerial image in right column: Google Earth. Henn et al. [2012]

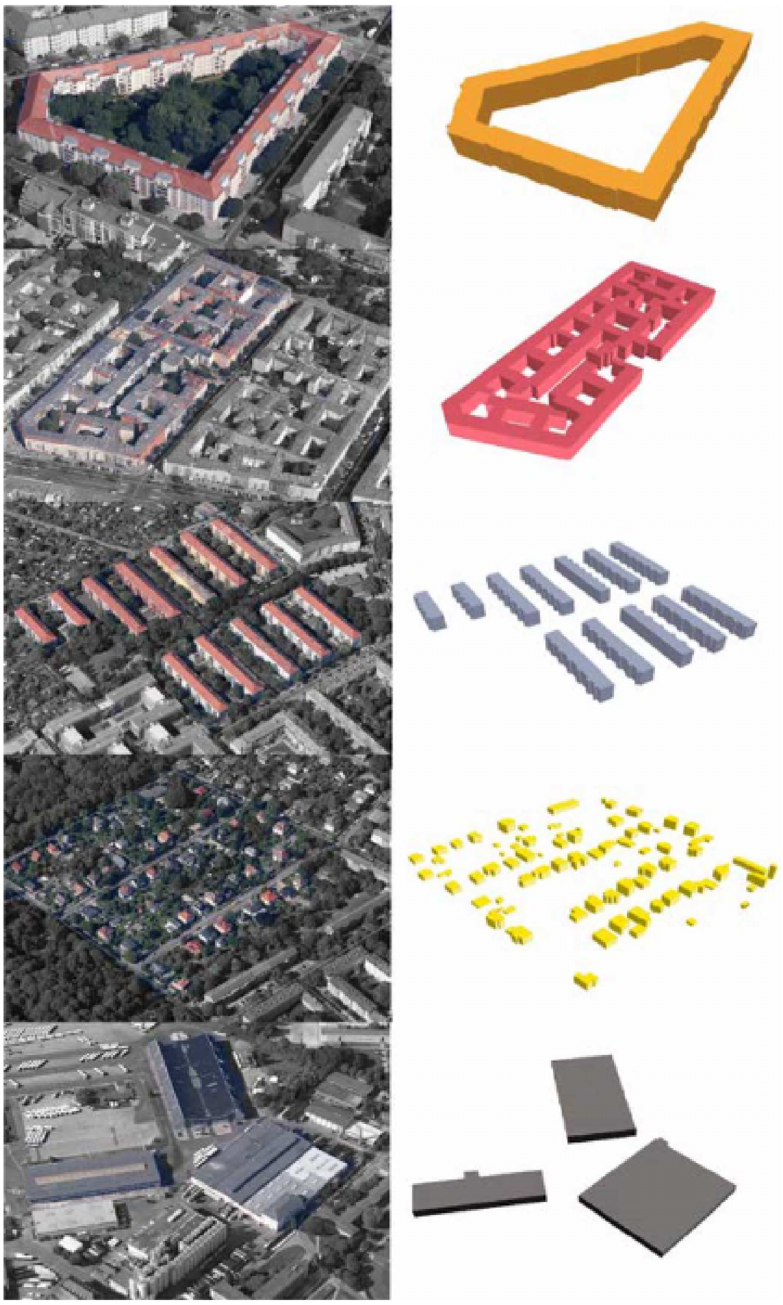


Figure A.2: Building types as viewed from aerial imagery (left; Google Earth) and as they are represented in the building model (right) from top to bottom: perimeter block development; block development; terraced houses/row houses; detached/semidetached; halls. Wurm et al. [2016]

