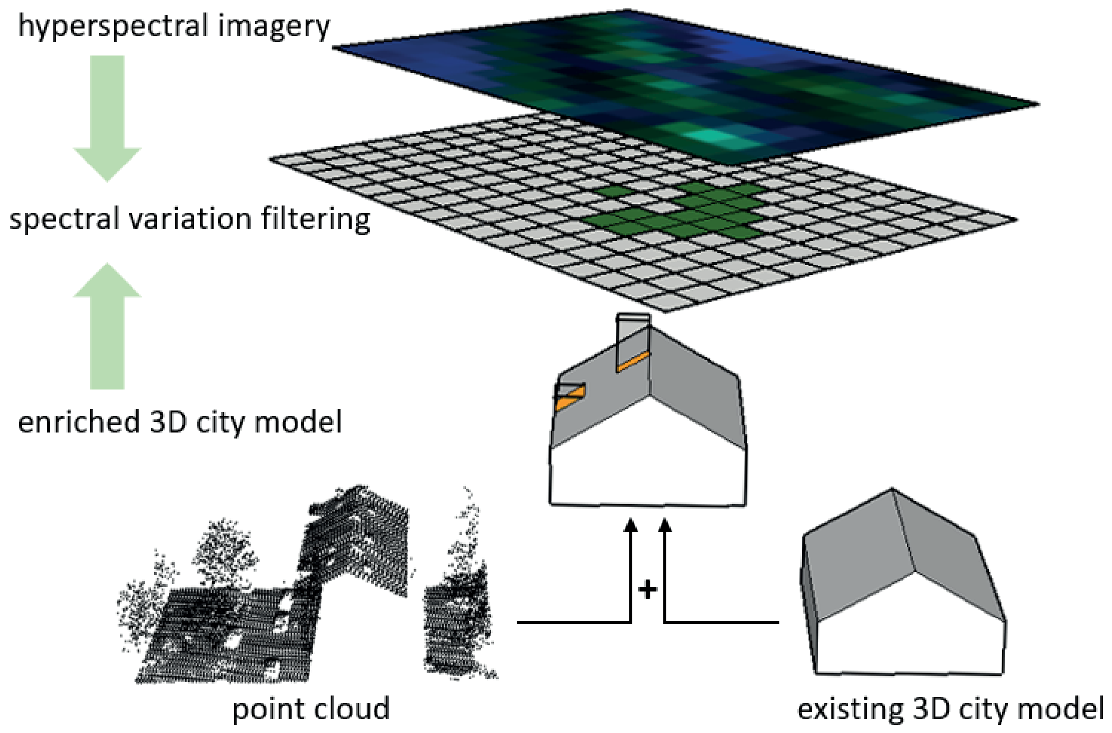
MSc thesis in Geomatics for the Built Environment

3D City Models in the Context of Urban Mining

A case study based on the CityGML model of Rotterdam

Pablo Antonio Ruben

2019



3D CITY MODELS IN THE CONTEXT OF URBAN MINING
A CASE STUDY BASED ON THE CITYGML MODEL OF ROTTERDAM

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

Pablo Antonio Ruben

July 2019

ISBN 978-94-6366-193-5

The work in this thesis was made in the:

H2020 REPAiR Project (REsource Management in Peri-urban Areas)
Department of Urbanism
Faculty of Architecture & the Built Environment
Delft University of Technology

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

In memory of Gitta Scheenhouwer. We will always remember and miss you.

ABSTRACT

Recently, the application of machine learning and data fusion techniques on hyperspectral imagery have demonstrated potential for ground cover classification at material level. Hereby, specific locations of resources enclosed in cities (e.g. roof materials) can be identified, which is critically relevant within the field of urban mining.

A limitation of this approach is the so-called *pepper and salt effect*, the oversensitivity of the classifiers to spectral variations within a pixel (e.g. chimneys, roof windows). Identifying and correcting affected pixels can be done statistically (e.g. using a majority filter), but not in cases where spectral variations affect a majority of pixels characterizing a surface.

A solution to this limitation would be the usage of 3D city models containing the objects inducing the spectral variations. However, such highly detailed 3D city models are often unavailable as they cannot be produced automatically yet.

An alternative covered by this research is to use a less detailed 3D city model and semantically enrich it with the required data. As 3D city models are usually produced using a point cloud, such a point cloud is used to perform the enrichment. The main research question addressed is therefore: *How can a CityGML LOD2 model be semantically enriched in order to improve material classification performed on roof surfaces?*.

To address this, an existing LOD2 model was compared to a point cloud acquired by Ligth Detecation and Ranging and 'deviation' points were identified. This identification uses a distance check for seed selection and performs a region growing with an orientation check. In a subsequent step, 'deviation' point regions were translated into a geometric shape by usage of their Voronoi diagram and fused with the pixels of hyperspectral imagery.

Part of this research is also a nominal validation analyzing a total of 41 buildings and 831 pixels located in the south of Rotterdam (Netherlands). Overall kappa values of up to 0.7 and commission errors as low as 10% (for the class 'clean' pixels) were obtained, showing potential of the chosen method. Additionally, a rational validation was performed to assess the impact of potential tolerance of classifiers for 'spectral deviations'. This one only included 10 buildings, but took into account 328 pixels located up to 30% outside the roof surface

A main outcome is the recommendation on settings to use depending on the specific user needs. To accurately quantify materials, relatively 'loose' settings are recommended. In contrast, to identify presence of materials, stricter settings are recommended. Beyond this, recommendations to data suppliers and potential applications of the method to other fields are formulated.

ACKNOWLEDGEMENTS

First of all, I would like to thank my tutors for the critical but always kind and constructive mentoring during these 9-month graduation process. While the student obviously takes the lead of the thesis research, I am now more than ever convinced that the context matters a lot too. Rusnė and Giorgio, I would like to thank you for having made this thesis possible. Your interpersonal skills are highly appreciated, please continue taking care of students this way!

The world of geomatics would not be what it is without the generous, open user communities supporting it. This was also the case here and, beyond the online communities many of us use, I would like to thank some people more specifically:

- the Rotterdam 3D department (especially Christian Wisse) for their kind interest and support, sharing with me a high-density point cloud and a number of technical specification documents. And also for giving me the opportunity to present and discuss my work at their offices in March and July 2019.

- the team of the Airborne Prism Experiment, especially Andreas Hueni, and Dominik Brunner from the Swiss Federal Laboratory for Material Science; for granting me access to the data acquired during the 2014 flight above Rotterdam.

- geoforum.nl and especially Ingrid Alkemade and Rinaldo Bhailal for sharing the specifications of aerial images and the AHN point cloud.

- last but not least, I would like to thank my co-reader, Kaixuan Zhou who thoroughly and critically read my document for the second to last assessment.

A big thank you also goes to my second half, Camille for always listening and motivating me, even when it got very technical. The advices from your recent graduation were great! I also want to thank my parents and my little brother Shura for their curiosity and supporting words. Thanks for your unfailing support during these years of studies - And I look forward to many more inspiring discussions!

CONTENTS

LIST OF FIGURES

LIST OF TABLES

List of Algorithms

ACRONYMS

3DCityDB	Application Domain Extension	35
ADE	Application Domain Extension	22
AHN	Algemeen Hoogtebestand Nederland - General height dataset of the Netherlands	23
APEX	Airborne Prism EXperiment	xvii
BAG	Basisregistratie Adressen en Gebouwen - Basis Registration Addresses and Buildings	26
BGT	Basisregistratie Grootschalige Topografie - Basis registration high scale topography	26
CCD	Charge Coupled Device	61
CityGML	City Geographic Markup Language	xi
CMOS	Complementary Metal-Oxide-Semiconductor	61
csv	comma separated values	57
FDN	fixed-distance neighbors	20
FME	Feature Manipulation Engine	15
GDAL	Geospatial Data Abstraction Library	67
GPS	Global Positioning System	24
kd-tree	k-dimensional tree	20
khat	an estimate of cohen's kappa	76
KNN	k-nearest neighbors	xiv
LiDAR	Light Detection and Ranging	xi
LOD	Level of detail	xii
PCA	Principal Components Analysis	xiv
QGIS	Quantum Geographic Information System	15
SWIR	Short-Wave InfraRed	xvii
SQL	Structured Querying Language	35
VNIR	Visible and Near InfraRed	xvii
WKT	Well known text format	35

1 INTRODUCTION

This document describes the graduation research on semantic enrichment of digital three-dimensional city models (such as CityGML LOD2 datasets) by means of point cloud analysis. On top of geometry and sometimes textures, a semantic 3D city model includes other, more thematic knowledge (by making some features and properties explicit) (Billen et al., 2014). This research will be conducted with regard to a specific application, which is the improvement of material classification (i.e. hyperspectral imagery) performed on building roof surfaces. This improvement consists in using an enriched 3D city model to identify spectral variations.

This research is the final part of the Master of Science program *Geomatics for the Built Environment* at Delft University of Technology, the Netherlands. It starts with an introduction to the context, the societal relevance of urban mining (part 1.1). A second step introduces existing approaches to support urban mining (part 1.2) and their weaknesses. Based on this, the the problem statement (1.3) will be formulated along with relevant background knowledge (1.5). This first chapter is then closed by the research questions (part 1.4), scope (1.6) and reading guide (part 1.7).

1.1 SOCIETAL RELEVANCE AND CONTEXT: URBAN MINING FOR A MORE SUSTAINABLE WORLD

In order to reach global goals in greenhouse gas emission reduction, the building industry will inevitably be affected. A 2008 study conducted by the UK government has shown that no less than 47% of national CO2 emissions are the result of the building's life cycles (of which 83% result from the usage and 15% from the construction). This makes clear that on one hand, retrofitting - improving the existing building stock is required. On the other hand, the building process itself, the way this retrofitting and construction of new buildings use resources, will also have to change.

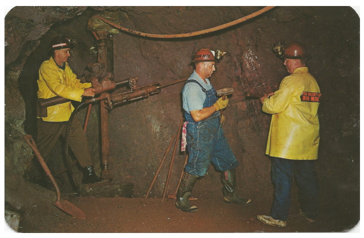


Figure 1.1: Mining in two different contexts. On the left, mineral mining - on the right, urban mining. [1]

One of the approaches to improve the resource consumption of construction activities is the one of circular economy and more specifically urban mining. The main idea is "the exploration and exploitation of material stocks in urban systems for anthropogenic [i.e. human] activities" (Baccini and Brunner, 2012). Instead of

1 Source left: flickr (https://www.flickr.com/photos/upnorthmemories/6092963634), source right: wikimedia

extracting resources from natural mines, urban mining looks for their presence in cities to reuse the resources that are already present locally.

One building element of particular interest on which this research will focus is the roof. Due to exposure to climate, it needs to be renewed at regular intervals which are often shorter than the ones of general refurbishments. Promising initiatives (e.g. *www.roof2roof.nl*) already exist and aim at recycling freed materials such as bitumen. This can be done by reconditioning for reusage on roofs or downcycling for road asphalt (Townsend et al., 2007).

In order to determine if such processes are environmentally beneficial, mapping of the resources is required. In fact, the recycling/downcycling does consume energy and creates flows (from or to treatment facilities) which impacts the environment. Therefore several situations including the initial status quo and alternatives need to be evaluated. Patouillard et al. (2018) discusses why aggregated approaches (such as statistical ones) can be insufficient to do so.

When doing such evaluations, the granularity of the data (the level of aggregation) deserves major attention. In fact, for some cases, data aggregated at a neighborhood level might be sufficient but for others, detailed object-level data will be necessary. This aspect is clearly formulated by Patouillard et al. (2018) and needs to be considered within all three steps of life cycle analysis as they impact each other linearly (also see figure 1.2):

Input data is required and needs to be obtained by analyzing either the existing objects or past processes (step 1). This material, which gets released by demolition/-maintenance would then be reused, preferably locally. To estimate the quantity and distribution of release, demolition/maintenance prediction is necessary (step 2). In the end and optimally, another material flow would be replaced, which raises the question of the impact of this change (step 3).

The mapping of resources enclosed in cities is thus important to develop accurate circularity strategies for the future. Additionally, not only thematic but also positional accuracy matters, especially for flows at a local scale.

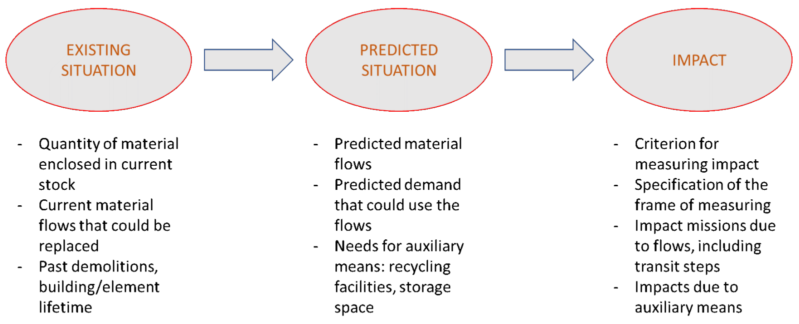

EXISTING SITUATION
PREDICTED SITUATION
IMPACT
- Quantity of material enclosed in current stock
- Current material flows that could be replaced
- Past demolitions, building/element lifetime
- Predicted material flows
- Predicted demand that could use the flows
- Needs for auxiliary means: recycling facilities, storage space
- Criterion for measuring impact
- Specification of the frame of measuring
- Impact missions due to flows, including transit steps
- Impacts due to auxiliary means

Figure 1.2: Division of spatial life-cycle analysis in three steps and corresponding tasks (own work).

1.2 SCIENTIFIC RELEVANCE: EXISTING APPROACHES TO URBAN MINING AND LIMITATIONS

As shown in the previous part (1.1), the mapping of enclosed building materials is a highly relevant task with regard to sustainability. In order to do this, mainly two approaches exist and will be discussed here: data disaggregation and remote sensing. The two approaches show fundamental differences but can also be used in complementarity to each other. In fact, any detailed data obtained from the second, more detailed one can also serve as an input for the first one.

1.2.1 Data disaggregation

A statistical, top-down approach is possible and involves disaggregating historic material usage data (e.g. national bitumen coating production) by using spatial indicators such as house types, their usage of specific materials at a given period and their respective spatial distribution. A good example is (Müller, 2006) who has performed such analysis for concrete in the Netherlands but without being more specific geographically.

A similar approach but at a smaller scale (8 and 11 km2) was conducted by Tanikawa and Hashimoto (2009). They used historic data to estimate the material inflow and linked that to historical building construction and demolition data. Moreover, this data was used to predict lifetime by the construction period and future demolitions/release of material flows (see figure 1.3 for an example result).

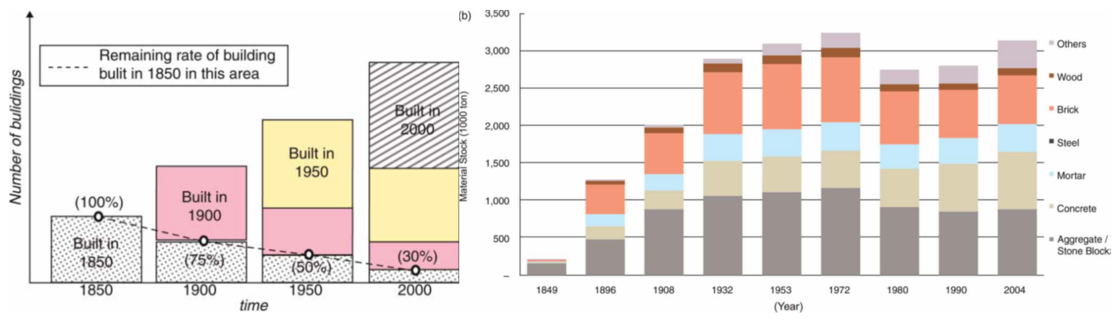


Figure 1.3: Accumulation and remaining rate of buildings (left) and material stock over time (right, Salford Quays - Manchester)(Tanikawa and Hashimoto, 2009).

The main shortcoming of the data disaggregation approach is that it is hardly possible to disaggregate up to object (e.g. building) level. For the specific requirements of some applications (e.g. if a municipality wants to send letters to owners of houses with a specific roof material), such an approach might thus not be suited.

1.2.2 Usage of remotely sensed imagery

A more specific approach is the usage of remote sensing data to identify materials using their spectral characteristics. The field of land cover classification is not new and can be divided into several levels of detail (figure 1.4), similarly to 3D city models (which will be discussed in part 1.5.1).

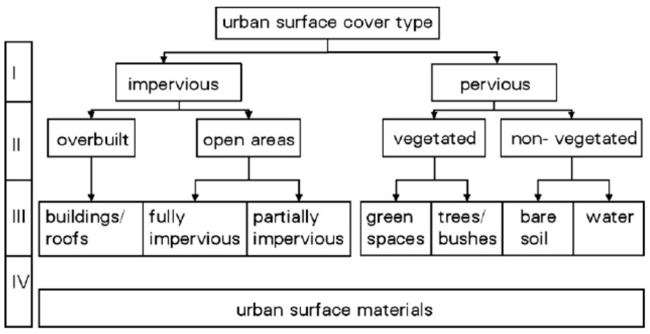


Figure 1.4: Hierarchical classification of urban surface materials (Heiden et al., 2007)

Numerous studies have been conducted for classification but level IV only became possible with higher resolution images. On one hand, the spatial resolution has considerably increased (Frick, 2007) and created opportunities for more detailed classification as fewer pixels are affected by spectral mixing. On the other hand, new technologies appeared for spectral classifiers: the range and the number of bands was increased by including numerous visual, VNIR and SWIR bands at identical, below 10m resolutions. In that way, a move from multispectral to so-called hyperspectral imagery took place. A higher number of spectral bands means that material spectra can be acquired with much more detail (more measures/points are available to plot the absorption curve), facilitating the classification at level IV. Often, the acquisition sensor is embarked on a manned aircraft, but other options using, for instance, unmanned airborne vehicles (UAV) also exist and have been extensively covered by Beth (2016).

In the past years, several research projects have demonstrated the ability of hyperspectral images to differentiate between ground cover materials. Heiden et al. (2007) conducted a study for Dresden and Potsdam using the HyMap sensor (see in figure 1.5). Results are good for some materials such as polyethylene where spectral feature identification resulted in commission and omission errors lower than 5%. Challenges nevertheless exist - especially for bituminous roof materials: "In contrast, some of the dark materials, such as asphalt and roofing tar paper do not show distinct absorption features. Their separation from other materials is only possible based on brightness."

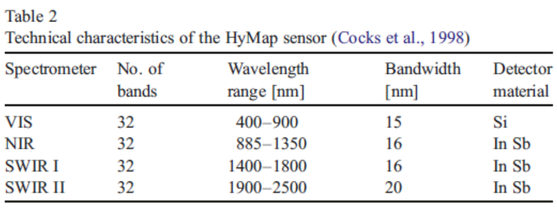


Table 2
Technical characteristics of the HyMap sensor (Cocks et al., 1998)

Spectrometer	No. of bands	Wavelength range [nm]	Bandwidth [nm]	Detector material
VIS	32	400–900	15	Si
NIR	32	885–1350	16	In Sb
SWIR I	32	1400–1800	16	In Sb
SWIR II	32	1900–2500	20	In Sb

Figure 1.5: Technical characteristics of the HyMap hyperspectral sensors (Heiden et al., 2007)

In a study on Brussels, Demarchi et al. (2014) compared different unsupervised machine learning algorithms with the aim to reduce the high dimensionality resulting from the high number of spectral bands. Machine learning was also used for the material classification. Results are positive, with overall kappa values as high as 0.82 for support vector machine classification.

Priem and Canters (2016) further build on this finding, by performing a fusion of the support vector classification result with auxiliary data. After classification, correction using LiDAR data is performed using height, roughness, and slope attributes. The result is convincing as the overall kappa was increased by 0.8 to 0.87 for sunlit and from 0.65 to 0.69 for shaded pixels (another difference with regard to the previous studies is that classification is performed on shaded pixels too). Depending on the material, high conditional kappa values can be obtained for sunlit (e.g. 0.93 for 'bitumen' and 1.00 for 'extensive green roofs') and shaded pixels (e.g. 0.94 for 'bitumen' and 0.84 for gray metal.

Next to auxiliary data, Priem and Canters (2016) introduced another important aspect. Heiden et al. (2007) and Demarchi et al. (2014) perform their studies (thus both training and validation) on a pixel-basis, which is correct from a scientific point of view but does not take into account what Priem and Canters (2016) names the *pepper and salt effect*.

In fact, hyperspectral imagery is oversensitive to spectral variations within pixels (e.g. the presence of chimneys, roof windows or solar panels) - even if they only cover a minor part of the pixel. To address this, Priem and Canters (2016) use a majority filter on a 3*3 kernel to correct the classification result. An illustration can be found in figures 1.6 and 1.7. A variant of this approach was developed by (Thiel, 2016) in a bachelor thesis where the majority filter is replaced by a moving window approach, including pixel's neighborhood information in the classification process directly.

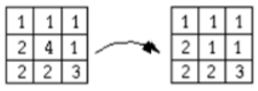


Figure 1.6: Illustration of the working of a majority filter (University of California Berkeley, ated).

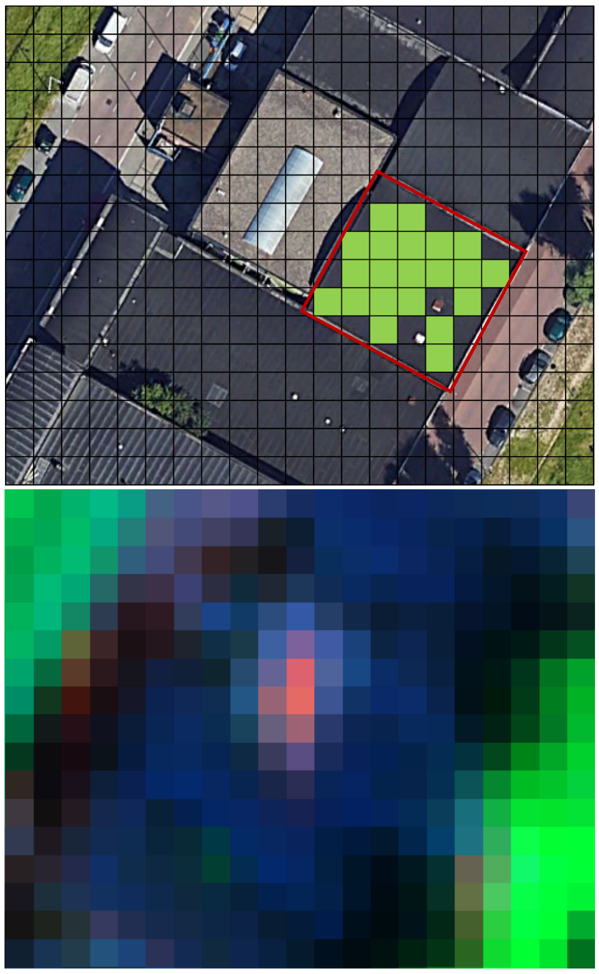


Figure 1.7: Top: Example of the application of the majority filter on a building with spectral deviations (two chimneys on the roof). In green the 'clean' pixels (4*4m) which are neither on the border of the footprint nor do contain a spectral deviation. Here, the majority filter can be used to filter out the pixels (fully or partially) containing a chimney. Imagery: (c) google earth. Bottom: false color image of the building as acquired by the APEX project (Red = 399-413 nm, Green = 1145-1155nm, Blue=2423-2432nm).

The limitation of this approach is that this solution only works if a critical share of the 8 neighbors is of the correct class. For buildings with a majority of pixels containing spectral variations, the majority filter is likely to favor the outcome of mixed pixels. An example of such case where the majority filter is unlikely to work can be found in figure 1.8. This one can be applied to all pixels contained in the footprint of the building and not containing any shadow (in brown). Applying a majority filter is unlikely to reduce the high share of pixels containing spectral variations (pixels containing roof windows), leaving the clean pixels in light green in minority.