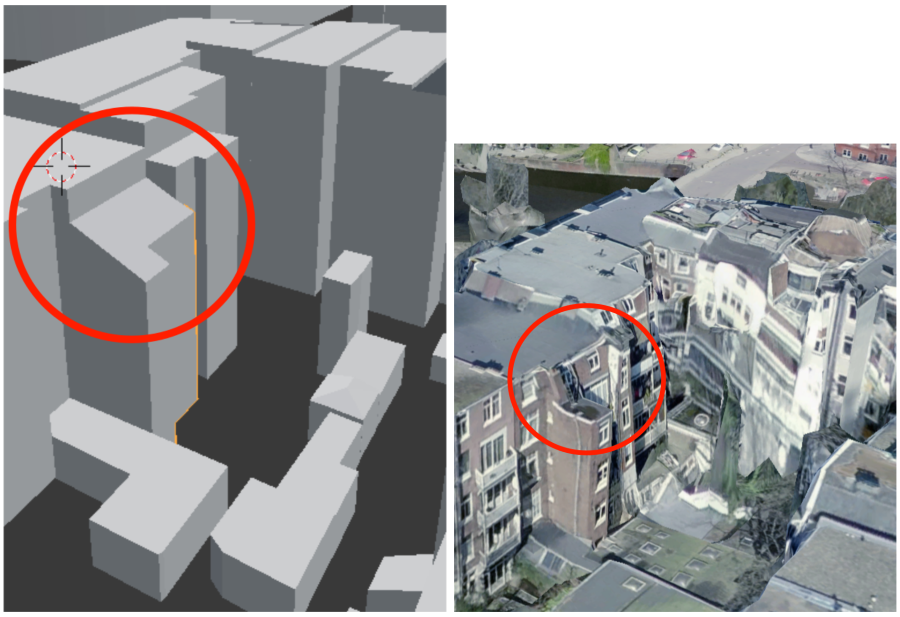


Figure A.3: Building model in Amsterdam city center in LoD2, where the terrace was reconstructed with the wrong roof type (red circle). Source: Google Earth for the aerial image; VirtualCitySystems for the 3D model

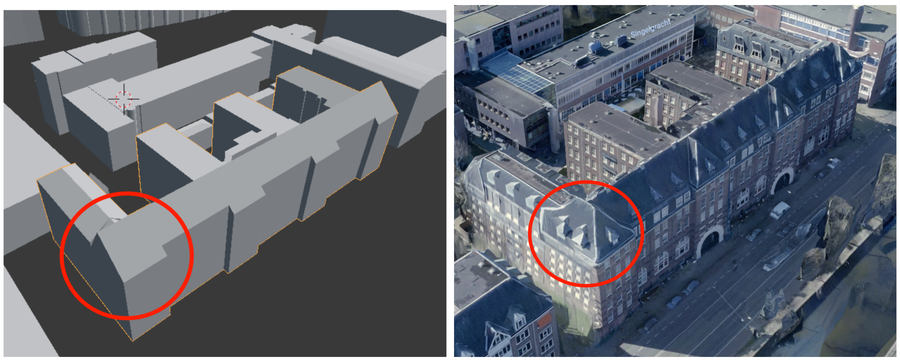


Figure A.4: Building model in Amsterdam city center in LoD2, where the corner of the roof was reconstructed with the wrong roof type (red circle). Source: Google Earth for the aerial image; VirtualCitySystems for the 3D model