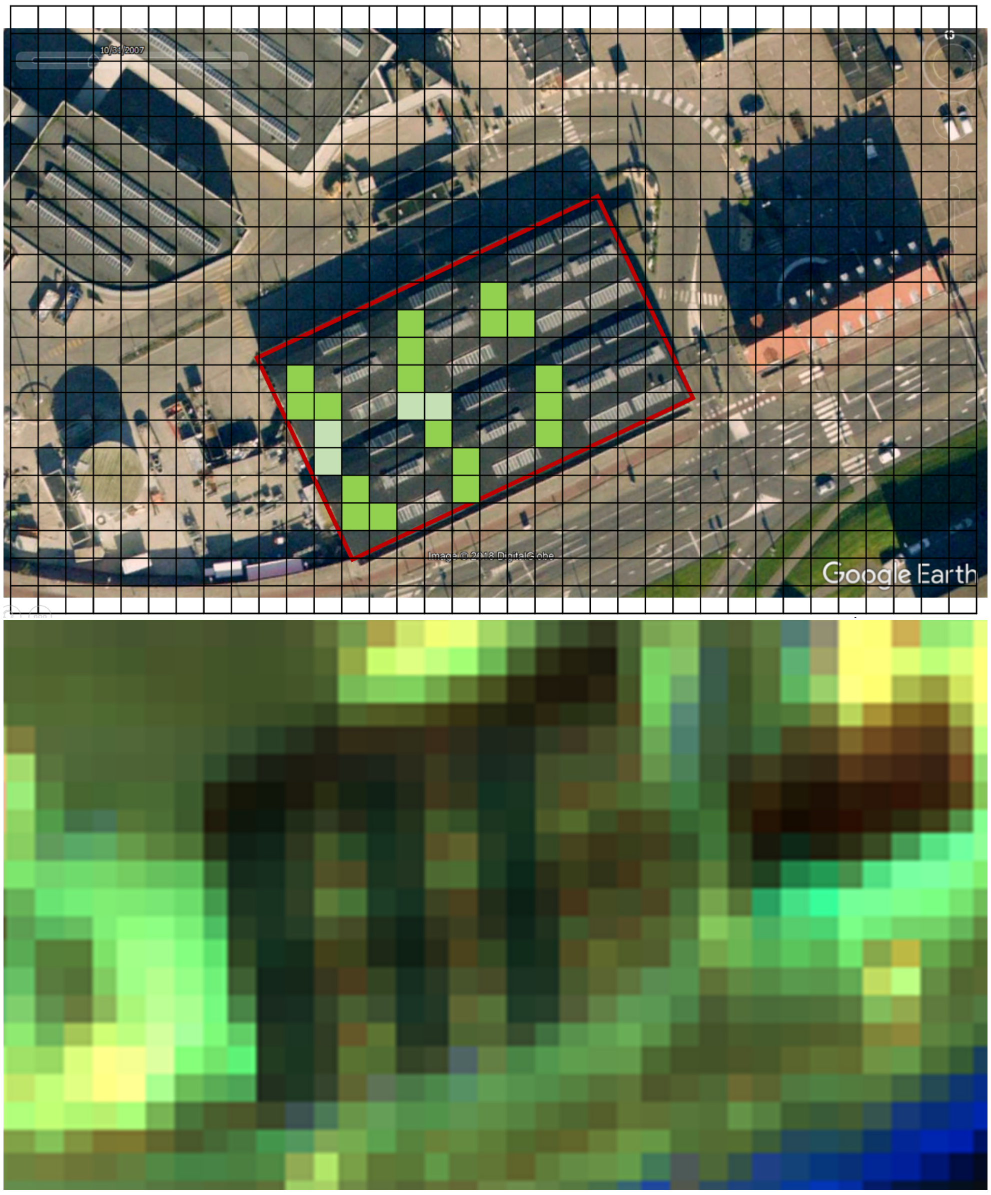


Figure 1.8: Top: Example of a building where the majority filter might not work due to the strong presence of spectral variations (roof windows). In dark green the 'clean' pixels (4*4m) that can be used. In light green the pixels that might be used but are affected by shadow. Here, the majority filter is unlikely to filter out the cells (fully or partially) containing the roof windows. Imagery: (c) google earth. Bottom: false-color image of the building as acquired by the APEX project (Red = 399-413 nm, Green = 1145-1155nm, Blue=2423-2432nm)

1.3 ADDRESSING THE LIMITATION AND PROBLEM STATEMENT

An approach to tackle the limitation of the majority filter (and, in a bigger context - of the resource mapping methods at building level) is the usage of a 3D city model. The advantage of using the additional information contained in the latter is twofold:

- first, detailed 3D city models might help with the identification of spectral variations by indicating the position of elements such as chimneys or roof windows. To do so, a city model fulfilling at least LOD3 standard (see part 1.5.1) is required.

- second, 3D city models have an additional dimension with regard to aerial imagery (which is only 2-dimensional). In fact, non-flat roof surfaces can improve the area indication stored in aerial images (the latter is, in fact, a projection). In order to perform this, a city model fulfilling at least LOD2 (see part 1.5.1) is required (the first level supporting sloped roof surfaces).

While the production of LOD2 models can be done highly automatically (Gemeente Rotterdam, n.d.), this is not the case for more detailed LOD3 models at the time of writing. Some critical elements such as chimneys, roof windows or ventilation boxes might thus be missing but strongly needed for reliable identification of spectral variations.

In a nutshell, the problem statement can be formulated as follows:

Automatic classification of roofs using aerial imagery requires pixels containing roof material deviations to be identified as such. While a highly detailed 3D city model would fulfill such criterion, this is often unavailable.

1.4 APPROACH TO ADDRESS THE PROBLEM AND RESEARCH QUESTIONS

A potential alternative on which no research has been conducted yet is to use a less detailed 3D city model and semantically enrich it with the required data. As 3D city models are usually produced using a point cloud (see parts 1.5.2 and also 3.1.1 for relevant background knowledge), this thesis will use such a point cloud to perform the enrichment.

The main research question is stated as follows: *How can a CityGML LOD2 model be semantically enriched in order to improve material classification performed on roof surfaces?*

This question can be structured as three sub-questions which will be addressed by this thesis:

1. Which method is suitable to identify 'deviations' of LiDAR point clouds compared to LOD2?

2. What are the requirements with regard to CityGML LOD2, LiDAR point clouds and hyperspectral imagery data?

3. To which extent does such a method support the identification of clean pixels?

1.5 BACKGROUND KNOWLEDGE

1.5.1 3D city models, the CityGML format and richness of detail

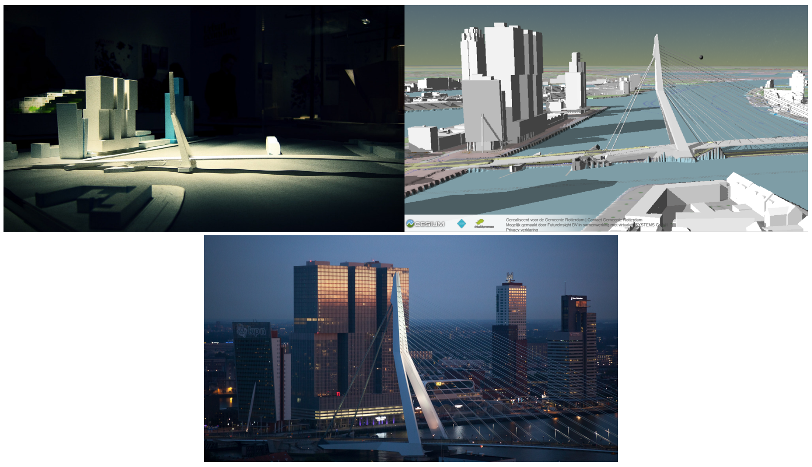


Figure 1.9: Both physical urban scale (top left) and a digital city model (top right) are abstractions of the reality (bottom). The main difference lies in the storage of 'meaning', i.e. of semantics. [2]

Just as for physical urban scale models, one can notice that digital 3D city models are always a generalization of reality, too. In fact, making a 1:1 representation would be very labor-intensive, if not impossible. Therefore, choices - often referred to as generalization - must be made (see figure 1.9). In fact, depending on the application, not every detail might be relevant and excessive work can often be avoided by defining needs beforehand.

In contrast, a major difference between physical and digital 3D city models is the support of semantics by the latter. While for physical models the meaning of objects can only be obtained by human interpretation (e.g. types of objects or building surfaces), digital models allow storage of such information in the model itself, thereby allowing them to be used by computer algorithms. This is not the case for all digital models as some only serve for visualization purposes where no semantics are required.

A good example of a 3D city model format supporting semantics is CityGML. It is an open and standardized data model recognized by the Open Geospatial Consortium for the storage of urban and landscape 3D city models. As its name suggests, it bases on the GML (Geographic Markup Language) which is an extensible international standard for spatial data exchange.

The inclusion of semantics was one of the project's core priorities: "One of the most important design principles for CityGML is the coherent modeling of semantics and geometrical/topological properties." (Gröger et al., 2012). In fact, even at a most basic level, clearly distinguishing between modules (such as terrain and buildings) requires more than geometric information. One of the CityGML strengths is therefore that thematic and geometric data can be queried in the same way, making navigation between hierarchies easy (Gröger et al., 2012).

On top of providing content information at the object level, semantics in CityGML are also used to store metadata related to the fulfillment of specifications established by the CityGML standard. A major one is the level of generalization (or accuracy), of which 5 versions are supported and denominated LOD 0 till 4 (0 is most coarse, 4

2 Source top left: 3drotterdam.nl, top right: https://www.flickr.com/photos/victortsu/5175960711/in/photostream/, bottom: https://www.goodfreephotos.com/albums/netherlands/rotterdam/city-view-of-rotterdam-netherlands.jpg

most detailed). The specifications defined by the standard for building roofs are as follows (CityGML version used here is 2.0, Gröger et al. (2012)):

- LOD0: The entire building is represented by a horizontal surface (*gml:MultiSurfaceType*). The main difference with (conventional 2D) cadastral registration datasets is that this surface is embedded in 3D space (for instance by including a digital terrain model). The horizontal surface can either represent the footprint (and be located at ground level) or a horizontal projection of the building roof (and are then located at the height of the eave) as shown in figure 1.10:

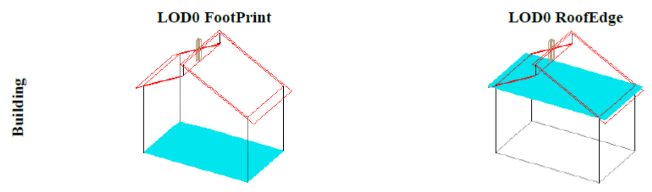


Figure 1.10: Representations of a building as a polygon in LOD0 (Gröger et al., 2012)

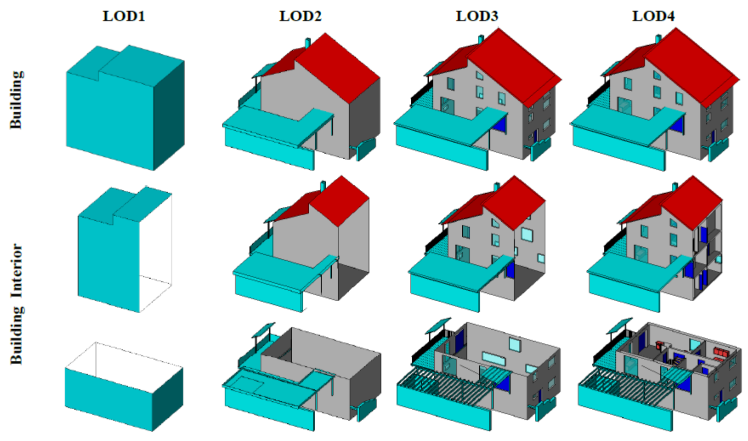


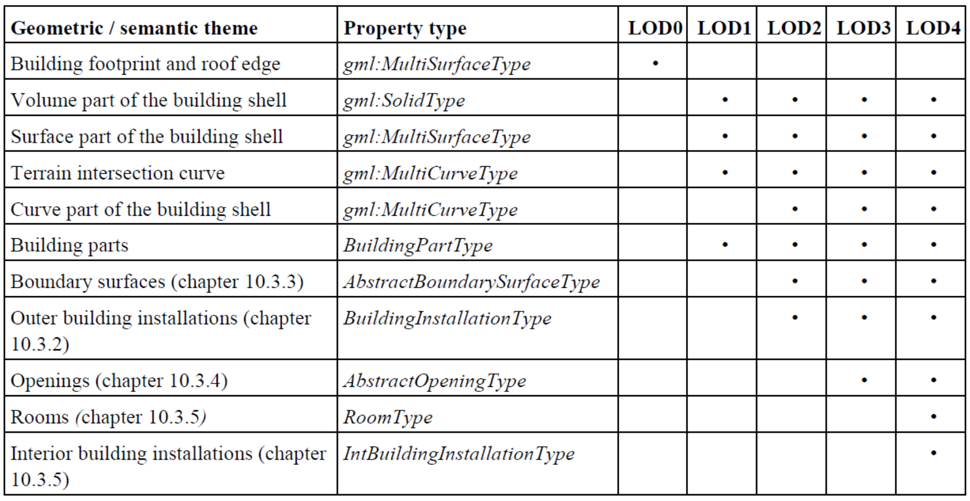
Figure 1.11: Building representation in LOD1-4 (Gröger et al., 2012)

- LOD1: From this level on, the buildings are represented as a volume. In LOD1, entities are aggregated to a simple block (the outer shell) which can either be expressed as solid (*gml:SolidType*) or multiple surfaces (*gml:MultiSurfaceType*). From LOD1 buildings might also be split in building parts but each of these might only have one height value. Therefore, the outer shell does not contain additional details such as the roof shape.

- LOD2: From this level on, semantic objects are required to compose the exterior shell of a building. Theses ones are all of classes *_BoundarySurface* or *BuildingInstallation*. The first class contains special functions like walls, roofs, ground plates, outer floors, outer ceilings or closure surfaces. The latter class is used to store building elements like balconies, chimneys, dormers or outer stairs which are "strongly affecting the outer appearance of a building" (as a threshold 4 by 4m is proposed for LOD2, 2 by 2m for LOD3). Furthermore, roof overhanging parts should be modeled if known.

- LOD3: On top of being more detailed/representing smaller exterior objects, openings (windows, doors) should be modeled separately, creating a hole in the surface within which they lie (*AbstractOpeningType*). From LOD3 on, overhanging roof parts must be modeled as such (this is recommended but optional in LOD2).

- LoD4: LOD4 is the highest level of detail as it covers the building interior too. Rooms and interior installations are therefore modeled too. Moreover, although not formally part of the standard, an increase in absolute 3D point accuracy and decrease in shape generalization is proposed as in figure 1.12:



Geometric / semantic theme	Property type	LOD0	LOD1	LOD2	LOD3	LOD4
Building footprint and roof edge	*gml:MultiSurfaceType*	•				
Volume part of the building shell	*gml:SolidType*		•	•	•	•
Surface part of the building shell	*gml:MultiSurfaceType*		•	•	•	•
Terrain intersection curve	*gml:MultiCurveType*		•	•	•	•
Curve part of the building shell	*gml:MultiCurveType*			•	•	•
Building parts	*BuildingPartType*		•	•	•	•
Boundary surfaces (chapter 10.3.3)	*AbstractBoundarySurfaceType*			•	•	•
Outer building installations (chapter 10.3.2)	*BuildingInstallationType*			•	•	•
Openings (chapter 10.3.4)	*AbstractOpeningType*				•	•
Rooms (chapter 10.3.5)	*RoomType*					•
Interior building installations (chapter 10.3.5)	*IntBuildingInstallationType*					•

Figure 1.12: Property usage for LOD0-4 (Gröger et al., 2012)

1.5.2 LiDAR and remotely sensed point clouds

The LiDAR technology mentioned for the classification correction stands for *Light Detection And Ranging*. This technology is widely spread for the acquisition of point clouds which are also used for the construction of 3D city models discussed in the previous part 1.5.1. It is an optical measurement technique that uses light to measure times-of-flight and/or phases. In fact, as the speed of light is known, the time taken by a light pulse to be reflected (i.e. travel back to the emitter) can be measured and converted into a distance. In combination with the position and orientation of the emitter/receiver, the reflection point position can be determined. LiDAR systems can perform static measurements (on a pole, similarly to total stations) but also be mounted on moving vehicles or planes (in which case an inertial measurement unit is usually required on top of the GPS receiver) (Vosselman and Maas, 2010).

The fact that LiDAR is an optical system means that more than one return can be registered. By registering the full return waveform, several echoes can be identified. This is especially practical for objects which partially transmit light, such as trees (Vosselman and Maas, 2010). Depending on the wavelength, LiDAR signals might also be absorbed by materials such as water (Lemmens, 2011). This mainly depends on the wavelength of the signals: a special application of LiDAR is bathymetry where two pulses are used: one in the longer infrared (e.g. 1060 nm) and another in the green spectrum (e.g. 530 nm). While the first one is reflected by water, the latter is transmitted and reflected by seabed (Vosselman and Maas, 2010).

Another advantage of LiDAR being an optical system is that intensity (often called amplitude) measurements can easily be performed. To do so, the amplitude of the pulse, which characterizes the reflectance of the spot, must be recorded. This value depends on the material reflectivity (which again depends on the wavelength), and on the reflection type (which can either be specular, diffuse or a mix of both) (Vosselman and Maas, 2010).

One should note that LiDAR is far from being the only acquisition technique for point clouds. In fact, laser scanners can also be mounted on other vehicles such as cars and ships, but will hardly be able to give good coverage of roofs that might be out of sight. A more relevant alternative is the usage of dense image matching algorithms. These ones use two high resolution aerial or satellite images taken from different positions to find matching points of which the 3D position is subsequently determined. The advantage of this technique is that aerial images are often already acquired on a regular basis. A drawback is that such algorithms have difficulties with low texture as it leads to lower accuracy (Zhou et al., 2018).

1.6 OBJECTIVE AND SCOPE

CORE LOD2 city models might show a potential if they can be enriched with the missing parts with regard to LOD3. As hyperspectral images are only 2-dimensional a 2-dimensional enrichment (e.g. projected on the existing geometry), thus a semantic enrichment might be sufficient (see figure 1.13). Hereby the requirement for a LOD3 model (as formulated in 1.5.1) might be replaced by one for an enriched LOD2 model.

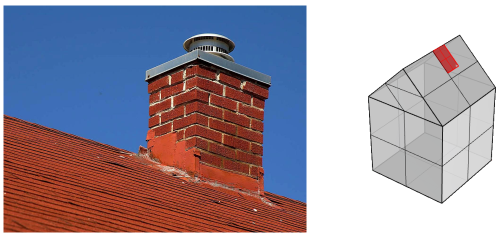


Figure 1.13: Instead of modeling the chimney (left, source: wikimedia) as such it can also simply be indicated semantically in an existing LOD2 model (right, nb: assuming the chimney size is below the custom threshold to include it in LOD2).

The core of the research will thus be to develop a geometric approach to identify "deviations" between CityGML LOD2 models and LiDAR point clouds (thus using only positional, not intensity or multiple return information). The method must be robust for a critical mass of buildings but might exclude particularly complex cases (e.g. smooth edges, roofs overlapping each other, moving elements). The method will build on the fusion of information of different datasets (LiDAR, CityGML) and therefore needs to cope with their specific qualitative strengths and weaknesses.

EXPLORATORY To the knowledge of the author at the time of writing, no study has performed the specific comparisons of point clouds and roof surfaces. The only similar example known is the enrichment of a CityGML model using a 3D mesh that is currently being developed by Willenborg et al. (2018) and bases on research on visualization enrichment performed by Tryfona (2017). Therefore, the research is expected to include a substantial exploratory part.

LINK WITH HYPERSPECTRAL IMAGERY As the motivation of the research is to enhance clean pixel identification for material identification in aerial images (i.e. land cover classification on hyperspectral imagery), the feasibility of this will also be addressed. A connection to such datasets will be developed and resulting performance tested.

VALIDATION The testing of resulting performance means that the scope of the study also covers the validation of obtained results. This validation should be done with a representative set of buildings (e.g. 40 buildings) and will mainly use the imagery pixel as the unit of analysis. Furthermore, both nominal (boolean - thus clean vs. not clean) and rational (degree of cleanliness expressed as a percentage) data types will be addressed in order to get a better idea of the performance of the developed method.

OUT OF SCOPE A topic out of the scope of this thesis but still essential for hyperspectral image classification is the differentiation between sunlit and shadowed pixels (see figure 1.14). In fact, many materials show a different spectrum in shadow, requiring a parallel classification to be identified. It has been shown that this can be calculated either geometrically or using LiDAR intensity values (using an invariant color model) (Priem and Canters, 2016).

ORTHOCORRECTION A topic that is out of scope but has still been addressed in order to improve the specificity of the validation is the orthocorrection. Therefore, a 'rough' version with relatively high error margins will be developed and implemented. In order to determine the degree of specificity, the error margins have been estimated in a mathematical way. The reason why this topic is out of scope is that the exact position of the hyperspectral cells differs from one data acquisition to another. Therefore, the specific position only matters if the method is eventually implemented within ground cover classification algorithms.

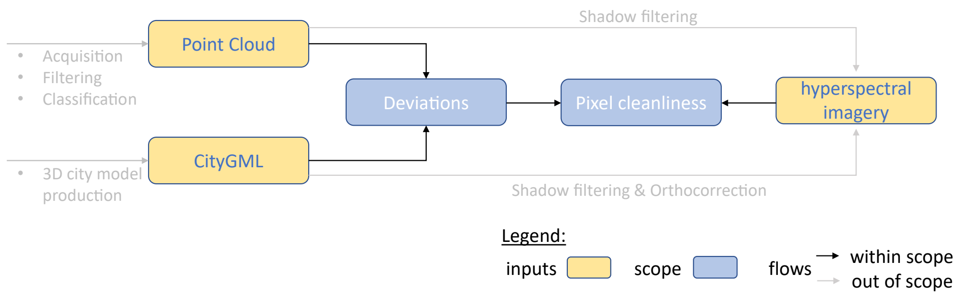


Figure 1.14: Illustration of the scope of this thesis. The parts in light gray are out of scope.

1.7 READING GUIDE

For the ease of reading and to structure the research, this thesis has been divided into several parts. First of all, chapter 2 introduces the core of the research and the way in which the subsequent chapters 3, 4, 5 and 6 are structured. The latter closes the research by providing both quantitative and qualitative findings. Finally, the thesis is concluded by chapter 7 in which an answer is given to the research questions, recommendations are made for data suppliers and topics for future research identified. Chapter 8 contains a number of annexes which are mentioned in the relevant parts.

2 METHODOLOGY

This chapter will give a short introduction to the overall approach that was taken for this research. It will introduce the different steps addressed by the subsequent chapters, along with the assumptions (criteria) guiding each decision. In the second part, the structure in which the following chapters will be presented is introduced.

While the needs of the specific steps do differ (and criteria/assumptions therefore too), there is a shared assumption framework linked to the aim of adding information. In fact, completeness was an important criterion from start to end. As the aim is to improve the identification of spectral variations, leaving some of these deviations aside (e.g. to make the algorithm faster) is not desired. For each step of the outline that will be introduced in part 2.1, this common criteria will be specified. Furthermore, additional criteria that are more specific to the aims of the respective step will be formulated where needed.

Overall, this research takes place under the paradigm of remote sensing data fusion which, as formulated by Zhang (2010), "aims to integrate the information acquired with different spatial and spectral resolutions from sensors mounted on satellites, aircraft and ground platforms to produce fused data that contains more detailed information than each of the sources.". In fact, this research combines information from three spatial datasets of different nature: a LiDAR point cloud, a 3D city model and an optical imagery file (see figure 2.1).