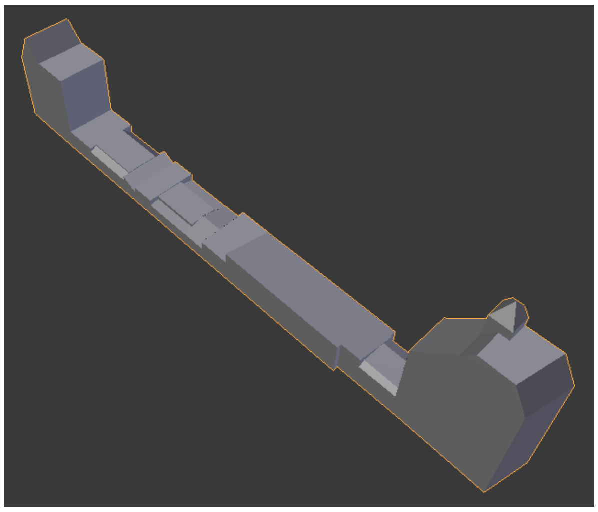


Figure A.5: Building model `BID_363100012130645`

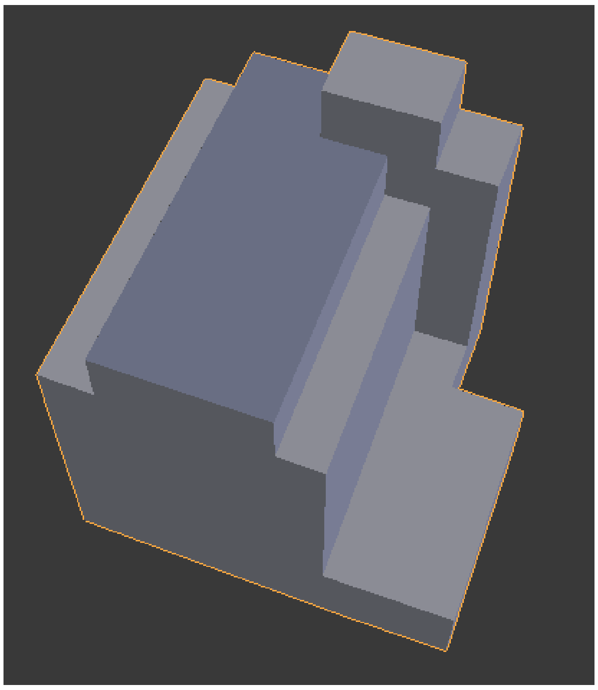


Figure A.6: Building model `BID_363100012168777`

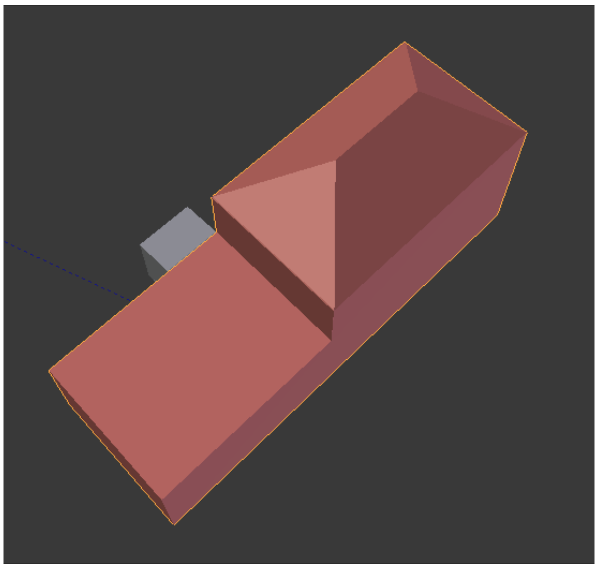


Figure A.7: Building model `BID_363100012169940`

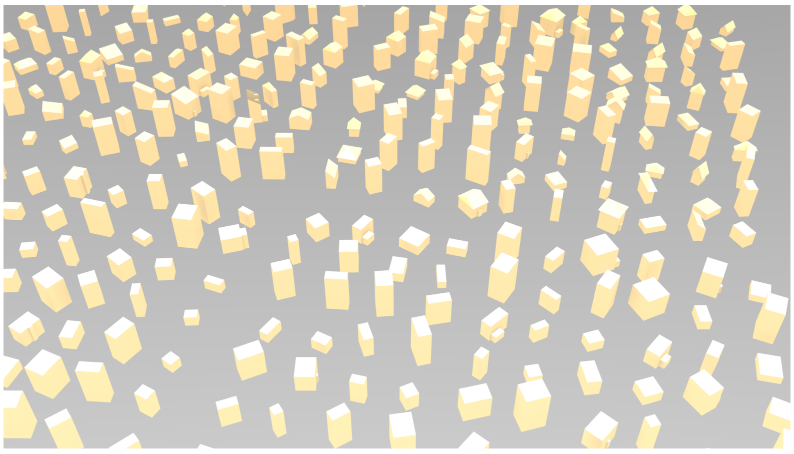


Figure A.8: A close-up of the synthetic data set generated with Random3DCity

B REFLECTION

It was important to put the systems in place among the first things. This included a clear work-flow for reading and writing data, taking it apart and putting it back together (e.g.building components), visualize it, document the process. In theory the better these systems work, the more time one has later to actually work on the research problem, instead of just hacking away.

Having a starting data set in important. Finding the appropriate city model is a daunting and very risky task, because MSc students probably have a very limited knowledge, experience with city models. Thus they probably don't really know what to look for and they can choose an unsuitable data set.

FME is a good tool for geometry processing, and developing own tools might be less beneficial. I wasted weeks on this, practically in vain as it didn't contribute to my research.

As a starting point, the manual validation of a city model an invaluable excercise. This should take a day or two, but its an invaluable exercise to get to know a lot about city models. I think its more effective than just browsing a model, because the LoD specifications provide a structure for what to look for. When the student is just beginning to learn about CityGML in depth, they might not know what to consider when just browsing a model. OBJ + Blender is their friend in this as in Blender they can measure in 3D, in FME they cannot.

The results of this work can benefit professionals who need to develop a validation tool for the LoD in a city model. Therefore a method that detects inconsistent LoD in a data set could warn the user that the accuracy of the spatial analysis will be possibly reduced. Furthermore, price of the 3D city models increases with the LoD and the current automatic or semi-automatic building reconstruction methods are still prone to errors. Therefore it is in the interest of the data buyer to validate that LoD of the purchased data set is homogeneous and according to its documentation. Manual validation of large city models might not be feasible. However, such a method will probably be less useful for data sets that consists of only a single building, very few buildings or buildings with homogeneous LoD. In these cases it is likely to be more efficient to manually observe and assess the LoD.

BIBLIOGRAPHY

(2016). European Public Tender: Digital 3D Building Model of Rotterdam. https://www.gim-international.com/content/news/european-public-tender-digital-3d-building-model-of-rotterdam. Accessed: 2018-01-10.

(2016). Tender of Rotterdam 3D. https://www.tenderned.nl/tenderned-tap/aankondigingen/86797. Accessed: 2018-01-10.

(2017). OpenStreetMap Wiki - 3D buildings roof modeling guide. https://wiki.openstreetmap.org/wiki/Simple_3D_buildings#Roof. Accessed: 2018-01-10.

(2018). Wavefront OBJ. https://en.wikipedia.org/wiki/Wavefront_.obj_file. Accessed: 2018-01-11.

(n.d). CityDoctor homepage. http://www.citydoctor.eu/. Accessed: 2018-01-10.

(n.d). CityGML homepage - 3D city models. https://www.citygml.org/3dcities/. Accessed: 2018-01-10.

(n.d.). Rotterdam consortium wins EU tender of 17.7 million for Smart City applications. https://en.rotterdampartners.nl/news/rotterdam-consortium-wins-eu-tender-of-17-7-million-for-smart-city-applications/. Accessed: 2018-01-10.

(n.d). virtualcitySystems - Building reconstruction. http://www.virtualcitysystems.de/en/products/buildingreconstruction. Accessed: 2018-01-10.

Ankerst, M., Kastenmüller, G., Kriegel, H.-P., and Seidl, T. (1999). *3D Shape Histograms for Similarity Search and Classification in Spatial Databases*, pages 207–226. Springer Berlin Heidelberg, Berlin, Heidelberg.

Biljecki, F. (2017). *Level of detail in 3D city models*. phdthesis, Delft University of Technology.

Biljecki, F. and Arroyo Ohori, K. (2015). Automatic Semantic-preserving Conversion Between OBJ and CityGML. In *Eurographics Workshop on Urban Data Modelling and Visualisation 2015*, pages 25–30, Delft, Netherlands.