Zhang (2010) also mentions that this fusion can take place at pixel, feature or decision levels. This research belongs to the first group as it will ultimately indicate whether the pixels of the optical imagery file are clean or not, an additional information with regard to the standalone input file.

2.1 OUTLINE OF THE RESEARCH AND RESPECTIVE CRITERIA USED

The research conducted in this master thesis can be divided into several steps. As mentioned in 1.4, the aim is to find a method that allows the identification of 'clean' pixels (within the field of hyperspectral material classification) by first semantically enriching a CityGML LOD2 model.

Resulting from the introduction and more specifically part 1.3, the the following aspects had to be considered:
- the societal and scientific context of material classification; the needs of the final usage(s) of the data output (e.g. identification vs. quantification of materials).
- the hyperspectral imagery acquisition techniques and their implications for spatial data fusion (e.g. ground sampling distance and technology, data processing and georeferencing).
- the lacking availability of LOD3 models, but the need for details that are usually below the (custom) threshold size for LOD2.
- the sampling interval (e.g. point cloud density or ground sampling distance) and accuracy of the different datasets used.

The overall approach that has been chosen is to compare a LiDAR-acquired point cloud with an existing CityGML model and subsequently merge the result with the hyperspectral imagery. A general overview can be found in 2.1.

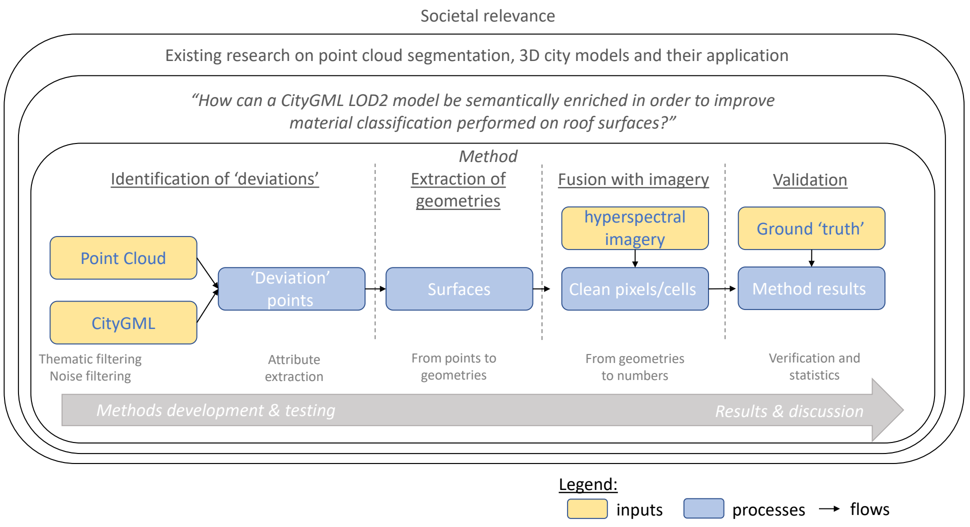

Societal relevance
Existing research on point cloud segmentation, 3D city models and their application
"How can a CityGML LOD2 model be semantically enriched in order to improve material classification performed on roof surfaces?"
Method
Identification of 'deviations'
Extraction of geometries
Fusion with imagery
Validation
Point Cloud
CityGML
'Deviation' points
Surfaces
hyperspectral imagery
Clean pixels/cells
Ground 'truth'
Method results
Thematic filtering
Noise filtering
Attribute extraction
From points to geometries
From geometries to numbers
Verification and statistics
Methods development & testing
Results & discussion
Legend:
inputs
processes
flows

Figure 2.1: Main steps of the research methodology used. For the four central parts/chapters, specific background knowledge will be discussed before introducing the implementation.

CHAPTER 3, 'DEVIATION IDENTIFICATION' covers the extraction of information from the data inputs and the identification of points belonging to 'deviations'. Here, parallels to the earlier mentioned works of Willenborg et al. (2018) and Tryfona (2017) who compare a 3D mesh and a 3D city model to enrich the latter can be found. Similarities can also be found with the research conducted by Zhou et al. (2018) using dense image matching to update LiDAR datasets, hereby comparing two datasets of similar nature.

For the first part, the completeness criterion was translated as the identification of all deviations within the scope of this research (an example that is out of scope are non-geometric deviations). Therefore the 'completeness' criterion is limited to geometric deviations, which should be identified as reliably as possible.

Another assumption was the computational load criteria. In fact, point clouds as used in this research are datasets of considerable size (in this research, the selected point cloud covers the study area with 27 564 055 points). As the first step needs to perform computations on the entire input, a small difference inside a single iteration can have considerable consequences. In order to ensure the feasibility of the research, this had to be considered.

CHAPTER 4, 'FROM POINTS TO SURFACES: EXTRACTION OF GEOMETRIES' covers the conversion of the set of points identified as 'deviations' into surfaces. In fact, LiDAR acquisition can be seen as a form of sampling where the points actually represent a surface beyond the points themselves. Furthermore, a vectorized representation is necessary for other purposes such as reliable area quantification and visualization of 'deviations'.

The second part interprets completeness as the ability to cover all input cases. No matter the shape or the size of the 'deviation', the approach should support it. Also, the presence of holes inside point groups and even disconnected groups deserved a part of this focus.

A related, but not exactly identical criteria used is automation. It is here understood in opposition to human interventions in algorithms. These ones can occur if a setting of the algorithm needs to be adapted to the specific situation (i.e. point density, shape typology). While a single setting for the entire dataset might still be acceptable, human interventions at a smaller level (e.g. tuning a setting for each building or 'deviation') should definitely be avoided.

CHAPTER 5, 'DATA FUSION WITH HYPERSPECTRAL IMAGERY' presents the third step that consists of combining the geometric data from a hyperspectral imagery dataset with the previously created 'deviation' geometry. This step is necessary to allow the selection of 'clean' pixels among the set of candidate pixels of a building.

This third part understands completeness as staying as close as possible to raw data (from a spatial point of view). In fact, the hyperspectral imagery data contains all geometric distortions that appear during acquisition. Along with this, the corrected coordinates are obtained from a separate file. Often enough, these two files are merged to obtain a resampled raster. The result of this might, however, be data loss and interpolation artifacts (de Miguel et al., 2014; Vreys et al., 2016), which is the reason it will be avoided here.

Within this third part, the orthocorrection which was already mentioned as being out of the main scope (but still part of this report) was additionally performed. As the aim of this correction is to increase the specificity of the subsequent validation without consuming extensive time budget, the focus within this orthocorrection was on limiting the number of computational inputs and steps.

CHAPTER 6, 'RESULTS AND ANALYSIS' covers the validation of the outcomes of this approach and discusses the findings. This final step is more specific as it performs a quantitative analysis, which requires a specific study area and control data (so-called ground 'truth'). This validation area is located on the south bank of the Maas in Rotterdam and was chosen with regard to representativeness, data availability and limited time budget considerations.

Furthermore, the method was tested with several variables with the aim to contribute to completeness too. Additionally, qualitative observations were also made along with the quantitative ones and are discussed too. The findings will form the basis for the final chapter 'Conclusions, lessons learned and outlook' (chapter 7).

2.2 REPORTING STRUCTURE

Chapters 3, 4 and 5 are structured in a fairly similar way as they all cover a part of the data processing scheme. The splitting in three parts emanates from the usage of different background knowledge which will always be presented at the start of the chapter. If applicable, the input datasets will also be introduced.

In the second step, each chapter will explain the reasoning that led to the chosen theoretical framework. This one can either rely on experimental observations or build directly upon theoretical knowledge based on the existing literature. Furthermore, the resulting theoretical framework will be clearly formulated and serve as a starting point for the final part, the implementation.

The implementation will further elaborate on the specific tools used for performing data processing. Some general software used in this thesis that can be introduced here are:
- the main programming language that has been used for this thesis is *Python* 3.7. along with the *numpy* 1.15.4 library.
- for visualization of 2D geometries and aerial images, *Quantum Geographic Information System (QGIS)* 3.2.3 has been used.
- for visualization of point clouds, *CloudCompare* 2.9.1 has been used.
- for visualization of CityGML geometries, in combination or without point clouds, *Feature Manipulation Engine (FME)* 2018.1 has been used.

One should further note that the code of the implementation will not be covered by the implementation parts. To allow a better understanding, pseudo-code will be used. The source code of this thesis can nevertheless be found at the following repository:

https://github.com/Flyalbatros/3D-Models-in-Urban-Mining.git

3 DEVIATION IDENTIFICATION

This chapter will present the process that led to the choices regarding the first part of the research, namely the identification of 'deviation' points. For this purpose, mainly two input datasets have been used: a 3D city model (CityGML LOD2 compliant) and a point cloud dataset.

In the first part, relevant background knowledge from the field of 3D surface reconstruction (see part 3.1.1) and a more detailed study of relevant CityGML semantics will be introduced (see part 3.1.2).

In the second part, the theoretical framework will be established, starting with the selection of input files (in part 3.2). Subsequently, an exploration study is performed (part 3.3.1) and concluded by formulating the theoretical framework specific to this step.

The last part of this chapter presents the implementation, elaborating in detail on the algorithmic approaches and mathematic reasonings used (see part 3.4).

3.1 BACKGROUND: 3D SURFACE RECONSTRUCTION AND CITYGML SEMANTICS

3.1.1 3D surface reconstruction

According to Nguyen and Le (2013), the field of feature recognition in point clouds can be divided into six subfields as shown in figure 3.1. These subfields approach a similar topic with different paradigms and are shortly explained.

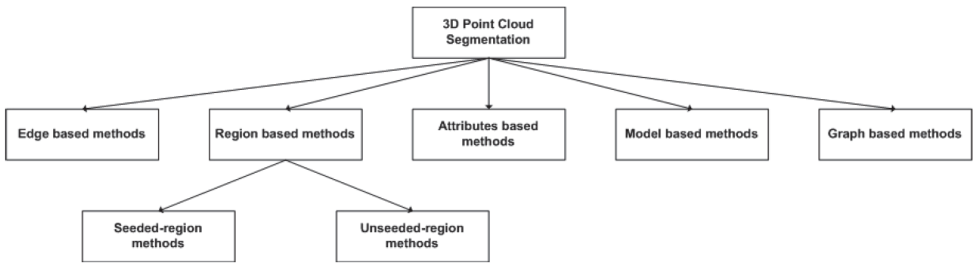


Figure 3.1: Taxonomy of 3D point cloud segmentation methods as proposed by Nguyen and Le (2013)

One might note that the classification provided by Nguyen and Le (2013) is not the only possible perspective as it focuses on mathematical techniques. Other ones focus on type of output (part-type or primitive geometries vs. patch-type or homogeneous regions) (Vosselman et al., 2004; Wang and Shan, 2009) or on representation used (edge/boundary vs. surface-based) (Wang and Shan, 2009). Also, some mathematical techniques might be used for both types of representation (such as the ones introduced in part 3.1.1), so these techniques are rather characterizing than classifying.

Taxonomy of 3D point cloud segmentation methods

A. EDGE-BASED METHODS These methods aim at outlining boundaries by detecting geometric or intensity properties. To do so, mainly edge-detection algorithms from computer vision area are used (e.g. canny edge detection using Sobel operator). Therefore, 3D data needs to be converted into a 2.5D range image, which often implies a loss of information (Wang and Shan, 2009).

B. REGION-BASED METHODS. This category which uses neighborhood information to combine nearby points with similar properties can be divided into two subcategories: seeded and unseeded methods. The first one identifies a number of characteristic surface patches (using attributes such as planarity or curvature) and then gradually extends these patches to sufficiently similar points (basing on measures such as proximity, slope, curvature, normals). A big challenge of these methods is optimal seed selection (Nguyen and Le, 2013) and sensibility to noise (Nguyen and Le, 2013; Nurunnabi et al., 2012; Gilani et al., 2016).

Unseeded methods use an alternative top-down approach. First, all points are in a single group which is split until all parts satisfy the given threshold criteria. Challenges here are avoiding over-segmentation and prior knowledge requirements which usually cannot be satisfied in complex scenes (Nguyen and Le, 2013).

C. ATTRIBUTES BASED METHODS In this method, first, an attribute computation is performed: each point is associated with a feature vector which consists of one or several geometric or radiometric measures. In a second step, unsupervised clustering is performed (popular methods include k-means, fuzzy clustering, maximum likelihood). As it is carried out on the feature space, this method can be used on point clouds, raster data, and TINs directly. Reliable results can be achieved but depend on the quality of the attributes, the computation of the feature vectors and the clustering technique. Also, the dimensionality of the feature vector can be a challenge for computation speed (Nguyen and Le, 2013; Wang and Shan, 2009).

D.MODEL-BASED APPROACHES These approaches are also known as "direct extraction of parameterised shapes" (Vosselman et al., 2004) and rely on geometric primitives such as planes, cylinders or spheres. Popular examples are the 3D Hough transform (for planes, with variants for cylinders) and RANSAC (for RANdom SAmple Consensus). Advantage of these methods is their speed and their robustness with regard to outliers (Nguyen and Le, 2013).

E. GRAPH-BASED METHODS In this category, the point cloud is represented as a graph with the points as vertices and edges representing connections with a weight attribute for their similarity. Segmentation is then achieved by decimating (partitioning) the graph, for instance where connections are weakest. Popular techniques include normalized cut, minimum spanning tree and spectral graph partitioning (Wang and Shan, 2009).

Challenges of point cloud data segmentation

While all taxonomies have their specificities, shared aspects also exist. Three recurring challenges resulting from this are discussed by Nurunnabi et al. (2012):

A.TYPES OF EDGES Three different types of edges can be considered (illustrated in figure 3.2): (a) crease edges which can, for instance, be found at the meeting of planes of a same roof. While there is a proximity between the planes' edge points, a difference between surface normals can be observed. (b) jump edges can, for instance, be found between a roof's eave and ground surface. As the name indicates, there is a discontinuity between the planes: planes' edge points show no proximity but the normals of the surfaces might be the same (such as in the case of a flat roof above flat ground). (c) a third type is smooth edges which are characterized by surface continuity but show a change of curvature [1] (Nurunnabi et al., 2012; Wang and Shan, 2009).

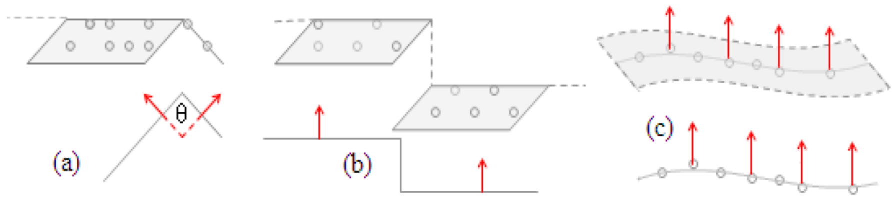


Figure 2. (a) crease edge (b) jump edge (c) smooth edge.

Figure 3.2: Illustration of the different types of edges (Nurunnabi et al., 2012)

B.GAPS As point clouds are scattered and unorganized (Wang and Shan, 2009), gaps in the data are no exception. Occlusion depending on the geometry and the scan angle or absorption by the presence of water (for some laser beam with near-infrared wavelengths) occur in nearly all datasets (Vosselman and Maas, 2010). Robust methods, therefore, need to be designed to handle gaps, for instance avoiding splitting an affected surface (Nurunnabi et al., 2012).

C.OUTLIERS/NOISE As mentioned in the explanations of the region based methods, outliers (which are a form of noise) can be very challenging for point cloud surface reconstruction. In fact, they can contaminate statistics calculated on the local neighborhoods, resulting in, for instance, a tangent plane biased in the direction of the outlier (Nurunnabi et al., 2012). Nurunnabi et al. (2012) and Gilani et al. (2016) proposed region-based methods with the aim to be more robust to outliers using respectively Projection Pursuit combined with Minimum Covariance Determinant and Low-Rank-Subspace with prior Knowledge methods. Depending on the type of dataset obtained, outliers might also have been filtered out at an earlier stage - and it is therefore a critical point to consider beforehand (see part 3.2.1).

Relevant tools

In anticipation of the next parts, two relevant tools which are commonly used by different strategies presented in part 3.1.1 will be introduced here.

PCA The first tool is the PCA which in practice creates the best fitting plane for a group of 3 or more non-aligned points. This can be relevant in order to derive an orientation attributes (the normal vector) from a given group of points. The first step is the creation of the covariance C matrix of size n×n where n is the number of dimensions (in the case of this thesis: 3 dimensions).

$$C_{3x3} = \frac{1}{k}\sum_{i=1}^{k}(p_i - \overline{p})(p_i - \overline{p})^T, \overline{p} = \frac{1}{k}\sum_{i=1}^{k} p_i \tag{3.1}$$

With p_i the input points and $\overline{p}$ the mean of the latter.
Basing on the covariance matrix, the eigenvalues λ can be extracted along with the

1 The smooth edges are less common in the built environment and will thus not be addressed by this research

eigenvectors V (the number of eigenvalues and eigenvectors is equal to the number of dimensions):

$$\lambda V = CV \tag{3.2}$$

The smallest of all three eigenvalues can then be used to identify the corresponding eigenvector which is the normal vector. Furthermore, the curvature σ can also be obtained using the eigenvalues λ (Nurunnabi et al., 2012):

$$\sigma = \frac{\lambda_0}{\lambda_0 + \lambda_1 + \lambda_2} \tag{3.3}$$

The identification of a point group's normal vector (by PCA/identifying the best fitting plane) can function as a feature indicating the orientation of a group of points.

FIXED-DISTANCE NEIGHBORS (FDN) AND K-NEAREST NEIGHBORS (KNN) A second set consists of two tools that allow the formal definition of the 'neighborhood of a point'. This is necessary in order to formally define local groups of points which can be used to derive features. Here, two different approaches exist.

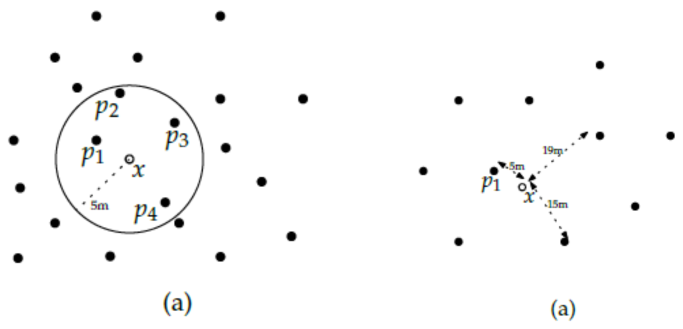


Figure 3.3: Illustration of the fixed-distance neighbors (FDN) (left) and KNN (right) approaches. While on the left all points within 5m are selected independently of their number, on the right the three closest points are selected independently of their distance (source: course material GEO1015).

The first one is the FDN. This one defines the neighborhood of a point by drawing a circle with a given radius (the fixed-distance) around the point (see figure 3.3 on the left). All points lying inside that circle are then selected as neighbors. Therefore, the number of neighbors for a given distance will vary, depending on the distribution in the vicinity (and might even be 0 in some cases) (Bentley, 1975).

This is in contrast with the KNN. This one defines a fixed number of neighbors rather than the radius of a selection circle (see figure 3.3 on the right). For each point, the defined number of neighbors is then selected, independently of their distance to the query point (for instance using a k-dimensional tree (kd-tree)). Therefore, the number of points selected as neighbors will be static and independent of the distribution of the vicinity. The method is thus more resistant to irregular point densities, i.e. the number of nearest neighbors will be k in all circumstances (and thus never 0) (Beyer et al., 1999). As point clouds are characterized by irregular distribution, only the KNN approach will be used in this thesis (Vosselman and Maas, 2010).

3.1.2 Semantics of the CityGML format

As mentioned in part 1.5.1, the research of this thesis requires a semantic 3D city model. The city model must, therefore, contain more than merely geometries and textures, it must also contain an ontological structure with semantic properties. Examples of the latter are thematic properties (for instance type of objects - see figure 3.4).

A wide-spread data model covering such requirements is the CityGML format, which has been adopted by several cities in the world (according to *citygml.org*, at least than 17 cities/regions have produced datasets in CityGML, of which 11 fulfill LOD2 standard). One of these LOD2 city models has been developed for Rotterdam, of which the specific dataset is described in part 3.2.2. As the scope of this thesis is limited to roof materials, only the semantic storage of building roof geometries and possibly materials will be discussed here (although the possibilities offered by CityGML are much bigger).

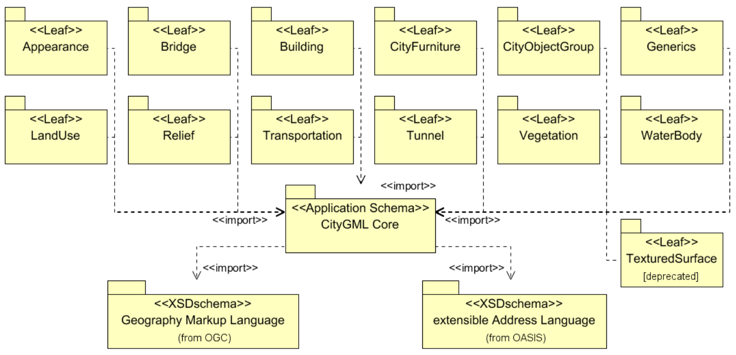


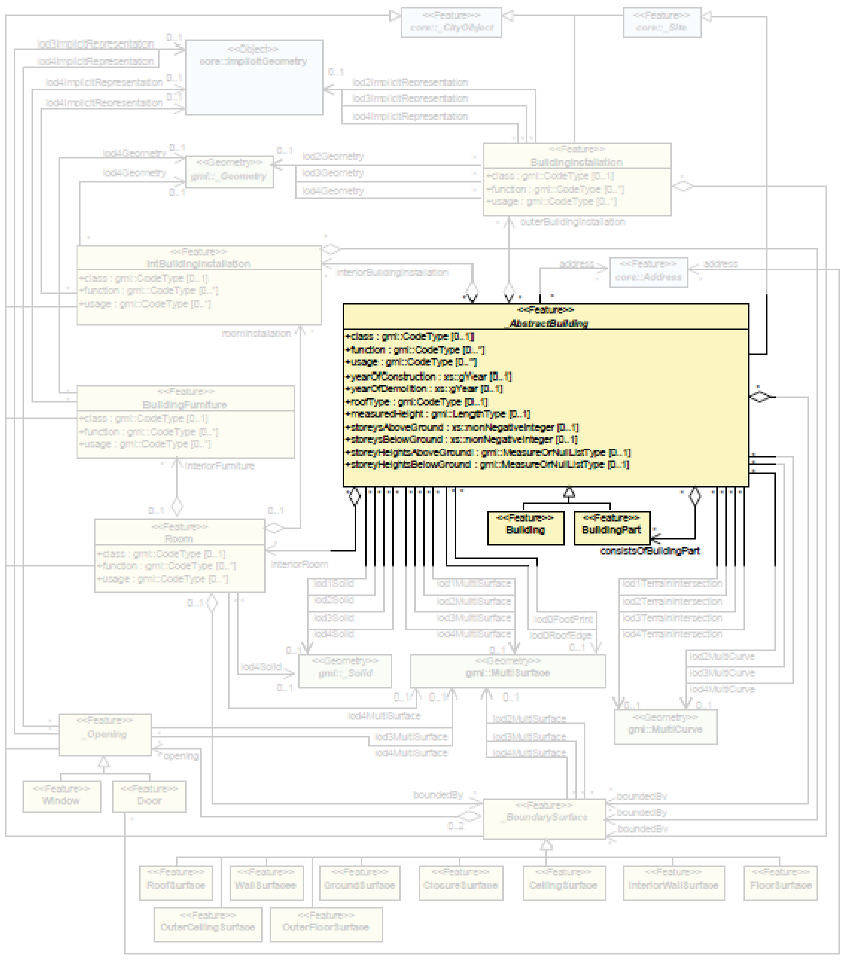
Figure 3.4: UMl diagram illustrating the different modules of CityGML (Gröger et al., 2012). This thesis will only cover the 'Building' module

.

BUILDINGS AND BUILDING PARTS A first aspect of relevance is that from LOD1 on, buildings can be subdivided into parts. As the UML diagram provided by the CityGML specifications shows (figure 3.5), the central feature of the *Building* module is called *_AbtractBuilding*. Both *Building* and *BuildingPart* features inherit of the class *_AbtractBuilding*, the only difference being that several *BuildingPart* can be aggregated into a *_AbtractBuilding* (thus either another, higher level *BuildingPart* or a *Building* feature). As mentioned in part 10.3.9 of Gröger et al. (2012), the highest level class of such an aggregation always has to be a *Building* class (Gröger et al., 2012).

In practice, this means that any *Building* feature can be subdivided into *BuildingPart* classes. There is no limit to the number of *BuildingPart* features. Furthermore, *BuildingPart* classes can be aggregated into groups which are represented by another *BuildingPart* class (except when the highest level is reached, then it needs to be a *Building* class). This means that it is possible to hierarchize *BuildingPart* feature into as many levels as one wishes. Any *Building* and *BuildingPart* classes do further support the same attributes (semantically, thematically and geometrically) (Gröger et al., 2012).

GEOMETRIC AND SEMANTIC SURFACES Any *_AbstractBuilding* class is related to *geometry* class. This class exists for each of the supported LOD. In each LOD, this can either be a $gml :: \_Solid$ or a $gml :: MultiSurface$ geometry (with the exception of LOD0 where the geometry must be either a footprint or a roof edge, and thus $gml :: MultiSurface$) (Gröger et al., 2012). Furthermore, for a given LOD, the geometry might also be stored both in solid and multisurface versions (for instance to support faster visualization).



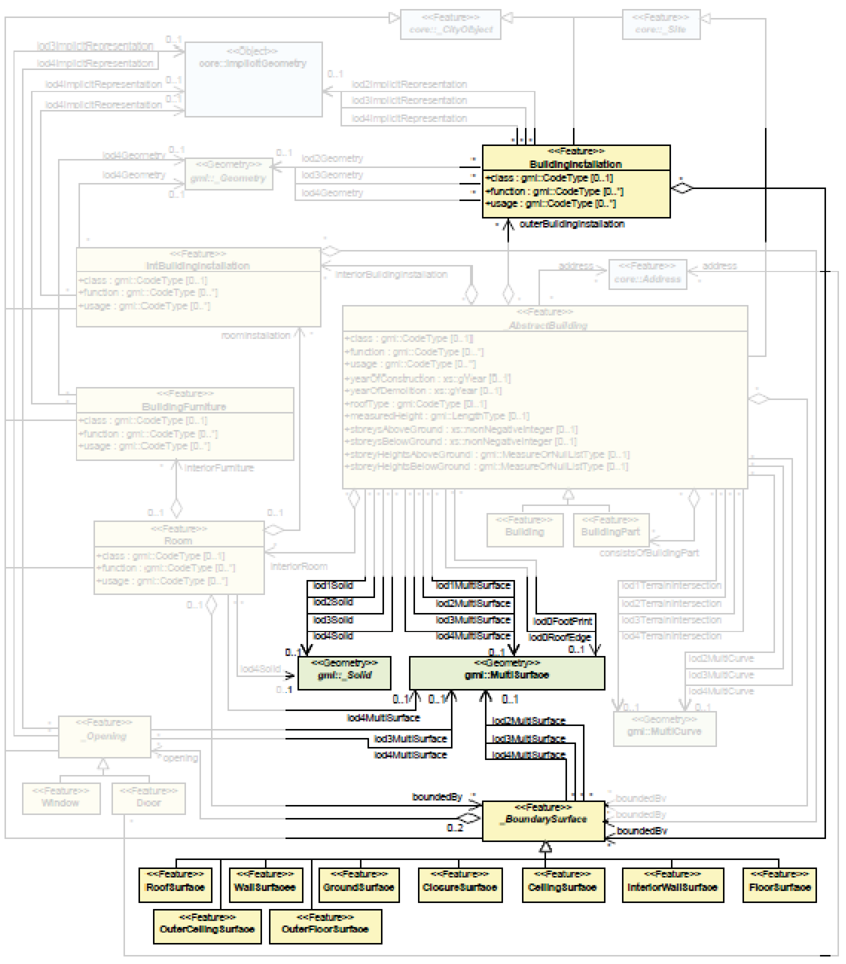
<<Feature>> core::_CityObject
<<Feature>> core::_Site
<<Object>> core::ImplicitGeometry
<<Geometry>> gml::_Geometry
<<Feature>> BuildingInstallation
<<Feature>> IntBuildingInstallation
<<Feature>> core::Address
<<Feature>> _AbstractBuilding
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
+yearOfConstruction : xs::gYear [0..1]
+yearOfDemolition : xs::gYear [0..1]
+roofType : gml::CodeType [0..1]
+measuredHeight : gml::LengthType [0..1]
+storeysAboveGround : xs::nonNegativeInteger [0..1]
+storeysBelowGround : xs::nonNegativeInteger [0..1]
+storeyHeightsAboveGround : gml::MeasureOrNullListType [0..1]
+storeyHeightsBelowGround : gml::MeasureOrNullListType [0..1]
<<Feature>> Building
<<Feature>> BuildingPart
consistsOfBuildingPart
<<Feature>> BuildingFurniture
<<Feature>> Room
<<Geometry>> gml::_Solid
<<Geometry>> gml::MultiSurface
<<Geometry>> gml::MultiCurve
<<Feature>> _Opening
<<Feature>> Window
<<Feature>> Door
<<Feature>> _BoundarySurface
<<Feature>> RoofSurface
<<Feature>> WallSurface
<<Feature>> GroundSurface
<<Feature>> ClosureSurface
<<Feature>> CeilingSurface
<<Feature>> InteriorWallSurface
<<Feature>> FloorSurface
<<Feature>> OuterCeilingSurface
<<Feature>> OuterFloorSurface

Figure 3.5: Extract of the UML diagram for the building class (big version in annex 8.1), showing the *_AbtractBuilding* class and its relation to *Building* and *BuildingPart* classes (Gröger et al., 2012)

.

From LOD2 on, semantic information can be added to the outer facade of a building (to the *gml* :: *MultiSurface*). This is done by adding for each *geometry* a relation to a *_BoundarySurface* or a *BuildingInstallation* class (Gröger et al., 2012). The *_BoundarySurface* class will not be found as such as it is a superclass consisting of a multitude of other features covering the different functions of boundary surfaces. The most interesting subclass in the frame of this thesis is the *RoofSurface*. Another class that might be encountered in building shell parts that are visible from the sky is *OuterFloorSurface*. This one can, for instance, be encountered to model a rooftop terrace. Smaller elements that do not have the significance of building parts (e.g. dormers, balconies, ventilation boxes) but still affect the boundary surface can be modeled using the *OuterBuildingInstallation* class (Gröger et al., 2012).

ENERGY ADE AND STORAGE OF ROOF MATERIALS The CityGML standard as such does support the storage of visual textures but not of materials from a semantic point of view (Gröger et al., 2012). While the CityGML standard is, in theory, extensible by using so-called *genericAttribute*, a better option is the usage of Application Domain Extension (ADE) as they provide a more standardized way. A good example is the *Energy ADE* which introduces the notion of thermal boundary. Such thermal boundary can be based on a building's boundary surface but does not have to. Furthermore, *AbstractConstruction* class is related to this thermal boundary and facilitates the storage of one or several materials (Agugiaro et al., 2018).



<<Feature>> BuildingInstallation
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
outerBuildingInstallation
lod1Solid
lod2Solid
lod3Solid
lod4Solid
lod1MultiSurface
lod2MultiSurface
lod3MultiSurface
lod4MultiSurface
lod0FootPrint
lod0RoofEdge
<<Geometry>> gml::_Solid
<<Geometry>> gml::MultiSurface
lod4MultiSurface
lod3MultiSurface
lod4MultiSurface
lod2MultiSurface
lod3MultiSurface
lod4MultiSurface
boundedBy
<<Feature>> _BoundarySurface
boundedBy
<<Feature>> RoofSurface
<<Feature>> WallSurface
<<Feature>> GroundSurface
<<Feature>> ClosureSurface
<<Feature>> CeilingSurface
<<Feature>> InteriorWallSurface
<<Feature>> FloorSurface
<<Feature>> OuterCeilingSurface
<<Feature>> OuterFloorSurface

Figure 3.6: Extract of the UML diagram for the building class (surfaces highlighted, big version in annex 8.1), showing the relation between *_AbtractBuilding* and its different geometric representations, including the class *_BoundarySurface* (Gröger et al., 2012).

3.2 SELECTION OF INPUT DATASETS

3.2.1 Point cloud files

For the point cloud concerning the study area in Rotterdam (which is further defined in chapter 6), the choice was given between two files: the Algemeen Hoogtebestand Nederland - General height dataset of the Netherlands (AHN) which is openly accessible and a proprietary file owned by the city of Rotterdam and acquired for the production of the 3D city model.

AHN The AHN is an open point cloud dataset that can be downloaded in *.las* format from https://www.pdok.nl/nl/ahn3-downloads. It provides the following attributes for each point (information obtained from (American Society for Photogrammetry & Remote Sensing, 2013)):
- x,y,z coordinates in the local reference system (EPSG[2]:28992)
- intensity of the return (pulse return magnitude)
- return number
- total number of returns
- classification (using the *.las* standard classes and a custom one, code 26 is used for

2 European Petroleum Survey Group, a widely used data base for coordinate reference systems)

infrastructure such as bridges)
- Global Positioning System (GPS) time of the acquisition moment

some other, less interesting characteristics are also included:
- scan direction (this one is 0, for a negative scan direction, 1 for a positive one)
- scan angle (output angle of the beam, seems to always be 0)
- flight line edge (is usually 0, 1 only if the point is at end of scan)

two last characteristics are not relevant for the user:
- point source id (indicates the file from which the point originated)
- user data (this is a custom field, which can be freely used by the data producer)

The flight paths of acquisition flights can be found on http://www.ahn.nl/index.html. For the study area used later in chapter 6, the point cloud was acquired on 4/12/2016.
According to the requirements established by the Dutch government (AHN, 2015), there is no strict minimum number of points per m^2. Instead, other requirements are established:
-the stochastic height error must not be higher than 5 cm (standard deviation).
-the systematic height error must not be higher than 5 cm.
-objects of 2 by 2 meters must be identifiable with a positional deviation of no more than 50cm.

As the optimal settings for the point density, point distribution, and height accuracy depend on the hardware used, the exact specification of the point density was left to the data acquiring parties. For the study area in this research, the density is about 8 points/m^2 [3].

POINT CLOUD PROVIDED BY THE CITY OF ROTTERDAM A second point cloud that was obtained for the study area was acquired for the City of Rotterdam directly. One of the purposes of this acquisition (while then AHN does already provide an open dataset) is to have a more suited product for the production of 3D City models (as mentioned in part 3.2.2). For this thesis, the technical requirements were shared by the city of Rotterdam and contain the following information (Gemeente Rotterdam, 2016):
- for the parts above the city, acquisition with two flight lines that cross each other is required. This allows the reduction of occlusion.
- the density of the point cloud must be of 15 respectively 30 points/m^2 (single and double flight lines). When sampling the dataset into cells of 1 m^2, at least 95% of the cells must fulfill this.
- no more than 0.1% of the points are outliers.
- the stochastic height error has a standard deviation of no more than 5cm, and systematic height errors are not higher than 5 cm.

Furthermore, the point cloud provided by the city of Rotterdam also underwent classification. In the specific case of this thesis, only the class 'building' was obtained, mainly due to data size considerations. The dataset obtained was also in *.LAS* format and contains all data fields which the AHN contains, with the exception of the GPS time.

COMPARISON An overview table of the characteristics for the two point cloud datasets available for the study area that will be used in chapter 6 can be found below:

3 Basing on a sample of 10 000 m^2 of roof surface within the study area used in chapter 6.

	Comparison of the two LiDAR datasets	
	AHN dataset	Rotterdam dataset
Stochastic height error	max. 5cm	max. 5cm
Systematic heigh error	max. 5cm	max. 5cm
Acquisition flight lines	1	2, crossing
Points per m^2	8	30
Acquisition moment	4/12/2016	2016
Classified	yes	yes

Table 3.1: Comparison of the two LiDAR datasets that were available for this study.

From a technical point of view, the two datasets only differ in their point density and number of flight lines (see table 3.1). In order to make a choice, a sample building (Lentstraat 20) with 'deviations' (chimneys, roof windows) was selected for closer analysis. This allowed discovering significant differences that are of major importance for the aim of this research. The class 'building' seems to be too strictly defined for the case of the second point cloud. In fact, two chimneys present on the building are completely missing, while they are present in the AHN dataset (see figure 3.7). It seems rather unlikely that these ones were missed by the laser scanner as each chimney is already represented by 5-10 points in the low-density version (although a different wavelength might have been used for the two datasets, and one of them absorbed or reflected differently).

Another explanation is that the classification algorithm used for labeling the 'building' points interpreted this chimney as a group of outliers and did thus filter it out.

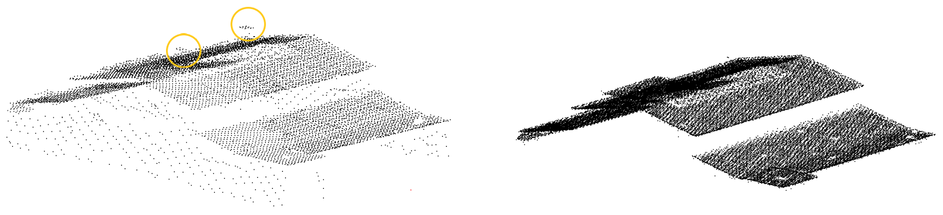


Figure 3.7: Samples of the two available point clouds. On the left AHN, the chimneys on the roof can clearly be seen. In contrast, the points of the chimney are completely missing in the example on the right (City of Rotterdam point cloud).

It should nevertheless be noted that the AHN does also have disadvantages. The classification algorithm might, in fact, have been too 'loose' as in the same case at Lentstraat 20, points were found inside the building. Interestingly, this must explicitly be avoided according to the official requirements (AHN, 2015), but can nevertheless be encountered.

Furthermore, the AHN point cloud contains more occluded locations, which is rather unsurprising given it was only acquired in one direction. This can be observed for the sample (Albert Plesmanweg 103-110, which was also used for the exploration study) in figure 3.7. Interestingly, some occlusion locations persist in the high-density versions.

While the second point cloud dataset has a potential due to its higher density and less occlusion, this is not out weighed by the observations made here. Also, it should be noted that while a higher point cloud density will cover more details, it also involves heavier computation. For the same building, more points would have to be processed. Especially for some data preparation steps such as identifying the points inside a roof surface (see part 3.4), this is not marginal.

With these considerations in mind, the decision was taken to conduct the research with the AHN point cloud and to only consider the one of the city of Rotterdam if difficulties related to point cloud density and/or occlusion arise.

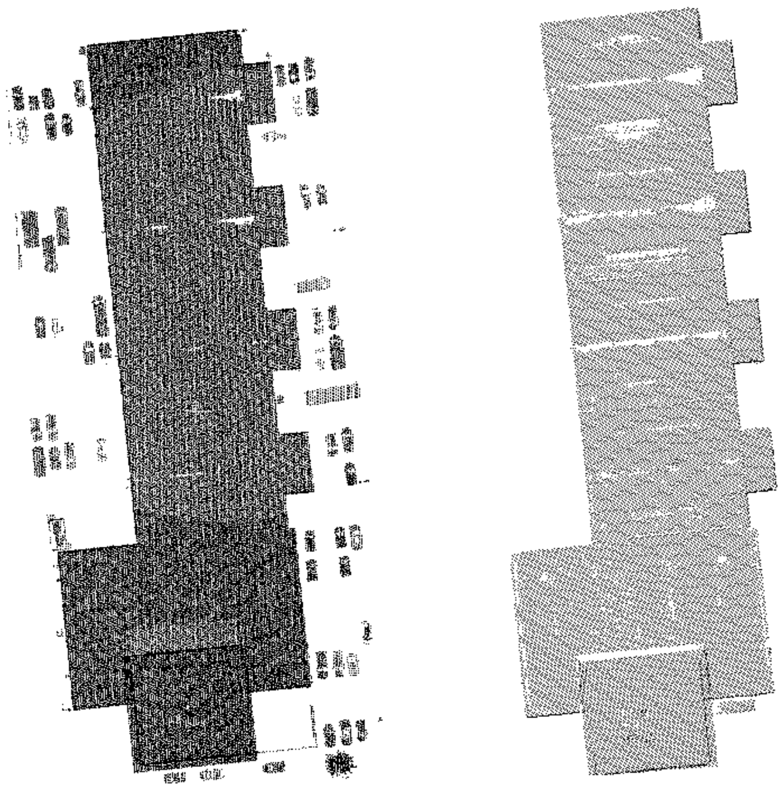


Figure 3.8: Samples of the occlusion in the two available point clouds. On the left (AHN), the chimneys on the roof can clearly be seen. In contrast, the points of the chimney are completely missing in the example on the right (City of Rotterdam point cloud).

3.2.2 CityGML file

For this research, which eventually results in a case study for validation on a part of the city of Rotterdam (see chapter 6), a CityGML file was obtained from the city of Rotterdam. This dataset is open and can be visualized and downloaded by anyone on the website *www.3Drotterdam.nl*.

The technical requirements published for the tender (Gemeente Rotterdam, n.d.) were obtained for this thesis and contain the following technical characteristics:
- the CityGML file should be of LOD2 and fully semantic (Building Solid and thematic boundary surfaces).
- all roof surfaces with an area of at least $4m^2$ and a height difference of 30cm between the point cloud and the roof surface must be modeled.
- the footprint of the 3D buildings must be based on the data of the Basisregistratie Adressen en Gebouwen - Basis Registration Addresses and Buildings (BAG).
- modeling of buildings must be made automatically, with the exception of objects where a difference of more than 10% between the BAG and Basisregistratie Grootschalige Topografie - Basis registration high scale topography (BGT) exists.

Within this dataset, mainly the first subclass of the *_BoundarySurface* class introduced in part 3.1.2, *RoofSurface* is of interest as it can be considered most representative for the area where the roof surface material can be found. Furthermore, considering it only is sufficient from a geometric point of view as *OuterBuildingInstallation* and *OuterFloorSurface* are not simply docked but truly integrated with the boundary surface (see figure 3.9). This approach is not mandatory (the *OuterBuildingInstallation* could also be 'glued' without being integrated with the boundary surface) but was followed here. As the creation of the dataset which will be used in this thesis was automatic (see 3.2.2), it will be assumed that this approach was coherently used for all buildings.

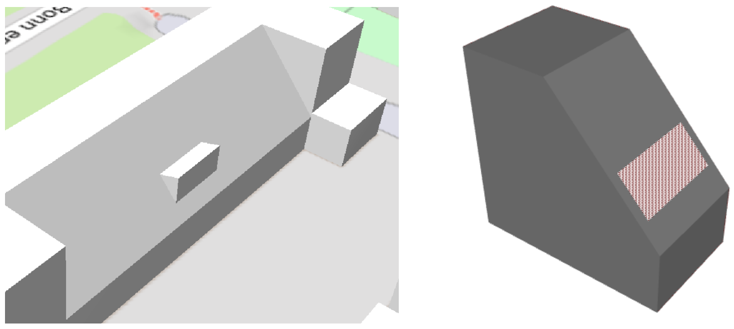


Figure 3.9: Example of an object of class *OuterBuildingInstallation* in the Rotterdam 3D city model: the dormer (left image). On the right, one can see that this results in a hole (inner border) in the roof geometry.

3.3 CHOICE OF THE CONCEPTUAL FRAMEWORK: ATTRIBUTE-BASED AND SEEDED REGION GROWING

This section will illustrate the steps that were taken to establish the conceptual framework for identifying 'deviation' points (the implementation can be found at the end of this chapter). Basing on the background presented in 3.1.1, an exploratory study was conducted. The goal of this one is to identify a number of attributes (derived or not) that might be used for the identification of 'deviations'.

The general method that was chosen to start with is the third (attribute-based) one. The motivation behind this is that next to the point cloud itself, a 3D city model is also available (which is generally not the case in point cloud segmentation). The attribute-based approach offers opportunities to merge information contained in both datasets. Hereby we are mainly referring to the following attributes:
- **positional**: geometry points' coordinates indicate 'where' the surface is inside a reference system.
- **orientation**: the surfaces have an orientation which can be expressed as a normal vector (this one is implicit as surfaces are stored as polygons and thus contain at least three non-aligned points).
- **curvature**: so far, CityGML only supports storage of planar surfaces (Gröger et al., 2012). Therefore, big curved surfaces need to be approximated by several planar surfaces. In contrast, big non-split surfaces indicate planar sections, therefore implicitly containing information about the curvature of surfaces.

3.3.1 Exploratory study

In order to get an idea of the usability of possible derived attributes, a number of them were computed and visualized. Furthermore, this study was performed on a higher and a lower density point cloud (as presented in 3.2). This exploratory study was limited to a single, flat-roofed building containing roof windows that were missing in the 3D model of Rotterdam (address: Albert Plesmanweg 103-110, also see figures 3.10 and 3.11).

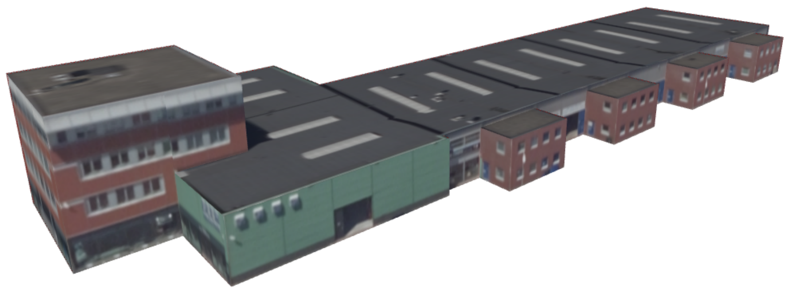


Figure 3.10: Sample building Albert Plesmanweg 103 used for the exploratory study, downloaded from the Rotterdam 3D city model. One can note the roofs which have been modeled as flat planes while roof windows visible in the texture are actually not flat.

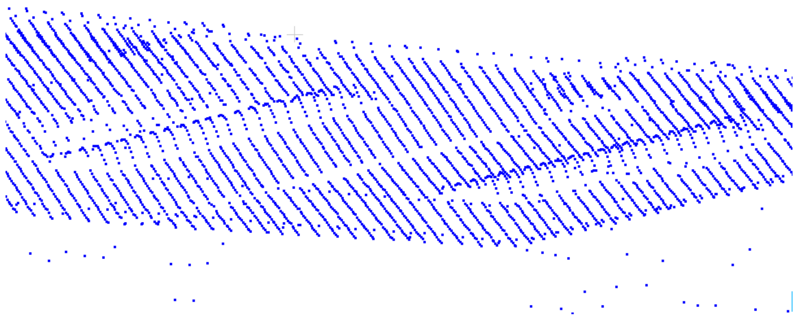


Figure 3.11: Close up zoom on roof windows in the AHN point cloud. Here, the height deviations of the roof windows are indeed present.

Distance to roof plane

As a starting point, a test was performed using only the positional data (thus without derived attributes):
- the height of the main plane of the 3D city model
- the z coordinate of the point cloud

Given that the roof here is flat, these two information are sufficient to calculate the distance to the roof plane. If the roof is not flat, a more complete approach such as explained in part 3.4.2 is needed.

Obviously, small variations within the roof surface and during LiDAR point cloud acquisition apply and a threshold needs to be set in order to distinguish between points representing the 'regular' roof and acquisition-related 'deviations'. The results with thresholds of 30 and 20 cm can be seen below.