Biljecki, F., Heuvelink, G. B., Ledoux, H., and Stoter, J. (2018). The effect of acquisition error and level of detail on the accuracy of spatial analyses. *Cartography and Geographic Information Science*, 45(2):156–176.

Biljecki, F., Ledoux, H., and Stoter, J. (2016a). Generation of multi-LOD 3D city models in CityGML with the procedural modelling engine Random3Dcity. *ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci.*, pages 51–59.

Biljecki, F., Ledoux, H., and Stoter, J. (2016b). An improved LOD specification for 3d building models. *Computers, Environment and Urban Systems*, 59:25–37.

Biljecki, F., Ledoux, H., Stoter, J., and Vosselman, G. (2016c). The variants of an LOD of a 3d building model and their influence on spatial analyses. *ISPRS Journal of Photogrammetry and Remote Sensing*, 116:42–54.

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

Biljecki, F., Zhao, J., Stoter, J. E., and Ledoux, H. (2013). Revisiting the concept level of detail in 3d city modelling. In *8th 3DGeoInfo Conference & WG II/2 Workshop, Istanbul, Turkey, 27–29 November 2013, ISPRS Archives Volume II-2/W1*. ISPRS.

Bishop, C. M. (2006). *Pattern recognition and Machine learning*. Information science and statistics. Springer, New York.

Boeters, R., Ohori, K. A., Biljecki, F., and Zlatanova, S. (2015). Automatically enhancing citygml lod2 models with a corresponding indoor geometry. *International Journal of Geographical Information Science*, 29(12):2248–2268.

Chapelle, O., Scholkopf, B., and Zien, A. (2006). *Semi-Supervised Learning (Adaptive Computation and Machine Learning series)*. The MIT Press.

Cignoni, P., Rocchini, C., and Scopigno, R. (1998). Metro: Measuring Error on Simplified Surfaces. *Computer Graphics Forum*, 17(2):167–174. bibtex: CGF:CGF236.

Clark, J. H. (1976). Hierarchical geometric models for visible surface algorithms. *Commun. ACM*, 19(10):547–554.

Costa, L. d. F., Cesar, R. M., and Costa, L. d. F. (2009). *Shape classification and analysis theory and practice*. CRC Press, Boca Raton. OCLC: 929288063.

Duin, R. and Pekalska, E. (2015). *Pattern Recognition: Introduction and Terminology*. 37 Steps.

Gröger, G., Kolbe, T. H., Nagel, C., and Häfele, K.-H. (2012). OGC City Geography Markup Language (CityGML) Encoding Standard, Version 2.0. OpenGIS Implementation Specification OGC 12-019, Open Geospatial Consortium.

Gröger, G. and Plümer, L. (2012). CityGML – Interoperable semantic 3d city models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 71:12–33.

Haala, N. and Kada, M. (2010). An update on automatic 3d building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6):570–580. ISPRS Centenary Celebration Issue.

Han, J. and Kamber, M. (2011). *Data mining: concepts and techniques*. Elsevier, Burlington, MA, 3rd ed edition.

Henn, A., Römer, C., Gröger, G., and Plümer, L. (2012). Automatic classification of building types in 3d city models: Using SVMs for semantic enrichment of low resolution building data. *GeoInformatica*, 16(2):281–306.

Kazmi, I. K., You, L., and Zhang, J. J. (2013). A survey of 2d and 3d shape descriptors. In *2013 10th International Conference Computer Graphics, Imaging and Visualization*, pages 1–10.

Kolbe, T. H. (2009). Representing and exchanging 3d city models with CityGML. In *3D geo-information sciences*, pages 15–31. Springer.

Kresse, W. and Danko, D. M., editors (2012). *Springer handbook of geographic information*. Springer, Berlin ; New York. OCLC: ocn795353459.