As shown in figures 3.14 and 3.13, a cluster of points belonging to the roof lies at a distance from the roof that exceeds 20cm. The most plausible hypothesis is that this is due to the roof in the 3D city model being modeled as entirely flat while it is in reality gently sloped. The question of the accuracy of the 3D city model and its impact will come back in chapter 6.

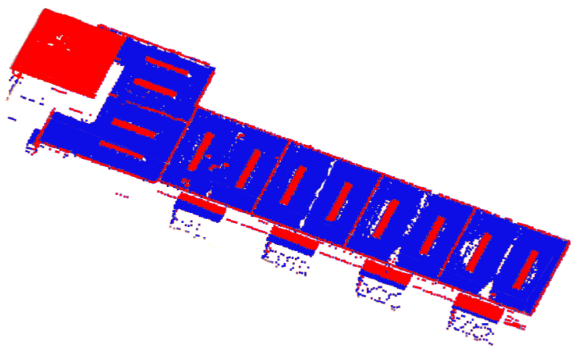


Figure 3.12: Distance of the points with regard to the roof surface of the 3D city model. Points with a distance exceeding 30cm are colored red, others blue. One can see that roof windows are partially identified; points close to the crease edge, to the junction with the roof are missing.

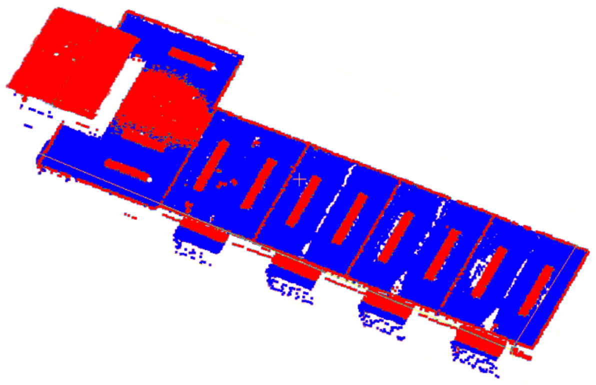


Figure 3.13: Distance of the points with regard to the roof surface of the 3D city model. Points with a distance exceeding 20cm are colored red, others blue. Note the red cluster on the left.

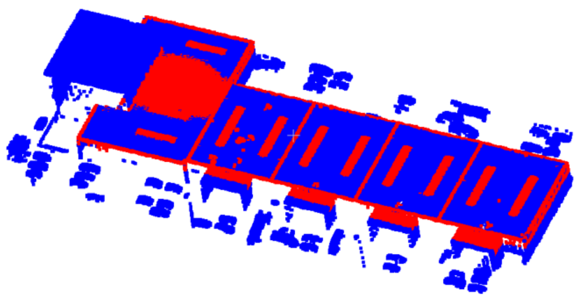


Figure 3.14: Same calculation as in the previous figure, but with the higher density point cloud (red: distance exceeding 20cm, others blue). Just as in the previous example, one can note a cluster of deviations on the left side. These are probably due to a roof slope that is absent in the 3D city model.

Height differences in neighborhood

For this part, a derived attribute was used: the maximum height difference within the neighborhood of each point. This one was computed by selecting the k-nearest neighbors (in this case 3, forming a set of 4 with the existing point) and calculating the biggest difference between the z coordinates (as the test case has a flat roof). Just as in the previous case, a threshold was applied - here 5cm.

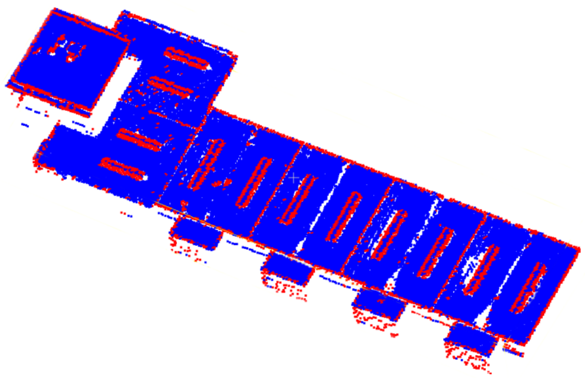


Figure 3.15: Result of the height difference within the neighborhood (KNN=3) of each point. A threshold of 5cm was applied: points above it are colored red, others blue.

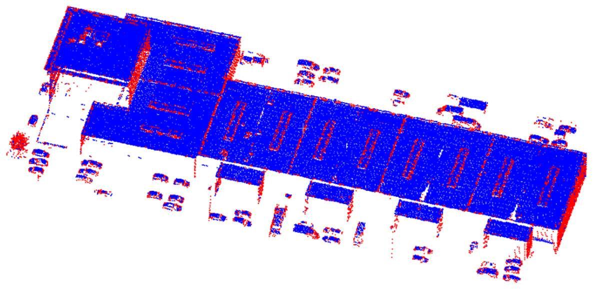


Figure 3.16: Results of the height difference within neighborhood (KNN=3) computation but using a higher definition point cloud. One can note influence of the point density.

One can clearly see that there is complementarity with the previous distance to roof plane attributes. The borders of the roof window, for instance, are identified while the more enclosed parts are not. This is due to the fact that there might be no or only slight height differences within the missing object. In the distance to the roof plane approach, the opposite is the case: the enclosed parts are further from the roof surface and therefore easier to identify.

While measuring height differences in the neighborhood of the points shows potential, applying it to non-planar surfaces requires modifications. In fact, confusion between the actual deviations and changes of roof surface (e.g. slope) itself must be avoided.

Therefore, adapting this to non-flat roofs requires re-projecting the point cloud to work in a local reference system adapted to the roof in question. This implies adding an additional computational layer, which can be avoided as the next part shows.

Normal vectors

One of the most common derived attributes in the field of surface segmentation and reconstruction is the PCA which was mentioned in part 3.1.1. By defining a neighbor with the KNN approach, one can fit a plane to the groups of points. This plane can be defined by a normal vector. By checking whether the normal vector matches the one defining the roof surface of the 3D city model, deviations can be identified, as figure 3.18 shows.

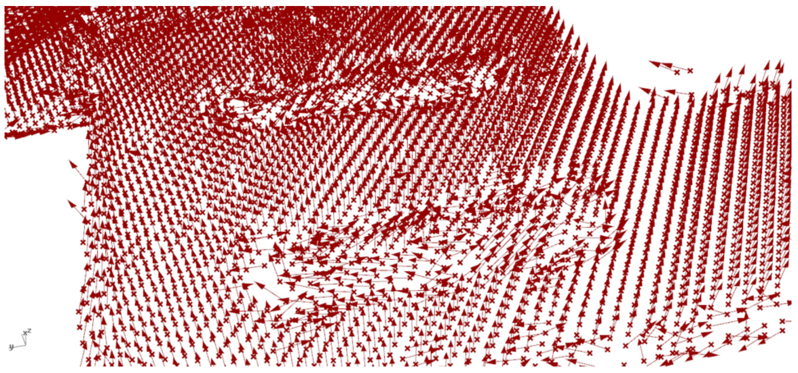


Figure 3.17: Visualization of the normal vectors obtained using principal components analysis on a KNN=10 neighborhood.

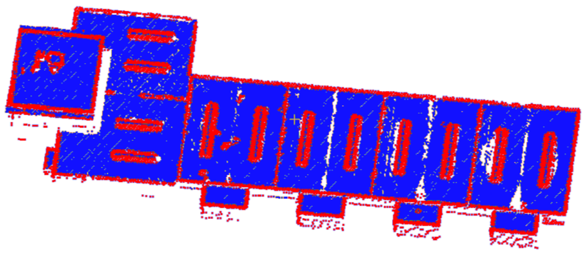


Figure 3.18: Normal vectors obtained by applying principal components analysis on KNN=10 neighborhood. The normal vectors that diverge more than 8 degrees from a vertical one are colored in red, the other ones in blue. Note the blue points at the very inside of the roof windows.

One can note that this method works quite well for the identification of 'deviations' with a surface that has a different orientation than the reference surface of the city model. One advantage is that similarly to the height differences method, it correctly identifies 'deviation' points until a crease edge. However, in the case of non-planar deviations with crease edges, some of the enclosed parts of the deviation will show vectors that are identical to the one of the roof surface, hereby not identifying the entire deviation.

In order to make a choice concerning the k-nearest neighbor setting, a visual evaluation with three different settings was done. As in order to calculate the normal vector using PCA at least three non-aligned points are needed, the minimal version was made with 5 nearest neighbors (which have more chances to include three non-aligned points). In order to see evolutions when the number of nearest neighbors increases (e.g. decreasing impact of noise), two additional values arbitrarily set at 10 and 15 were also tested (see figure 3.19).

As one can see in 3.19, the biggest difference occurs between KNN=5 and KNN=10 settings. In fact, some points which are not located at a deviation on the imagery appear to be red in the first one. As this does not happen with the KNN=10 setting, it is likely to be related to noise in the data.

An important additional observation is that the KNN=10 setting results in slightly bigger regions. Whether there is an over, an underestimation or both taking place is addressed during validation in part 6.4.

Between the KNN=10 and KNN=15 settings, local variations but no clear trend can be observed. As the KNN=10 takes fewer points as an input for the PCA, it leads to a lighter computational load and was thus chosen (in accordance with the criteria named in part 2.1).

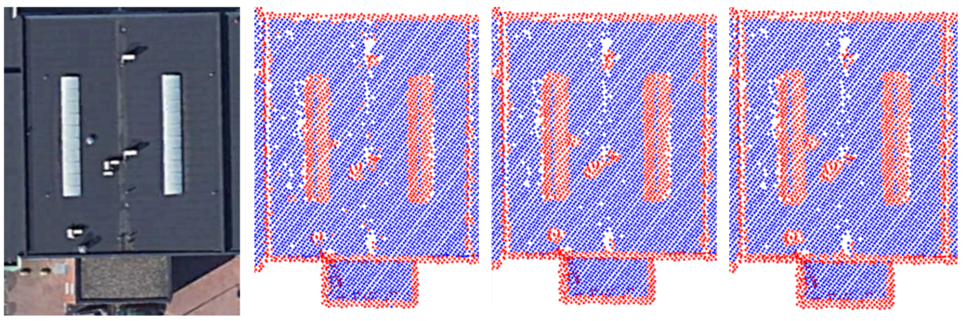
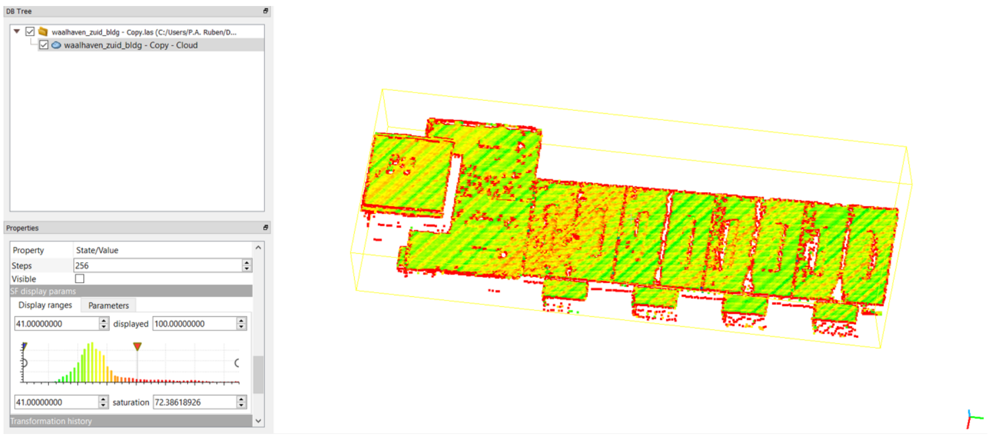


Figure 3.19: Results obtained for normal vectors obtained by using PCA on different KNN neighborhood settings. Normal vectors diverging more than 5 degrees from a vertical one are colored red, others blue. From left to right: satellite image from google earth (taken on 15/01/2019), KNN settings of 5, 10 and 15.

Curvatures

As discussed in part 3.3 another attribute that can be derived from principal components analysis on point neighbors is the curvature. In contrast to the previous explorations, this one was not successful. For both dense and less dense point clouds, it appears that external factors such as the acquisition lines of the LiDAR scanner have a clear impact on the curvature values (see figure 3.20 and 3.22. This is also the case for higher KNN settings where crease edges do not lead to curvature changes - the values might be less spread but the effect of scan lines is still clearly visible (see 3.21).



DB Tree

waalhaven_zuid_bldg - Copy.las (C:/Users/P.A. Ruben/D...

waalhaven_zuid_bldg - Copy - Cloud

Properties

Property	State/Value
Steps	256
Visible	☐

SF display params

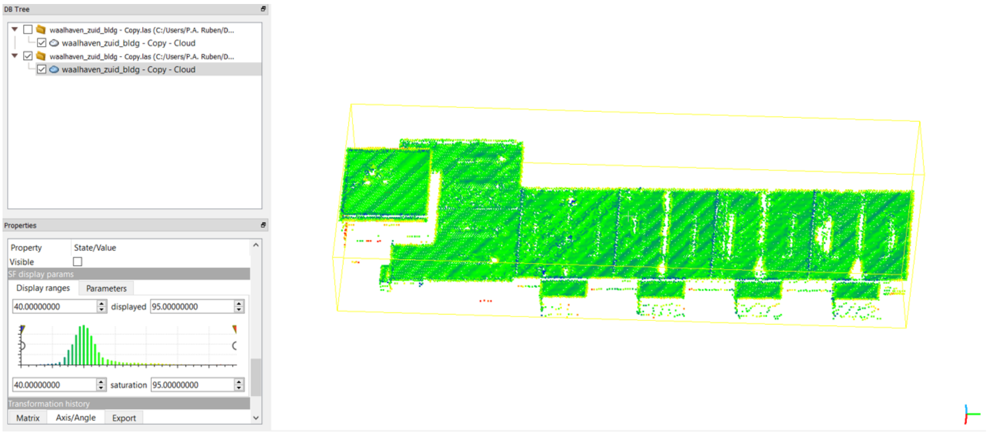
Display ranges | Parameters

41.00000000 displayed 100.00000000

41.00000000 saturation 72.38618926

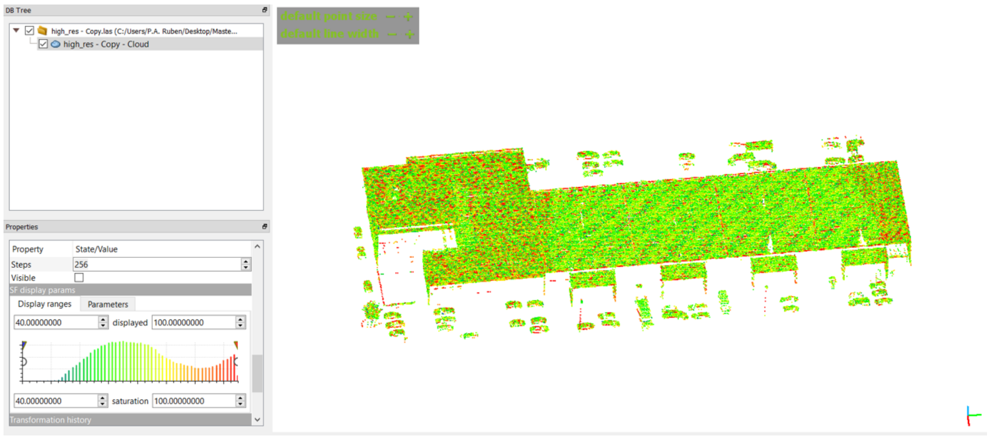
Transformation history

Figure 3.20: Curvature values as computed for each point with a KNN=5 setting. One can see curvature values change along some of the roof window crease edges. Other changes with smaller but still considerable magnitudes seem to be related to the scan lines and noise. Note: values on the left are curvature × 100


DB Tree
waalhaven_zuid_bldg - Copy.las (C:/Users/P.A. Ruben/D...
waalhaven_zuid_bldg - Copy - Cloud
waalhaven_zuid_bldg - Copy.las (C:/Users/P.A. Ruben/D...
waalhaven_zuid_bldg - Copy - Cloud
Properties
SF display params
Display ranges
Parameters
Transformation history
Matrix
Axis/Angle
Export

Property | State/Value
Visible | ☐
40.00000000 displayed 95.00000000
40.00000000 saturation 95.00000000

Figure 3.21: Curvature values obtained by applying principal components analysis on a KNN=20 neighborhood. Note the red points at the very inside of the roof windows. Note: values on the left are curvature × 100



Property | State/Value
Steps | 256
Visible | ☐
40.00000000 displayed 100.00000000
40.00000000 saturation 100.00000000

Figure 3.22: Curvature values obtained by applying principal components analysis on KNN=5 neighborhood on the higher density point cloud. Note the red points at the very inside of the roof windows. Note: values on the left are curvature × 100

While in some cases a curvature change occurs at the border of 'deviations', this test clearly shows that it will be hard to distinguish between value changes induced by noise and value changes induced by 'deviations'. As to the human eye, both phenomena are mixed, a threshold is assumed to be sufficient to tackle this. Moreover, one might note that not all crease edges show curvature changes - the suitability of this approach (especially with regard to the completeness criteria defined in part 2.1) is weaker than the previous ones.

3.3.2 Definition of 'deviations' and choice of the approach

In order to allow automated identification, 'deviations' need a clear definition with a strict logical framework. Based on the previous observations and the criteria established in 2.1 (completeness and computational load), it has been chosen to define 'deviations' as a group of points fulfilling the following two criteria:
a) with regard to the roof surface position (of the 3D city model), at least one point is located at a distance that exceeds a given threshold.
b) any point that is not exceeding the distance threshold mentioned in the first point must have a vicinity whose normal vector diverges more than a given threshold from the normal vector of the roof surface above which it lies. The vicinity is

defined as the k-nearest neighbors around a point. The normal vector of the latter is obtained by applying a principal components analysis to the group of points.

As two criteria exist, a remaining question to address is the order in which the groups of 'deviation' points (or regions) are constructed. One can note that the first part of the definition requires only additions, subtractions, and one division (see part 3.4.2). In contrast, the second part (see part 3.1.1) involves the construction of a covariance matrix and the extraction of eigenvectors which is more complex [4] and increases the computational load.

For this reason, a similar approach to the two-step one used by (Willenborg et al., 2018) in his research on enrichment using a 3D mesh is used. First, seed points are selected by using a distance threshold (attribute-based method). These seed points then serve as an input for a region growing approach, which extends the set using an orientation check (seeded region growing method). Resulting from this, a third criteria can be established:
c) while the points of a region do not necessarily have to be connected, any point belonging to a region is among the k-nearest neighbors of at least one other point of the region (except for regions composed of strictly one point).

3.4 IMPLEMENTATION: IDENTIFICATION OF SEEDS AND REGION GROWING

In this section, the implementation of the approach that was established in 3.3.2 is discussed. Globally, three phases can be observed:
- first, the input data needs to be retrieved and for each roof surface, a region of interest is defined to limit the part of the point cloud to be processed (in part 3.4.1).
- second, seed points which are located at more than a given vertical distance from the roof surface need to be defined (in part 3.4.2).
- finally, the region growing approach is applied using PCA (in part 3.4.3).

The two first phases are described in the algorithm 1 while the last one is described in algorithm 2 in part 3.4.3.

4 covering this in-depth is beyond the scope of the thesis. For more details, see: https://www.scss.tcd.ie/~dahyotr/CS1BA1/SolutionEigen.pdf

Algorithm 3.1: DEVIATION IDENTIFIER (*model*, *cloud*, $threshold_{dist}$, max_{dist}, min_{dist})

Input: A semantic 3D model of a city or neighborhood (e.g. in CityGML format) *model*; a point cloud covering the area of the latter *cloud*; $threshold_{dist}$ for the absolute distance threshold for deviations; max_{dist} and min_{dist} settings for the minimum and maximum distances from the roof taken into account

Output: *deviation_lists*, a list of 'deviation' point lists, each for a given roof surface

```
1  Initialization;
2  for $building \in model$ do
3      retrieve the bounding box $building_{bbox}$;
4      $building_{cloud} \leftarrow$ crop $cloud \cap building_{bbox}$;
5      retrieve $building_{roof\,surfaces}$;
6      for $roof_{surface} \in building_{roof\_surfaces}$ do
7          calculate normal vector $\vec{n}_{surface}$ ( using PCA on points $\in roof_{surface}$);
8          crop $building_{cloud} \cap roof_{surface}$ (2D) $\rightarrow cloud_{roof\,surface}$;
9          Processing;
10         for $point \in cloud_{roof\,surface}$ do
11             $point_{vert\,distance} \leftarrow$ vertical distance to roof;
12             if $min_{dist} < point_{vert\,distance} < max_{dist}$ and
                 $|point_{vert\,distance}| > threshold_{dist}$ then
13                 append $point_{index}$ to $deviation\_list$;
14             end
15         end
16         append $roof\_surface_{deviation\_list}$ to $deviation\_lists$;
17     end
18 end
19 Storage;
20 return $deviation\_lists$;
```

3.4.1 Retrieval of input data using *3DCityDB* and *laspy*

As mentioned in part 3.2, the 3D city model of Rotterdam was obtained in CityGML format and the AHN point cloud in *.laz* format (which is a zipped version of the *.las* format).

In order to facilitate efficient data retrieval (a total of 31 CityGML files were provided), the software *Application Domain Extension (3DCityDB)* 3.3.2 was used. This is a database building in *Structured Querying Language (SQL)* language which can be freely downloaded from www.3dcitydb.org. It basically translates the CityGML standard into a relational database by inserting the different types of classes into tables and using external keys to establish links. For practical reasons, the energy ADE extension was also installed as it provides views which are not present as such in native 3DCityDB. The connection between the main python script and the 3DCityDB database was implemented using the *psycopg* 2.8.3 library.

Within this framework, the query in figure 3.23 was used to extract the list of buildings (highest level class, thus including aggregations of building parts) and their respective bounding boxes (line 3 in the pseudo-code). By using the condition $'objectclass\_id = 26'$ it is guaranteed that the retrieved data is for the class 'building' (codes defined in table citydb.objectclass). Using the expression $'ST\_AsTest(envelope)'$, the bounding box is retrieved as a Well known text format (WKT) string (the latter are automatically generated when importing data into 3DCityDB).

```sql
select gmlid, ST_AsText(envelope)
from cityobject
where objectclass_id = 26;
```

Figure 3.23: SQL statement used to retrieve all lowest level building/building part ids and bounding boxes of the model, using 3DCityDB.

Using the bounding box that is obtained, a rather big initial point cloud can be cropped to the extents of the building (line 4 in the pseudo-code) by comparing the points' x and y coordinates to the minimum and maximum coordinates of the bounding box (using the *laspy* library). Hereby, a computationally efficient reduction of the number of points is provided and the cropped point cloud stored for further usage(s).
In the next step, all roof surfaces of a building need to be retrieved (line 6 in the pseudo-code). This is done by using the SQL code shown in figure 3.24. Reading it from bottom to top, one can see the following steps:
- the original '*building_id*' is converted into an internal id used by 3DCityDB
- all the *building* and *BuildingPart* entities which belong to that id are retrieved (using '*where building_root_id in*')
- all the surface identifiers belonging to roofs ('$objectclass\_id = 33$', codes defined in table citydb.objectclass) and to the entities of the previous step ('*building_id in*') are retrieved. This step involves a view ($citydb\_view.thematic\_surface$) which is provided by installing the Energy ADE.
- finally, using the surface identifiers, the roof surface geometries (as a WKT string) and their identifiers (identical to the ones provided in the input file) are retrieved using '$select\ ST\_AsText(geometry), gmlid$'.

```sql
select ST_AsText(geometry), gmlid
from surface_geometry
where parent_id in (
select lod2_multi_surface_id
from citydb_view.thematic_surface
where objectclass_id=33 and building_id in (
select id
from building
where building_root_id in (
select id
from cityobject
where gmlid='building_id')));
```

Figure 3.24: SQL statement used to retrieve all roof surfaces of a building, using 3DCityDB.

Following this second step, so-called regions of interest can be defined for each of the roof surfaces. These regions of interests are obtained in two steps, by further cropping the building-specific point cloud that was previously prepared. First of all, the *z* coordinate is omitted (thus in 2D) and all points that would lie inside the roof surface polygon are selected (see line 8 in the pseudo-code). Second, all points that lie inside a vertical buffer of the roof surface are selected (the buffer is thus not oriented in the direction of the roof surface normal). This buffer goes as a far as a min_{dist} below and max_{dist} above the roof surface (see line 12 in the pseudo-code). In practice, values of respectively 1 and 20m were used here (see figure 3.25). The usage of such a buffer (rather than just letting the region of interest vertically going to infinity) is motivated by the potential presence of outliers inside and outside buildings (such as identified in part 3.2.1).

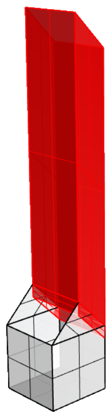


Figure 3.25: Illustration of the region of interest reaching from 1 to +20m, vertically from the roof surface.

3.4.2 Calculation of the vertical distance for seed selection

In order to allow the selection of points inside a buffer and, more importantly, the selection of seed points, it is necessary to calculate the vertical distance between the points and the roof surface (see line 11) in the pseudo-code). This can be implemented using the following trigonometric reasoning:
To calculate the vertical distance between a 3D point and a 3D roof surface, one first has to calculate the shortest distance. For these calculations, the 3D roof surface needs to be defined using a 3D point and a normal vector (using PCA, this can be calculated from the set of coplanar points as which the 3D roof surface is initially stored).

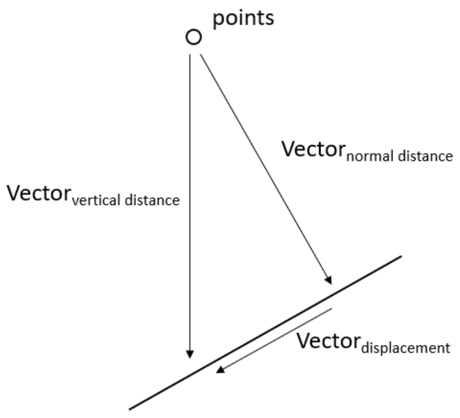

points
Vector$_{\text{normal distance}}$
Vector$_{\text{vertical distance}}$
Vector$_{\text{displacement}}$

Figure 3.26: Illustration of the trigonometric equation to calculate the vertical distance.

$$\overrightarrow{vector_{roof\,surface\,point-point}} = \begin{bmatrix} X_{point} - X_{roof\,surface\,point} \\ Y_{point} - Y_{roof\,surface\,point} \\ Z_{point} - Z_{roof\,surface\,point} \end{bmatrix} \tag{3.4}$$

$$shortest\,distance = \overrightarrow{vector_{roof\,surface\,point-point}} \bullet \overrightarrow{normal_v ector_{surface}} \tag{3.5}$$

Once the shortest distance has been calculated, the vertical distance can be calculated using a trigonometric equation. In fact, the normal (shortest) distance vector, the vertical distance vector and a third displacement vector connecting the latter form a right-angled triangle. In this triangle, two equations can be established. First of all, the vertical vector is the sum of the normal distance vector and the displacement vector:

$$\overrightarrow{vector_{normal\ distance}} + \overrightarrow{vector_{displacement}} = \overrightarrow{vector_{vetical\ distance}} \tag{3.6}$$

Furthermore, the vertical distance vector being vertical, we know that there are no x and y components:

$$\overrightarrow{vector_{vetical\ distance}} = \begin{bmatrix} 0 \\ 0 \\ Z_{vd} \end{bmatrix} \tag{3.7}$$

This results in the following three equations:

$$\begin{bmatrix} X_{nd} + X_{disp} = X_{vd} = 0 \\ Y_{nd} + Y_{disp} = Y_{vd} = 0 \\ Z_{nd} + Z_{disp} = Z_{vd} \end{bmatrix} \tag{3.8}$$

The only unknown to be found in order to determine the vertical distance (the length of the vertical distance vector) is thus the Z_{disp}.
Equation 3.9 relies on the fact that the normal distance vector and the shortest distance vector are orthogonal. In fact, as the displacement vector connects two points (the normal and the vertical projection of the 3D point on the roof surface) which both lie in the plane of the roof surface. By definition it is thus orthogonal to the normal vector of that same plane (and thus to the normal distance vector too).

$$\overrightarrow{vector_{normal\ distance}} \bullet \overrightarrow{vector_{displacement}} = 0 \tag{3.9}$$

Which can be developed as follows:

$$X_{nd} * X_{disp} + Y_{nd} * Y_{disp} + Z_{nd} * Z_{disp} = 0 \tag{3.10}$$

Using the statements of equation 3.8, one obtains:

$$Z_{disp} = \frac{-X_{nd}^2 - Y_{nd}^2}{Z_{nd}} \tag{3.11}$$

and thus the vertical distance:

$$Z_{vd} = Z_{nd} + Z_{disp} = Z_{nd} + \frac{-X_{nd}^2 - Y_{nd}^2}{Z_{nd}} \tag{3.12}$$

3.4.3 Region growing

The second part of the implementation consists of the region growing approach, described in algorithm 2. This one uses the previously identified seed points as an input and can be described with the following steps:
- first, a kd-tree of all points lying in the region of interest is created and a subset of the seed points is selected (line 2 and following).
- second, for each of the seed points, the 10-nearest neighbours are identified (line 8).
- third, if these points have not been identified earlier, a plane is fitted on the 10-nearest neighbors of that specific point (line 17, the value chosen basing on the exploration study presented in part 3.3.1). Then the normal vectors are compared (line 20) to determine whether the point is a 'deviation' and the region can be extended to it or not.

- finally, if this process has been followed for the subset of seeds, a check is performed to make sure that all seeds have been assigned to a region before terminating the algorithm (line 28).

An example illustration of the identification of distance deviations and the subsequent region growing can be found in figure 3.27. The building used here is the same as in part 3.3.1(address: Albert Plesmanweg 103-110).

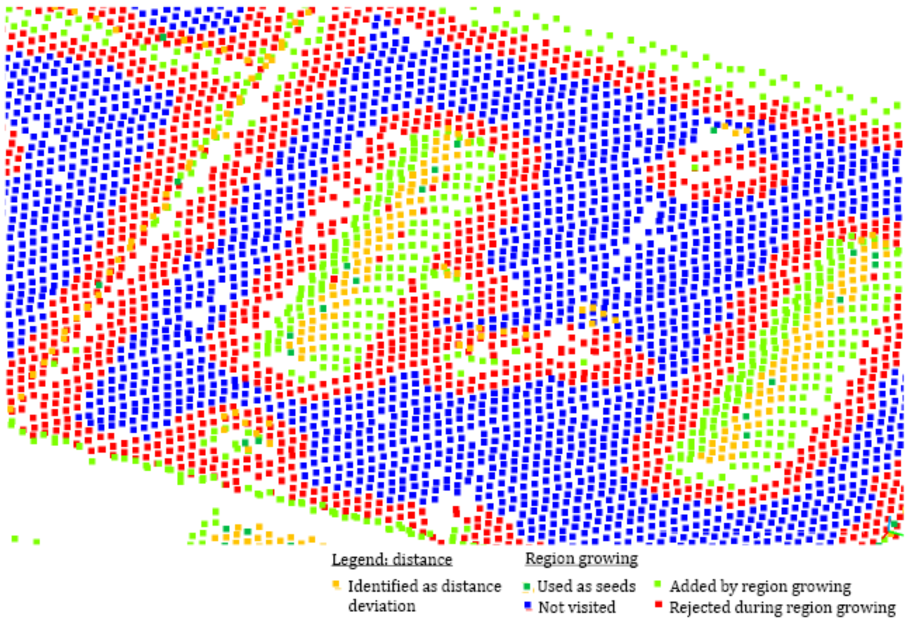


Figure 3.27: Illustration of the distance deviation identification and subsequent region growing approach(settings used: 40cm distance tolerance, 5.73 degrees angular tolerance and 10 for both KNN settings).

Algorithm 3.2: REGION GROWER ($roof_{surface}$,$\vec{n}_{surface}$, $cloud_{roof\,surface}$, $deviation\_list$, $roof\,surface_{deviation\,list}$, $threshold_{angle}$, $KNN_{plane\,fitting}$, $KNN_{growing}$)

Input: The region growing builds upon variables identified in 1: A roof surface $roof_{surface}$ represented by a list of coordinates, the normal vector $\vec{n}_{surface}$ belonging to the previous surface; a list of points $cloud_{roof\,surface}$ located inside the roof surface in 2D, the list of deviations $roof\,surface_{deviation\,list}$ previously identified, the orientation tolerance $threshold_{angle}$, the KNN setting for fitting the local plane $KNN_{plane\,fitting}$, the KNN setting for growing the region $KNN_{growing}$

Output: A list of deviation regions $deviation_{regions}$, each deviation region $region_{points}$ being a list of point indexes itself.

```
1  Initialization;
2  build a kd-tree of $cloud_{roof\,surface}$;
3  $seed\,stack \rightarrow$ every 10th point in $roof\,surface_{deviation\,list}$;
4  Processing;
5  while $seed\,stack \neq \emptyset$ do
6      pop a $point$ from $seed\,stack \rightarrow region_{points}$;
7      while $non\ visited\ point \in region_{points}$ do
8          $point_{neighbors} \leftarrow KNN_{growing}$-nearest neighbors of $point$ (using the kd-tree);
9          for $neighbor_{point} \in region_{points}$ do
10             if $neighbor_{point} \in region_{points}$ then
11                 pass;
12             else if $neighbor_{point} \in roof\,surface_{deviation\,list}$ then
13                 append $neighbor_{point}$ to $region_{points}$;
14                 if $neighbor_{point} \in seed\,stack$ then
15                     remove $neighbor_{point}$ from $seed\,stack$;
16             else
17                 $neighbor_{vicinity} \leftarrow KNN_{plane\,fitting}$-nearest neighbors of $neighbor_{point}$ (using a kd-tree); obtain normal vector $\vec{n}_{vicinity}$ of $neighbor_{vicinity}$ (using PCA);
18                 $deviation_{angle} \leftarrow \vec{n}_{vicinity} \bullet \vec{n}_{surface}$;
19                 if $deviation_{angle} < cos(threshold_{angle})$ then
20                     append $neighbor_{point}$ to $region_{points}$;
21                     (mark $point$ as visited);
22                 else
23                     pass;
24             (mark $point$ as visited);
25         end
26     end
27     Add $region_{points}$ to $deviation_{regions}$;
28     if $stack = \emptyset$ then
29         for $point_{deviation} \in roof\,surface_{deviation\,list}$ do
30             if $point_{deviation} \notin deviation_{regions}$ then
31                 Add $point_{deviation}$ to $seed\,stack$
32             end
33         end
34     end
35 end
36 Storage;
37 return $deviation_{regions}$;
```

3.4.4 Variables

It should be noted that the approach presented in algorithms 1 and 2 provides two main variables:
- a threshold setting the distance tolerance (between the point cloud and the 3D city model) for the identification of seed points ($threshold_{dist}$ in line 12 of algorithm 1).
- a threshold setting the angle tolerance used during the region growing (to distinguish between planes deviating and matching with regard to the roof surface). This one can be found as $threshold_{angle}$ in line 19 of algorithm 2.

Also, two minor variables min_{dist} and max_{dist} can be found in line 12 of algorithm 1. These ones are essentially used to describe the region of interest in the vicinity of the roof surface.

Furthermore, two additional variables for which (mainly for practical reasons which are elaborated in part 6.1.1) a usable constant has been identified exist:
- the number of vicinity points used to obtain the local normal vector using PCA ($KNN_{growing}$ in line 17 of algorithm 2). Basing on the exploration performed in part 3.3.1, this one will be set 10. It should be noted that this exploration is specific to the dataset used in this study, and the resulting setting might thus differ if other point cloud datasets are used.
- the number of nearest neighbors used for the region growing ($KNN_{growing}$ in line 8 of algorithm 2). As one can see in figure 3.27, the identified regions (green and yellow) are surrounded by a layer of refused points (in red). With regard to the criteria named in part 2.1, this layer needs to provide a sufficient margin for completeness (sufficient points should be checked) without leading to unnecessary computational load. Based on a visual evaluation, one can note that the red layer obtained in figure 3.27 with a setting of 10 has a thickness of at least 2 - 4 points, which is considered a good balance to fulfill the two criteria. Therefore, this setting of 10 will be used in this research. Here too, one should note that this is specific to the chosen point cloud dataset.

3.5 CHAPTER SUMMARY AND LINK TO ASSUMPTIONS MADE

This chapter has explored and identified a strategy for the identification of 'deviation' points by comparing two datasets: a LiDAR point cloud and a city model. First, an overview of existing background research in the field of 3D surface reconstruction was formulated. Along with this one, recurring challenges and relevant mathematical tools were mentioned. Also, an overview of the relevant semantics used by the CityGML format used has been given.

INPUT DATASET SPECIFICATION AND CHOICE A second part was dedicated to the specification of the input datasets, covering, among others, choices that were made within the CityGML framework. As two-point cloud datasets were available, a qualitative comparison using two sample buildings was performed and allowed a justified choice.

CONCEPTUAL FRAMEWORK: ATTRIBUTE-BASED AND SEEDED REGION GROWING In a third step, an exploratory study led to establishing the conceptual framework, namely 'attribute-based and seeded region growing'. By computing several attributes for a sample building (distance to roof plane, height differences within KNN neighborhoods, comparison of KNN cluster and roof plane normal vectors, and comparison of KNN cluster and roof plane curvatures), two relevant ones were identified. Considering both completeness and computational efficiency criteria, a seeded region growing method was chosen and implemented.

IMPLEMENTATION OF THE FRAMEWORK The implemented method consists of mainly two parts: first, the vertical distance of all points is calculated and where a given threshold is exceeded, the point is labeled as seed. These seeds are then used to perform a region growing within which the local normal vector (of a KNN cluster) is calculated using PCA. The local normal vector is compared to the one of the roof surface and if the difference is too big, the points are added to the 'deviation' regions.

During implementation, a number of critical details were addressed. Using a python framework, the CityGML roof surfaces were retrieved from a SQL database. Using a kd-tree, the point cloud was clipped before computation. Furthermore, a region of interest was defined in order to limit the impact of outlier points located inside a building.

VARIABLES USED In total, four variables were identified. The possibility to use a constant was studied with regard to the limited time-wise budget, mainly critical in chapter 6. Using the sample building, usable constants for the KNN settings during region growing and local plane fitting were successfully identified. Therefore, the two settings to focus on are the height and the angle tolerance thresholds.

4 FROM POINTS TO SURFACES: EXTRACTION OF GEOMETRIES

Unlike ”length”, ”volume”, ”position” or ”angle”,the geometric notion of ”shape” has no formal meaning (Edelsbrunner and Mücke, 1994). Therefore, a study of the different possibilities, the different definitions that exist to obtain a shape using a number of input points is necessary.

This is what this chapter aims at, starting with an overview of relevant background knowledge (part 4.1). Basing on this literature review, two interesting options will be selected and further explored, ultimately leading to the choice of the Voronoi diagram as the theoretical framework (part 4.2).

This chapter will be closed with the implementation of the chosen framework (part 4.3). Here, the usage made of the *scipy.spatial.Voronoi* library and the necessary improvements that were made will be detailed.

The concepts discussed in part 4.1 illustrate different approaches but are also subject to overlap. The notion of the convex hull will, for instance, come back in the boundary extraction approach. Different approaches presented in this chapter indeed build upon each other.

4.1 BACKGROUND: DEFINITION OF SHAPES BY POINT SETS

The background work of this section covers the question of how to transform a set of points into a geometric shape. As in the origin they all tackle a similar problem, the following premises will be shared:

- while in the previous chapter 3, the approaches mostly applied to 3D space, this is different here. As mentioned in the introduction (see part 1.3), the scope of this project is to produce a semantic enrichment. In other words, this one does not need to represent a full 3D geometric addition to the existing 3D city model. Therefore, from this part on, the topics covered will exclusively be referring to 2D space. The conversion from 3D space to 2D space takes place by simply omitting the 3rd spatial dimension of the points, the z coordinate [1].

- The entire set of points within the region of interest of a roof surface P (see part 3.4.1) surface is denoted R. $R = \{r_1, r_2, r_n\}$ are the points belonging to the set of the region of interest, with $r_n \in R$ and $R \cap P$.

- The points identified as 'deviations' are a subset denoted S with $S \in R$. $S = \{s_1, s_2, s_n\}$ are the points belonging to that subset.

Furthermore, it should be noted that the approaches illustrated here are not unrelated to each other. For instance, the convex hull discussed hereafter is reused as a tool in the boundary extraction method (see part 4.1.2). These are thus not different taxonomies in the sense of part 3.1.1 but rather different versions of a perspective on transforming a set of points into ”shapes”.

1 This creates an orthogonal projection which distorts the shape with regard to the roof plane (if this one is not flat). As the obtained geometries will ultimately be projected onto a horizontal image plane (see chapter 5), the distortion can be disregarded.

Convex hulls

A first option to translate a group of points into a geometric shape is the convex hull. For a set of points S, the convex hull is the smallest geometric shape containing all points of S. Furthermore, none of the (straight) lines connecting any two points of S are allowed to intersect the edges of the convex hull. This implies that all the angles formed by the consecutive edges of the convex hull are smaller or equal to 180 ° (see figure 4.1) (De Berg et al., 1997).

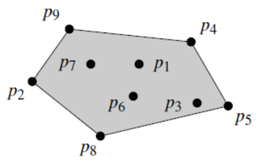


Figure 4.1: Example of a convex shape for a set P of input points (De Berg et al., 1997).

A major drawback of convex hulls is the fact that their usage is limited to convex shapes. Any concave shape cannot be obtained by algorithms computing convex hulls. To explain this, one might think about the 'rubber band' analogy: if one puts a rubber band around a L-shape (or points sampling that L-shape), the rubber band will not wrap around all edges of the L-shape. Instead, it will take a convex shape containing the original concave one. This also excludes the extraction of any shapes with holes. Furthermore at least three non-aligned points are required.

4.1.1 Concave hulls and alpha-shapes

The concept of the convex hull was already touched in the previous part about concave hulls. While the difference is clear, there is interestingly no universal approach for obtaining the concave hull of a set of points. The reason for this is that there is not just a single way to define the concave hull with regard to the edges composing it. There can be several concave-hulls for the same set of points (see 4.2).

One of the approaches is the α-hull which is composed of the connections between points for which a disk of radius $1/\alpha$ can be drawn so that:
- the two points lie on the border of the disk.
- the disk contains the entire point set (Edelsbrunner and Mücke, 1994).

Looking at the implementation level, there is always a need to further specify the definitions in order to make it applicable. Some identify and connect extreme points (Edelsbrunner et al., 1983), others restrict the angles of the consecutive edges and minimize the area (Asaeedi et al., 2013) or visit the set of points with a specific KNN approach (Moreira and Santos, 2007). A drawback of these approaches is that all except the KNN one require a setting, the so-called α. As the point cloud density is not a constant (see part 3.1.1), this one would need to be set differently for each situation. Another option would be to use a higher α setting than needed, but this would reduce the support for holes (which are not supported at all by the KNN approach of (Moreira and Santos, 2007) and the area minimizing by (Asaeedi et al., 2013)). Finally, one should note that, just as for the convex hull, at least three non-aligned points are required.

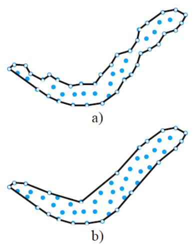


Figure 4.2: Example of two concave shapes obtained using the KNN approach (Moreira and Santos, 2007).

4.1.2 Boundary extraction: labeling and minimum spanning tree

Similarly to 3D surface reconstruction (as presented in 3.1.1), there is also the possibility to identify the border of the shape rather than the shape directly. In his paper, Wang and Shan (2009) suggests the usage of local convex hulls for the identification of border points which are later connected by a minimum spanning tree.

For each point in the set, a subset of k-nearest neighbors is identified. Each time, the convex hull of this subset is computed. All points which are inside the convex hull (thus not edge points) are then removed from the list of candidate border points (which initially contains all points). Once all the points are visited, a graph of the remaining (previously candidate) border points is created with a similar KNN approach. Ultimately, this graph is transformed into a minimum spanning tree which contains the border (see figure 4.3 for an illustration).

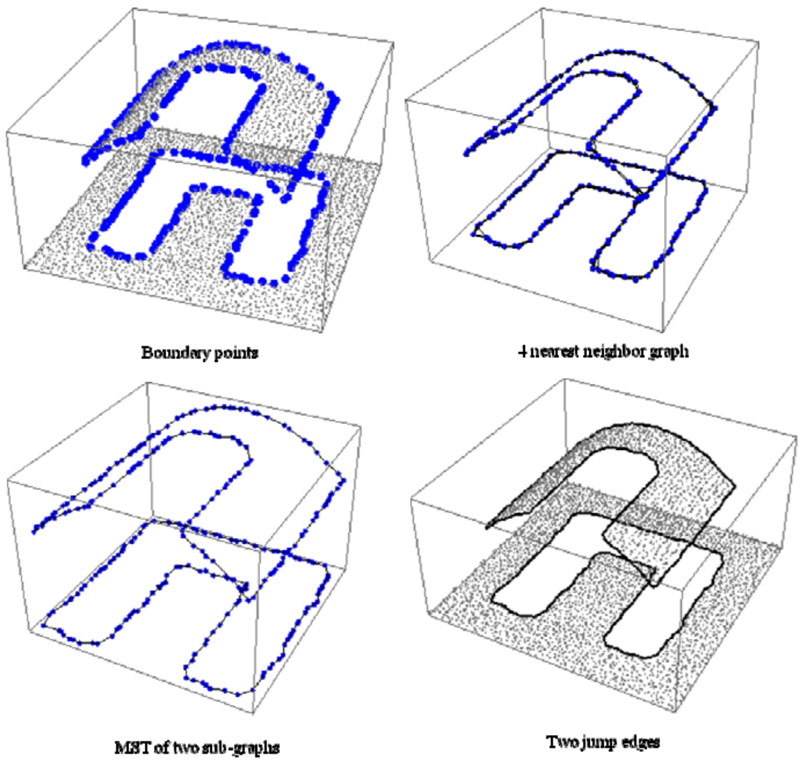


Figure 4.3: Illustration of the boundary extraction process proposed by Wang and Shan (2009).

This method has the advantage that it only requires convexity at a local scale. Concave shapes can thus also be obtained. Furthermore, shapes with holes can be supported if several graphs are supported for connecting and extracting border points. However, the same restriction as for the concave and convex hull applies: at least three non-aligned points are required.

4.1.3 Voronoi diagrams

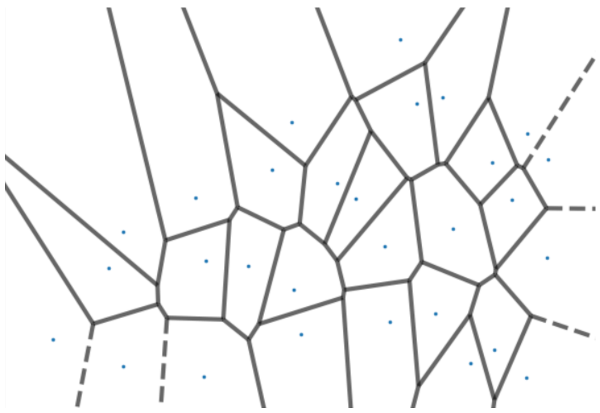


Figure 4.4: Example of the Voronoi diagram obtained for a set of points. Space is divided into cells. Any point lying inside a cell is closer to the respective input point of that cell than to any other input point (in blue).