Ledoux, H. (2013). On the Validation of Solids Represented with the International Standards for Geographic Information: On the validation of solids represented with the international standards. *Computer-Aided Civil and Infrastructure Engineering*, 28(9):693–706.

Löwner, M.-O., Gröger, G., Benner, J., Biljecki, F., and Nagel, C. (2016). PROPOSAL FOR A NEW LOD AND MULTI-REPRESENTATION CONCEPT FOR CITYGML. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W1:3–12.

Moser, D., Zechmeister, H. G., Plutzar, C., Sauberer, N., Wrbka, T., and Grabherr, G. (2002). Landscape patch shape complexity as an effective measure for plant species richness in rural landscapes. *Landscape Ecology*, 17(7):657–669.

Müller, A. C. (2016). *Introduction to machine learning with Python: a guide for data scientists*. O'Reilly Media, Inc, Beijing ; Boston, first edition edition.

Mäntylä, M. (1987). *An Introduction to Solid Modeling*. Computer Science Press, Inc., New York, NY, USA. bibtex: Mantyla:1987:ISM:39278.

Osada, R., Funkhouser, T., Chazelle, B., and Dobkin, D. (2001). Matching 3d models with shape distributions. In *Proceedings International Conference on Shape Modeling and Applications*, pages 154–166.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., and Duchesnay, E. (2011). Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830. bibtex: scikit-learn.

Redweik, R. and Becker, T. (2015). Change Detection in CityGML Documents. In Breunig, M., Al-Doori, M., Butwilowski, E., Kuper, P. V., Benner, J., and Haefele, K. H., editors, *3D Geoinformation Science*, Lecture Notes in Geoinformation and Cartography, pages 107–121. Springer, Cham.

Reimer, A., Kempf, C., Rylov, M., and Neis, P. (2014). Assigning scale equivalencies to OpenStreetMap polygons. In *Proceedings of AutoCarto international symposium on automated cartography*. bibtex: reimer2014assigning.

Theodoridis, S. and Koutroumbas, K. (2009). *Pattern recognition*. Elsevier, Amsterdam, 4. ed edition. OCLC: 550588366.

Touya, G. and Brando-Escobar, C. (2013). Detecting Level-of-Detail Inconsistencies in Volunteered Geographic Information Data Sets. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 48(2):134–143.

Touya, G. and Reimer, A. (2015). Inferring the Scale of OpenStreetMap Features. In Jokar Arsanjani, J., Zipf, A., Mooney, P., and Helbich, M., editors, *OpenStreetMap in GIScience*, pages 81–99. Springer, Cham.

Van Renterghem, T. and Botteldooren, D. (2010). The importance of roof shape for road traffic noise shielding in the urban environment. *Journal of Sound and Vibration*, 329(9):1422–1434.

Wagner, D. and Ledoux, H. (2016). Ogc citygml quality interoperability experiment. techreport OGC 16-064r1, Open Geospatial Consortium.

Wieland, M., Nichersu, A., Murshed, S. M., and Wendel, J. (2015). Computing solar radiation on CityGML building data. In *18th AGILE international conference on geographic informaton science*.

Wong, K. and Ellul, C. (2016). Using Geometry-Based Metrics as Part of Fitness-for-Purpose Evaluations of 3d City Models. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W1:129–136.

Wurm, M., Schmitt, A., and Taubenbock, H. (2016). Building Types' Classification Using Shape-Based Features and Linear Discriminant Functions. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 9(5):1901–1912.

Yang, M., Kpalma, K., and Ronsin, J. (2008). A Survey of Shape Feature Extraction Techniques. In Yin, P.-Y., editor, *Pattern Recognition*, pages 43–90. IN-TECH. 38 pages.

Zhang, D. and Lu, G. (2004). Review of shape representation and description techniques. *Pattern Recognition*, 37(1):1–19.

COLOPHON

This document was typeset using LaTeX. The document layout was generated using the `arsclassica` package by Lorenzo Pantieri, which is an adaption of the original `classicthesis` package from André Miede.