A further consideration that can be made in the context of this research is that the set of points is actually a set of 'probes'. In fact, the LiDAR system used for the acquisition (as mentioned in part 1.5.2) 'samples' the environment by sending laser rays at regular geometric intervals. Therefore, each obtained point can be seen as representing a part of the original geometry.

One way to translate this perspective into a geometry extraction process, is the Voronoi diagram. The Voronoi diagram does not translate a set of points into a "shape" directly as it does first divide the space occupied by the set of points (see figure 4.4). Formally, it can be defined as follows in 2D:

For any point set $R = \{r_1, r_2, r_n\}$ belonging to a plane P, the Voronoi diagram of R is the subdivision of the plane P into c parts respectively containing exactly one point of R. It has the property that any point $q \in P$ lies inside c_i if and only if $distance(q, r_i) < distance(q, r_j)$ for any $r_j \in R\, where\, j \neq i$ (De Berg et al., 1997; Botsch et al., 2010).

For a subset $S = \{s_1, s_2, s_n\} \in S$, the shape of the points $s_n \in S$ can be obtained by extracting the cells $cells_{deviation} = \sum_{i=1}^{n} c_i$ for any $s_i \in S$. The geometries of $cells_{deviation}$ can then be merged into one, resulting in a "shape" representing the input points S in the context of R (see figure 4.5).

In contrast to the previous methods, this one does take into account the context of the point set. Therefore, it requires other points located around the point set to extract. In this thesis, this is the case, with the exception of points being located close to the edge of a roof surface. This issue will further be addressed in part 4.3.

The shape that is obtained using this approach can be expected to be bigger than the 'strict' one might obtain with the concave hull approach (part 4.1.1). The extent to which it will be bigger depends on the distribution, on the distance between the points of the cloud. In fact, the translation from input points to Voronoi cells does actually perform what is called a 'nearest-neighbor' interpolation. Therefore, the deviation "shape" is the parts of the roof surface that are closer to a deviation point than to a non-deviation point.

An advantage of this approach is that it supports uneven point distribution (e.g. resulting from occlusion), which regularly occurs in point cloud datasets such as the one of this thesis (as shown in part 3.2.1). Other aspects to notice are that it does support geometries with holes and is able to provide a shape from only one point on.

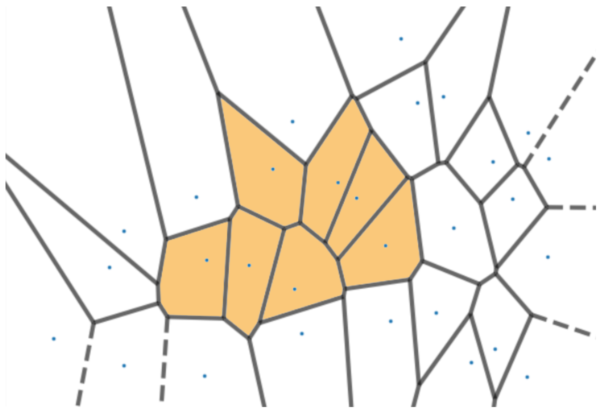


Figure 4.5: For the points to extract (located in the yellow cells), the cells can be extracted and merged into a single geometry.

4.2 CHOICE OF DEFINITION AND CONCEPTUAL FRAMEWORK

4.2.1 Pros and cons with regard to the needs

Using the observations based on literature, a comparison of the different methods has been made using several criteria. An overview can be found in table 4.1. The criteria have been selected in view of the needs of this thesis:

- First, the type of expected geometry. In the case of the convex hull, the resulting shapes will be overestimated. This overestimation is not a fixed one and might deliver strongly erroneous (over-estimating) results for some non-convex shapes. For the concave hull and the boundary extraction approach, a rather strict shape, close to the minimum area enclosing all points can be expected. For the Voronoi diagram approach, the resulting shape will also enclose all points but can be expected to have a buffer corresponding to the nearest neighbor interpolation along the edge.

- Second, the support of geometries with holes is technically possible for all approaches except the convex hull. For the concave hull, some algorithms do not support holes, others do.

- Third, the presence of irregular point densities (one of the challenges mentioned in part 3.1.1) is not expected to be a problem in the case of the convex hull and the Voronoi diagram approach. In the case of the boundary extraction approach, irregular point densities might become a problem if the irregularities are too strong and make it impossible to cover the dataset with a single KNN setting. Furthermore, depending on the α setting used (and the specific algorithm), the concave hull can be affected by low point densities and occlusions (e.g. creating holes at inappropriate locations).

- Fourth, the requirement for settings to be provided by the user. This is only the case for the convex hull and some of the concave hull approaches allow this. The concave hull versions that do not require settings do not support holes though.

- Fifth, the calculation of the geometry by using a subset only is possible for all approaches except the Voronoi diagram. In the case of the Voronoi diagram, points located on the outer border of a set of points lead to non-closed cells. This is a weakness that will be addressed in part 4.2.3.

- Finally, the minimum number of points required by the methods is 3 (non-aligned points) for all except the Voronoi diagram.

Basing on this comparison, the convex hull can be excluded as it is clear that the 'deviations' might also have shapes that are not convex. For the concave hull difficulties in implementing successfully both support of holes and absence of input settings must be expected. This is in contrast with the boundary extraction approach which shows similar results with a different paradigm, thus avoiding the concave shape drawbacks. Finally, for the Voronoi diagram approach - a clear definition does exist and can be reliably implemented.

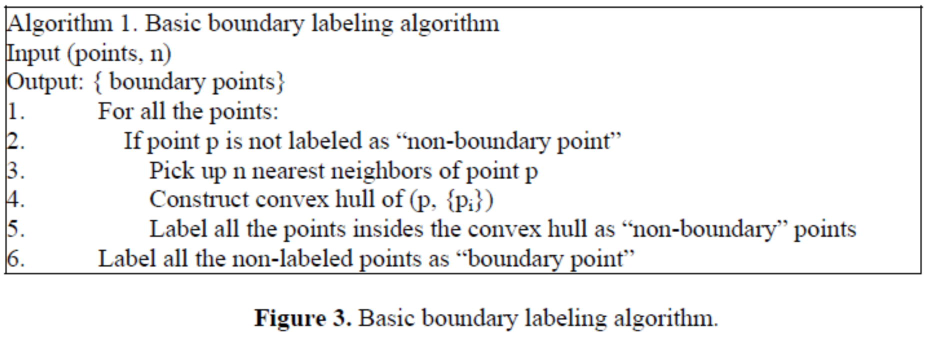
	Point to geometry transformation method			
	Convex hull	Concave hull	Boundary extraction	Voronoi diagram
Expected geometry	Strict, but wrong for non-convex	Strict	Strict	Voronoi cells
Allows holes	No	Yes/No [2]	Yes	Yes
Resistance to irregular density	High	Low	Medium	High
Input settings	No	No/Yes [1]	Yes	No
Calculation with subset only	Yes	Yes	Yes	No
Minimum number of points	3	3	3	1

Table 4.1: Comparison of the different approaches mentioned by the literature study in part 4.1

With these considerations in mind, it was decided to further explore the boundary extraction approach as proposed in (Wang and Shan, 2009). Furthermore, the Voronoi diagram approach will also be explored as its usage of nearest-neighbor interpolation to define the shape shows potential with regard to the actual meaning of the LiDAR acquired point cloud.

4.2.2 Exploratory study: boundary labeling and minimum spanning tree

A first exploratory study was carried out with the boundary labeling and subsequent extraction with a minimum spanning tree approach. Here, the pseudo-codes published by Wang and Shan (2009) (see figures 4.6 and 4.10) were translated into a Python code. One should note that this was performed in two dimensions only as the points were projected onto the same surface beforehand (the third dimension, the z attribute was omitted).



```
Algorithm 1. Basic boundary labeling algorithm
Input (points, n)
Output: { boundary points}
1.    For all the points:
2.        If point p is not labeled as "non-boundary point"
3.            Pick up n nearest neighbors of point p
4.            Construct convex hull of (p, {p_i})
5.            Label all the points insides the convex hull as "non-boundary" points
6.    Label all the non-labeled points as "boundary point"
```

Figure 3. Basic boundary labeling algorithm.

Figure 4.6: First part of the approach published by Wang and Shan (2009). The border points are identified by performing a local convex hull check for each point of the set.

In order to get a better idea of the results, a pilot was performed on the same building as in the exploration in part 3.3.1 (address: Albert Plesmanweg 103-110). An example result of the first part of the approach (the implementation of the pseudo-code in 4.6) can be seen in figure 4.7. The first challenge was identified during the implementation as 'deviation' regions with fewer points than the k setting (the k of KNN used for the local convex hull approach, in the case of this example '10') cannot be subject to this processing (e.g. points that are isolated seeds that

2 The versions that were studied do either not allow holes or require input settings. The ones that do allow holes have a low resistance to irregular density

did not grow into regions). In such cases, all the points belonging to the 'deviation' were de facto labeled as deviations (e.g. the four clusters located between the two windows on figure 4.7 are entirely colored in green).

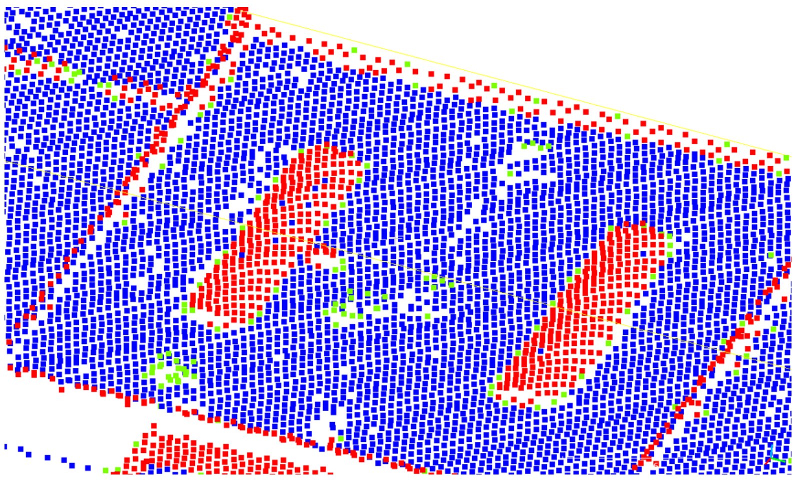


Figure 4.7: Example result obtained by implementing the pseudo-code presented in figure 4.6 (10-nearest neighbor setting). The points that were labeled as 'deviations' are in green if they were labeled as border points too and in red otherwise. The points in blue are the ones of the initial set that were not labeled as 'deviations'.

A more critical issue can be observed for non-convex shapes (e.g. the left window in figure 4.7 which has another object attached to the same region on its right side). In fact, the points located at the intersection of edges forming an angle of more than 180 ° can hardly be labeled as a border point. If we assume that at least one point along these two consecutive edges is among the 10-nearest neighbor set it is excluded that such a point will be on the edge of the convex hull formed by the neighbor set. This makes clear that the usage of the convex hull for testing at a local scale will lead to missing some of these critical points where two edges meet (also see figure 4.8).

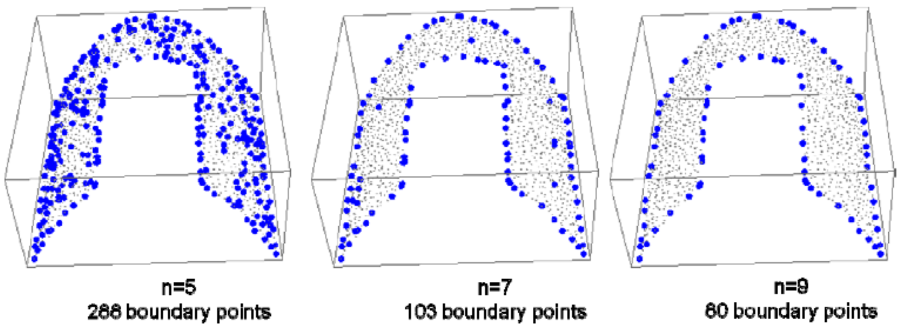


Figure 4.8: Results of the approach shown in figure 4.6 that are shown in Wang and Shan (2009). Similarly to the results shown in figure 4.7, some critical corner points are not identified.

It is important to note that an improved version of the algorithm 4.6 was proposed in the same publication (Wang and Shan, 2009). This one consists of adding a threshold to avoid labeling points close to the local convex hull border as 'non-boundary' points. As this is only an exploration study, it was not implemented here. However, it seems rather unlikely that this modification would allow the identification of the critical points at the intersection of edges forming an angle of more than 180 °. In fact, nothing suggests that these points are even close to the edge of the convex hull formed by their neighbor set. This is in line with the results presented by Wang and Shan (2009) which are shown in figure 4.9.

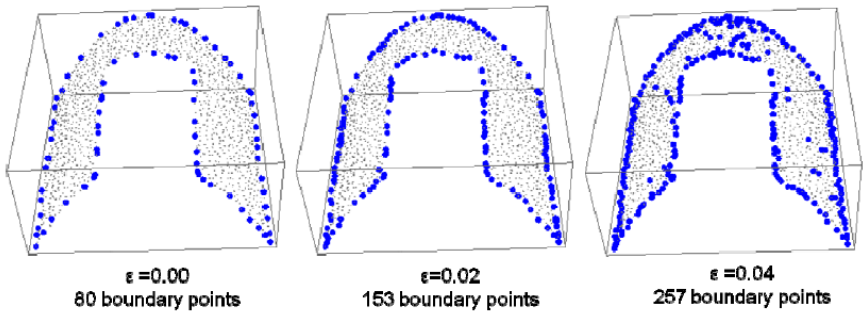
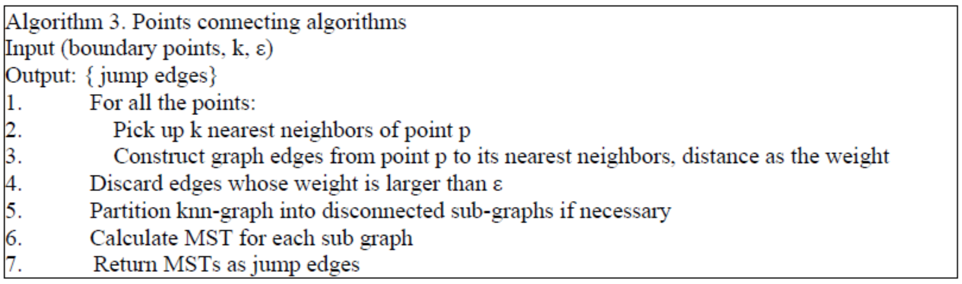


Figure 4.9: Results of the improved approach shown in figure 4.6 that are shown in Wang and Shan (2009). The improved consists of adding a threshold to avoid labeling points close to the local convex hull border as 'non-boundary' points. One can see that some critical corner points are not identified in this version either.



```
Algorithm 3. Points connecting algorithms
Input (boundary points, k, ε)
Output: { jump edges}
1.      For all the points:
2.          Pick up k nearest neighbors of point p
3.          Construct graph edges from point p to its nearest neighbors, distance as the weight
4.      Discard edges whose weight is larger than ε
5.      Partition knn-graph into disconnected sub-graphs if necessary
6.      Calculate MST for each sub graph
7.      Return MSTs as jump edges
```

Figure 4.10: Second part of the approach published by Wang and Shan (2009). The border points are converted into a graph basing on the distance between the k-nearest neighbors of the border point set. If required, the obtained graph is split using a distance threshold (for instance, to split inner borders from the outer border). Finally, the graph is converted into a line geometry by creating a minimum spanning tree.

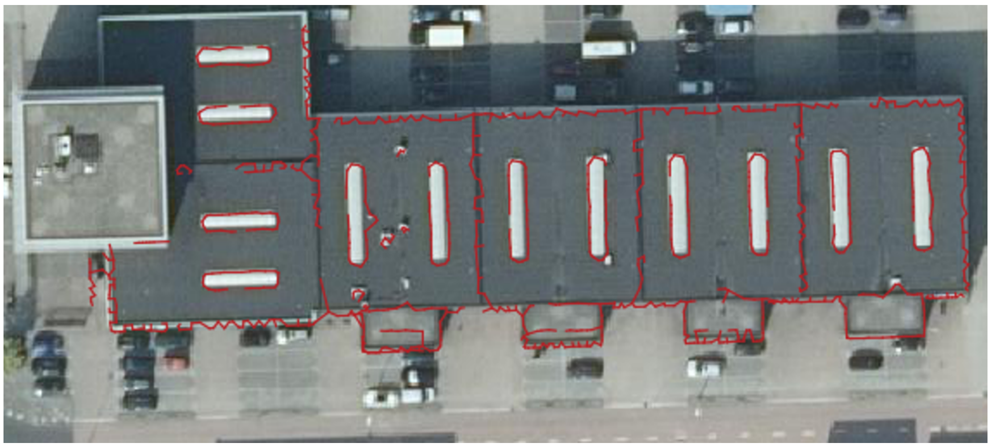


Figure 4.11: Example result obtained by implementing the pseudo-code presented in figure 4.10 to the result of which an extract was presented in 4.7. To allow easier implementation, the line segments were not closed in the cases where a circular line segment should be generated. In some cases, the minimum spanning tree is quite literally a tree: it is not composed of a single line segment but of several connected ones.

Looking at the results of the second part of the exploration (figures 4.10 and 4.11), one can observe the consequence of missing some of the critical corner points. A comparison between the window on the left in figure 4.7 and the first vertical window from the left on figure 4.11 shows that the corners where the object is attached to the 'deviation' region of the window are much smoother than in the point cloud. In the vicinity of these missed corner points, one can observe a shape with an overestimated area.

A second concern of this approach are the results of the second step for shapes with a limited width. A good example of these ones is the roof edges which have a thickness of about 50cm (based on measures taken from aerial images). In these cases, one can see that the minimum spanning tree does connect boundary labeled points of different sides of the edge alternatively, resulting in zig-zag patterns. In some other cases, branches grow out of the mainline segment, leading to geometry with several segments. As the minimum spanning tree works using the distance between border points only, this happens as the opposite boundary points are closer to each other than the consecutive ones. Avoiding this would require the boundary points of one side to be closer to each other, which can only be achieved by lowering the k-nearest neighbor setting of the first step (figure 4.6). Doing this will however lead to wrongly labeling points as boundary points, especially inside shapes with more width (as shown in figures 4.8 and 4.9).

In fact, the k-nearest neighbor setting defines the size of the local neighborhood to be taken into account in the first steps described in figure 4.6. Induced by figures 4.8 and 4.9, one can formulate the following observation:

As the k-nearest neighbor setting decreases, the chances of having wrongly identified boundary points increase.

4.2.3 Exploration study: Voronoi diagrams

In order to allow a justified choice before implementation, the option of the Voronoi diagram was explored as well. The definition of the Voronoi diagram (as presented in part 4.1.3) is unambiguous and several Python libraries do already exist, allowing an easier implementation.

The biggest challenge that was identified so far for the Voronoi diagram is the necessity of the subset to be surrounded by 'non-deviation' points. In fact, for a set of points (see the cell in green on figure 4.12) the points located on the edge (points that are edge vertices of the convex hull of the set) lie in non-closed cells (Botsch et al., 2010).

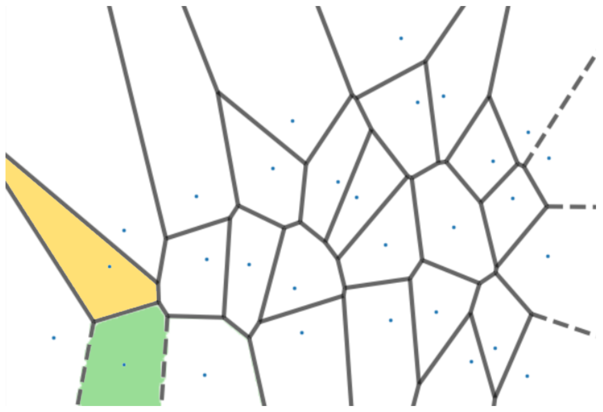


Figure 4.12: Example of the Voronoi diagram obtained for a set of points. One can observe that i) cells of points belonging to the vertices of the convex hull are non-closed and (e.g. green cell) ii) that some closed cells have a vertice located outside the convex hull (e.g. yellow cell).

Another challenge arises by the fact that some closed cells (i.e. the ones belonging to points located close to the convex hull border, but not part of it) have vertices lying outside the set of points (see figure 4.12). The extent to which these ones lie outside can be considerable (see figures 4.12). As can be seen in figure 4.13:

for three points s_i, s_{e1} and s_{e2} of a bigger set S and q_i, q_{e1} and q_{e2} their respective Voronoi cells of the Voronoi diagram Q;

with $\{s_{e1}, s_{e2}\} \in edge\,points \in convex\,hull(S)$; $edge_{e1-e2}$ the line connecting s_{e1} and s_{e2};

the following observation can be made for the point $voronoi\,vertex_{i-e1-e2}$ where q_i, q_{e1} and q_{e2} meet:

as distance $|(s_i - edge_{e1-e2})|$ decreases, distance $|(voronoi\,vertex_{i-e1-e2} - edge_{e1-e2})|$ increases toward infinity.

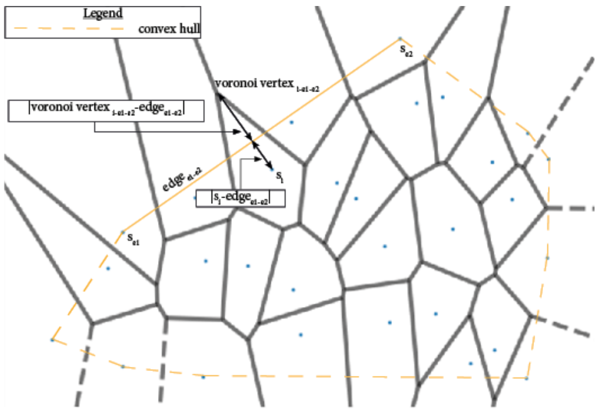


Figure 4.13: '
[Illustration that as distance $|(s_i - edge_{e1-e2})|$ decreases, distance $|(voronoi\,vertex_{i-e1-e2} - edge_{e1-e2})|$ increases toward infinity.]Illustration of the observation that as distance $|(s_i - edge_{e1-e2})|$ decreases, distance $|(voronoi\,vertex_{i-e1-e2} - edge_{e1-e2})|$ increases toward infinity.

In the context of this research, both of these challenges can be addressed by coming back to the original motivation for using the Voronoi diagram: the point in the cloud as a sample of the geometry it lies on. In fact, the point sets that are used here all refer to one roof surface geometry per set (independently of having being labeled as 'deviations' or not). Therefore, it is considered that these points sample the respective roof surface up to the edges.

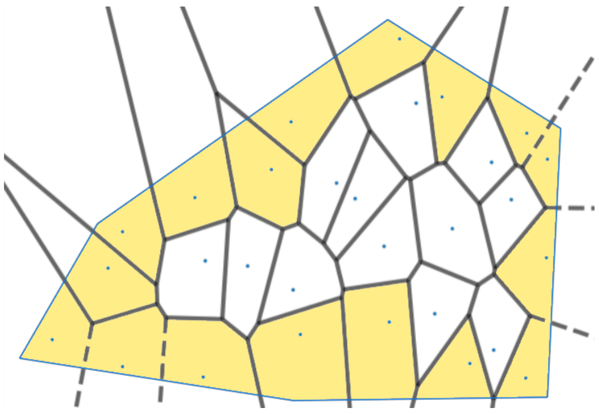


Figure 4.14: Illustration of the usage of a bounding box to limit the extent of non-closed and outlying cells. The roof surface geometry is shown as a blue line and the affected cells are colored in yellow.

By using this approach, the Voronoi diagram is in fact cropped to a bounding box which is the original roof surface (see figure 4.14). Non-closed cells and cells with a vertex outside the bounding box are thus closed by the edges of the bounding box. This approach respects the idea of the nearest neighbor interpolation. Although details are missed by the point cloud acquisition (e.g. in between any two points of the point cloud), this approach is similarly valid at the edges of the point cloud as it is inside the set of points (the size of the cell might be slightly bigger or smaller as the cell is cut by the edge of the bounding box rather than limited by the cell of a neighbor point).

4.2.4 Choice of the theoretical framework: Voronoi diagrams

FURTHER SPECIFICATION OF THE BOUNDARY EXTRACTION APPROACH The exploration study allowed to further specify the characteristics of the boundary extraction approach. Among the estimations that were made in table 4.1, some have to be nuanced. In fact, it appeared that irregular densities do not necessarily pose a problem to the approach itself. However, they create a requirement to the local neighborhood scale which leads to shapes with limited width being extracted as lines rather than polygons. Furthermore, the shapes obtained will still be 'strict', thus containing the points while having a minimum area in the case of convex shapes, this cannot be said for concave shapes. Similarly to the convex hull approach an overestimation of the area can take place for these types of shapes, be it to a lower degree than the convex hull.

FURTHER SPECIFICATION OF THE VORONOI DIAGRAM APPROACH Further, it was shown that the main challenge of the Voronoi diagram approach which is the presence of cells outside the set of points can be addressed by using the roof surface geometry which was already used in the previous chapter 3. Overall, one of the main advantages of the Voronoi diagram approach is that it does not require the selection of points among the set to be labeled as 'deviation'. It performs the conversion from points to geometry at a relatively early stage and continues by merging selected geometries. Also, the nearest-neighbor interpolation approach makes it rather resistant to changes in point density. Occlusion might nevertheless be a challenge as it can result in overestimating the area of 'deviations'. If there is a high distance between two points of which one is a deviation, the Voronoi diagram approach will simply assume the 'deviation' stops at the middle which might not always correspond to reality. Estimation for the area between the edge of the bounding box (of the roof surface) and the closest point might also alter the quality of the results if that point is located relatively far. Overall, it can nevertheless be said that the Voronoi diagram has better resistance to point cloud density changes than the boundary extraction approach. In opposition to these methods, it does not lead to artifacts such as holes in the resulting shape.

THE CASE OF SMALL REGIONS The last topic of interest is the support of 'deviation' regions with only a few points. If for instance only one of the points belonging to a roof surface is labeled as 'deviation', neither the convex hull, concave hull nor the boundary extraction approach can handle it. In contrast, the Voronoi diagram can do so.

CHOICE OF FRAMEWORK The Voronoi diagram approach does thus have considerable advantages with regard to the alternative approaches that were studied. The resistance to changes in point cloud density was further specified and identified as robust, although limitations obviously exist. Furthermore, it was observed that in opposition to the boundary extraction algorithm, the Voronoi diagram approach can handle different types of shapes without requiring any setting. For these reasons, the Voronoi diagram will be implemented for the conversion of points into geometries.

4.3 IMPLEMENTATION: EXTRACTION OF THE VORONOI DIAGRAM

The implementation of the geometry extraction by usage of a Voronoi diagram was made by building upon the *scipy.spatial.Voronoi* library for python. As one can see in line 2 of the algorithm 3, this library initiates a Voronoi diagram object which contains several attributes (object.vertices, object.regions, *object.ridge_vertices*, *object.ridge_points*;for the two last ones, also see figure 4.15).

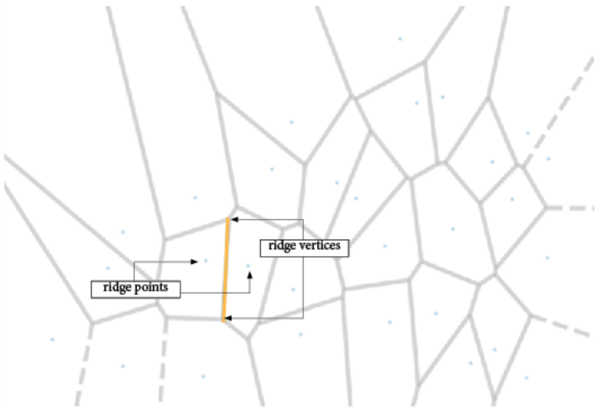


Figure 4.15: Illustration of the meaning of the terms 'ridge points' and 'ridge vertices' (as used in *scipy.spatial.Voronoi*), for a given Voronoi diagram edge.

4.3.1 Case of regular, closed cells

After building the Voronoi diagram, the extraction of the geometries can be performed starting from line 7. One might note that just before, the *centerpoint* (mean point) of the input point set is calculated. This is to allow the correct orientation of infinite edges which, if needed, is performed in lines 14 to 27. This process will further be elaborated in part 4.3.2. For closed cells, the standard process applies and involves fewer steps (for all geometric operations, *shapely* has been used):

- For each point labeled as 'deviation' the coordinates of the cell are extracted in counter-clockwise order (lines 10) and a geometric object (polygon) is built (lines 28).

- Once this has been done for all points, the extracted polygons are merged together (line 31).

- Finally, the latter shape is intersected with the roof surface edges (line 32) and the result is stored (as a WKT textstring, in a csv file, line 34).

Algorithm 4.1: POINTS TO VORONOI ($roof\,surface$, $cloud_{roof\,surface}$, $list_{regions}$,)

```
Input: The geometry extraction builds upon the output of algorithm 2:
       $roof\,surface$, a 3D geometry of the roof surface, a list of points
       $cloud_{roof\,surface}$ located inside that roof surface in 2D; and the
       respective list of deviation regions $deviation_{regions}$ previously
       identified
Output: $deviation_{polygons}$ A list of polygons each representing another
        deviation region.

Initialization;
build a 2D voronoi diagram of $cloud_{roof\,surface}$ $voronoi_{vertices} \leftarrow$ vertices of
  the Voronoi diagram;
$voronoi_{cell\,vertices} \leftarrow$ vertices of each cell, in counter clockwise order
  (including infinity vertices);
$voronoi_{ridge\,vertices} \leftarrow$ vertices of each ridge segments;
$voronoi_{ridge\,points} \leftarrow$ dual points of each ridge segment in $voronoi_{ridge\,vertices}$;
$centerpoint \leftarrow$ center point of $cloud_{roof\,surface}$;
Processing;
for $region_{points} \in deviation_{regions}$ do
    for $point_{to\,extract} \in region_{points}$ do
        $current_{cell\,vertices} \leftarrow$ find $voronoi_{cell\,vertices} \cap point_{to\,extract}$;
        if $infinity\ vertex \notin current_{cell\,vertices}$ then
            $current_{polygon} \leftarrow$ create a polygon geometry with
              $current_{cell\,vertices}$;
            append $current_{polygon}$ to $region_{cells}$;
        else
            $end\,vertices \leftarrow$ vertex before and vertex after infinity vertex in
              $current_{cell\,vertices}$;
            $far\,points \leftarrow \emptyset$;
            for $end\,vertex \in end\,vertices$ do
                find $ridge\,vertices$ and $ridge\,points$ where
                  $end\,vertex \in ridge\,vertices \in voronoi_{ridge\,vertices}$ and
                  $point_{to\,extract} \in ridge\,points \in voronoi_{ridge\,points}$ (for
                  respectively dual ridge points and vertices);
                $middle_{ridge\,points} \leftarrow \overline{ridge\,points}$;
                $\overrightarrow{normal_{line}} \leftarrow \overrightarrow{vector_{ridge\,points}}$;
                normalize $\overrightarrow{normal_{line}}$ so that $|\overrightarrow{normal_{line}}| = 1$;
                $\overrightarrow{vector_{line}} \leftarrow$ rotate $\overrightarrow{normal_{line}}$ by 90 °;
                if $\overrightarrow{vector_{line}} \bullet \overrightarrow{vector_{end\,vertex\,centerpoint}} > 0$ then
                    flip $\overrightarrow{vector_{line}}$;
                append $end\,vertex + \overrightarrow{vector_{line}} * 100$ to $far\,points$;
            end
            replace infinity vertex $\in current_{cell\,vertices}$ by $far\,points$;
            $current_{polygon} \leftarrow$ create a polygon geometry with
              $current_{cell\,vertices}$;
            append $current_{polygon}$ to $region_{cells}$;
    end
    $region_{polygon} \leftarrow$ merge all polygons $\in region_{cells}$;
    $region_{cells} \leftarrow current_{polygon} \cap region_{cells}$;
    Storage;
    append $region_{cells}$ to $deviation_{polygons}$;
end
```

4.3.2 Case of infinite edges

In the case of non-closed cells (14 to 27), the approach requires more processing steps. In fact, the framework provided by *scipy.spatial.voronoi* does only provide the information that a given edge is endless, but not what the orientation and direction of that endless edge are. Basing on the attributes of the Voronoi object, this can be computed using an approach that was found in the source code of the function *voronoi_plot_2d* of the *scipy.spatiallibrary* [3].

For this approach, it is important to realize that the non-closed cells of a Voronoi diagram are the cells of the input points which are vertices of the convex hull of the input set. In order to orient the infinite lines of these non-closed cells, one might thus use the convex hull. In fact, the infinite lines are always oriented 'away' from the convex hull, never towards the inside. In practice, this can be implemented as follows (basing on the source code of *voronoi_plot_2d*, also see figure 4.16):

- Beforehand, the *centerpoint* (mean point) of all input points is computed (last part of the initialization, line 6). This one lies by definition inside the convex hull of the input points.

- Second, for each open cell, the known endpoints of the edges/ridges are extracted (line 15, indexes before and after the '-1' in $voronoi_{cell\_vertices}$ of the respective cell).

- For each of these vertices the process illustrated in figure 4.16 is followed. The respective infinite ridges are identified using $voronoi_{ridge\_vertices}$ and $voronoi_{ridge\_points}$ (line 18. Then, the *middlepoint* between the two ridge points is computed in line 19. Using the vector connecting the ridge points (line 20), a vector of the infinite line is obtained by rotation (line 22) and, if applicable, flipping. The latter (line 24) takes place if the dot product of the rotated vector and another vector going from the *centerpoint* to the *middlepoint* is bigger than 0 (if the angle is smaller than 90 or bigger than 270 °).

- Finally, the missing vertex of the infinite ridge/edge is approximated by inserting a vertex that limits the infinite line at a safe distance from the set of points and the delimiting roof surface (e.g. 100 meters, 25). These points do then replace the '-1' index in $voronoi_{cell\_vertices}$, after what the cell does follow the same processing steps as any other cell (as described in part 4.3.1).

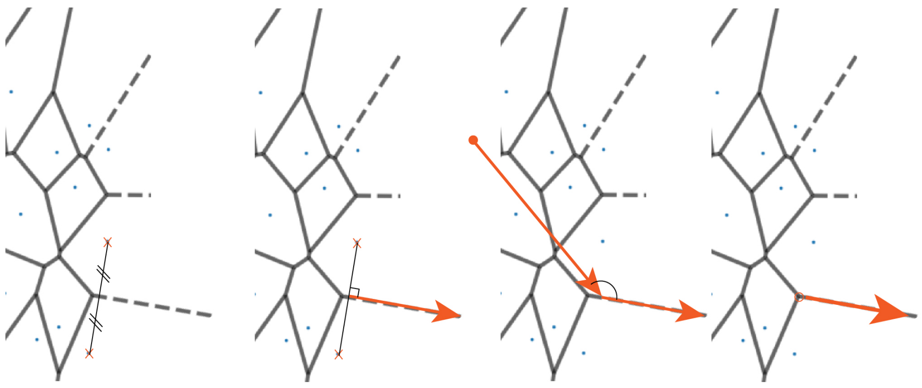


Figure 4.16: Illustration of the steps for orientation of infinite lines, from left to right: First, the middle point between the two ridge points of the infinite line is computed (line 6). Second, the line is then modelled by a vector (line 20) line , obtained by rotating the vector connecting the two ridge points (line 22). Third, the orientation of this vector is checked with regard to the vector going from centerpoint of the input (computed beforehand) to the middle of the ridge points. If needed, the direction of the infinite line is flipped (line 24). Fourth and finally, a far point is generated at a safe distance from the original set of points (line 25).

3 https://github.com/scipy/scipy/blob/master/scipy/spatial/_plotutils.py; this function uses the *matplotlib.pyplot* library too

4.3.3 Towards a semantic integration with CityGML

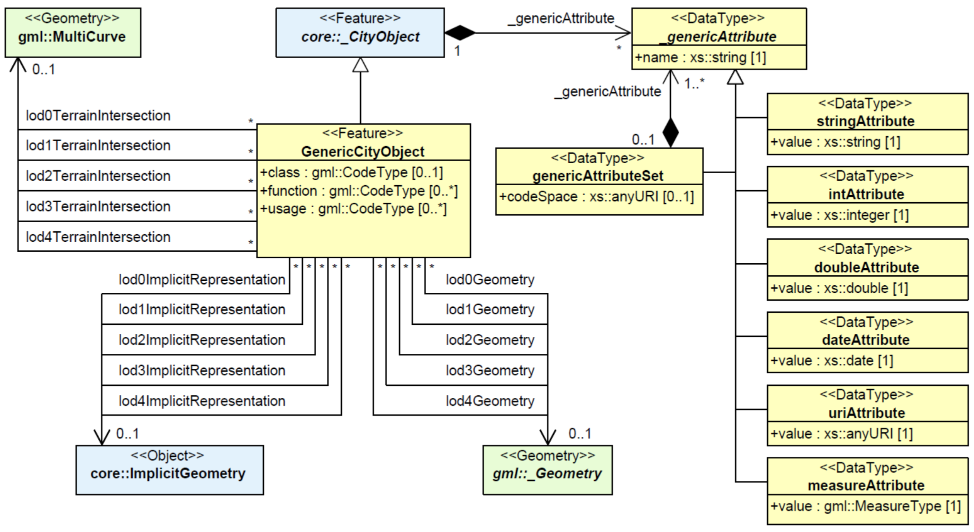
The enrichment performed within this research was limited to the requirements for the main goal: the identification of clean pixels. Therefore, the geometry and other attributes (see part 4.3.4) were stored in a .csv file.

In contrast, a stronger semantic enrichment would be the integration with the existing CityGML standard. Although not implemented, this raised some interesting questions. In fact, no framework for the addition of such 'deviations' exists yet and two possible paths were identified:

A first option is the usage of an existing feature which is a child of the building module: the *BuildingInstallation* (see annex 8.1) in the building model. Three attributes exist for this feature: class, function, and usage. The content of these attributes is customizable (more specifically, the code lists indicating the content options are). While the 'deviations' might be indicated by leaving the attributes empty (or putting 'others' as a value), filling at least the class with a specific value. In fact, beyond having an unknown function and usage, the 'deviations' are often a projection of 3D shapes onto a 2D surface. Even if the function and usage become known from tertiary sources, this will stay this way and should therefore clearly be stored in the *class* attribute.

For the storage of geometry, the standard geometry model is used and at least LOD2 is required (as defined by the CityGML geometry module). Therefore, the points defining the polygons which are now stored in the comma separated values (CSV) files would need to be projected onto the roof surface. This could easily be done by using the same vertical distance calculation as shown in part 3.4.2.

Furthermore, one might consider making holes in roof surfaces where the 'deviation' occurs to avoid overlapping geometries. While this is recommended, it is not required (as mentioned in part 3.2.2). If holes are made, it is further recommended to fill the holes using 'closure surfaces'. For the case of the specific application of this thesis (distinguishing between pixels indeed or not containing spectral deviations), making holes in the roof surfaces would not be a critical contribution. For other applications (mentioned in chapter 7) such as estimating surfaces to extrapolate material quantities or solar panel potential, it will, however, prove useful.



<<Geometry>> gml::MultiCurve
<<Feature>> core::_CityObject
_genericAttribute
<<DataType>> _genericAttribute
+name : xs::string [1]
0..1
1
*
_genericAttribute
1..*
lod0TerrainIntersection
lod1TerrainIntersection
lod2TerrainIntersection
lod3TerrainIntersection
lod4TerrainIntersection
<<Feature>> GenericCityObject
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
0..1
<<DataType>> genericAttributeSet
+codeSpace : xs::anyURI [0..1]
<<DataType>> stringAttribute
+value : xs::string [1]
<<DataType>> intAttribute
+value : xs::integer [1]
<<DataType>> doubleAttribute
+value : xs::double [1]
<<DataType>> dateAttribute
+value : xs::date [1]
<<DataType>> uriAttribute
+value : xs::anyURI [1]
<<DataType>> measureAttribute
+value : gml::MeasureType [1]
lod0ImplicitRepresentation
lod1ImplicitRepresentation
lod2ImplicitRepresentation
lod3ImplicitRepresentation
lod4ImplicitRepresentation
lod0Geometry
lod1Geometry
lod2Geometry
lod3Geometry
lod4Geometry
0..1
0..1
<<Object>> core::ImplicitGeometry
<<Geometry>> gml::_Geometry

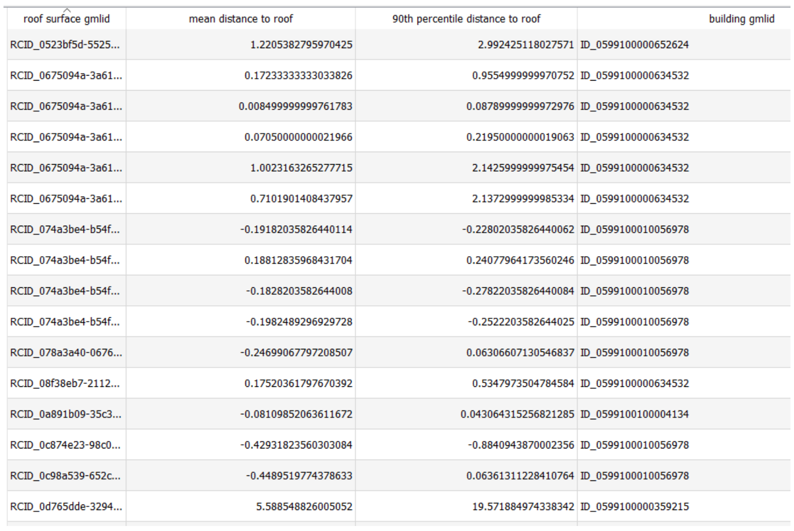
Figure 4.17: UML Diagram of the *GenericCityObject* class (Gröger et al., 2012).

The second approach relies on the fact that CityGML is an extensible standard and that so-called *GenericCityObject* can be created. Similarly to the *BuildingInstallation*, each *GenericCityObject* has the attributes class, function and usage (see figure 4.17). In the specific case of 'deviations', these ones should probably be left empty as they are unknown within this research. A difference with regard to the first option is also that the latter does allow the usage of LOD0 geometries (as defined by the CityGML geometry model). The 2D WKT geometries which are now stored in the csv file could therefore directly be used here, with no need to add a 3D coordinate.

In order to create the semantic link between the 'deviation' and the roof surface, the usage of another module, the *CityObjectGroup* is necessary. This one allows to create a group composed of several city objects. As the roof surfaces are also city objects (see 8.1), the link can directly be created with the roof surface. This is an advantage with regard to the first option as the *BuildingInstallation* is a child of the building module and can therefore only refer to the building and not the roof surface directly.

In both cases, the polygon(s) of a given 'deviation' region (in 2D or 3D) must be stored as as *gml::_Surface* object. All polygons (even if only one) should be aggregated into a *gml::MultiSurface* geometry in order to support 'deviations' composed of several polygons and to fulfill CityGML geometry requirements (see annex 8.2). Also, a few *GenericAttribute* should additionaly be created in order to store attributes such as the mean or the percentile heights (see the next part 4.3.4 for more details).

4.3.4 Additional attributes



roof surface gmlid	mean distance to roof	90th percentile distance to roof	building gmlid
RCID_0523bf5d-5525...	1.2205382795970425	2.992425118027571	ID_0599100000652624
RCID_0675094a-3a61...	0.17233333333033826	0.9554999999970752	ID_0599100000634532
RCID_0675094a-3a61...	0.008499999999761783	0.08789999999972976	ID_0599100000634532
RCID_0675094a-3a61...	0.07050000000021966	0.21950000000019063	ID_0599100000634532
RCID_0675094a-3a61...	1.0023163265277715	2.1425999999975454	ID_0599100000634532
RCID_0675094a-3a61...	0.7101901408437957	2.1372999999985334	ID_0599100000634532
RCID_074a3be4-b54f...	-0.19182035826440114	-0.22802035826440062	ID_0599100010056978
RCID_074a3be4-b54f...	0.18812835968431704	0.24077964173560246	ID_0599100010056978
RCID_074a3be4-b54f...	-0.1828203582644008	-0.27822035826440084	ID_0599100010056978
RCID_074a3be4-b54f...	-0.1982489296929728	-0.2522203582644025	ID_0599100010056978
RCID_078a3a40-0676...	-0.24699067797208507	0.06306607130546837	ID_0599100010056978
RCID_08f38eb7-2112...	0.17520361797670392	0.5347973504784584	ID_0599100000634532
RCID_0a891b09-35c3...	-0.08109852063611672	0.043064315256821285	ID_0599100100004134
RCID_0c874e23-98c0...	-0.42931823560303084	-0.8840943870002356	ID_0599100010056978
RCID_0c98a539-652c...	-0.4489519774378633	0.06361311228410764	ID_0599100010056978
RCID_0d765dde-3294...	5.588548826005052	19.571884974338342	ID_0599100000359215

Figure 4.18: Example of an attribute table, illustrating the additional attributes stored with the output geometry.

Furthermore, additional attributes were also kept through the process starting from algorithm 1 and stored together with $region_{cells}$ in line 34 in algorithm 3. These ones include for instance the building and roof surface identifiers (identical to the ones in the CityGML files provided by the city of Rotterdam). Also, height statistics were computed using the vertical distance to the roof surface (which was computed in algorithm 1 in line 11) of the points composing a 'deviation' region. These height statistics are of two kinds: on one hand the mean height and on the other hand the 90^th percentile (in the case of negative values, the 10^{th} percentile is used - 90^{th} percentile thus being defined with regard to absolute values). These height statistics are beyond the scope of this thesis but were implemented to show the potential for other applications as discussed in part 7.3. An example of the result storage can be found in figure 4.18.

4.4 CHAPTER SUMMARY AND LINK TO ASSUMPTIONS MADE

This chapter confronted itself with the problem of translating a group of points labeled as 'deviation' (resulting from chapter 3) into a 2-dimensional geometric shape. The fact that there is no formal definition for the shape of a given group of points (Edelsbrunner and Mücke, 1994) made it necessary to choose between specific definitions. This choice was made with an eye on the criteria introduced beforehand, namely the ability to cover all input cases and the automation of the approach (in opposition to human intervention).

As had been shown beforehand, the input cases can not only be subject to various shapes (connected vs. disconnected, with or without holes) and sizes (from only one up to many points), but also to uneven point densities (as mentioned by literature in part 3.1.1 and observed in part 3.2.1).

CHOICE BETWEEN METHODS Basing on literature, four options were identified: convex hull, concave hull, boundary extraction, and Voronoi diagram. The first ones were eliminated based on literature: the convex hull because its reliability is limited to convex shapes and the concave hull because it could hardly handle both changes in point density and holes.

EXPLORATION OF THE BOUNDARY EXTRACTION METHOD The boundary extraction approach was further explored using a pilot. In addition to the publications which suggested that uneven point densities might be a problem, the method also appeared to be shape-dependent (e.g. a minimum width is required). Both might have been solved by tuning settings for each different situation (thus for each deviation), but the second criterion, automation opposed it. The automatic selection of settings depending on the group of points and its density is a topic of research in itself and thus beyond the scope of this thesis.

CHOSEN METHOD: VORONOI DIAGRAM The method which was finally chosen is the Voronoi diagram method, which is able to handle changes in point density and is not influenced by the actual shape. Furthermore, the Voronoi diagram is able to handle 'deviation' regions composed of a single or a pair of points only. Nevertheless, a challenge still had to be tackled: the case of cells that are infinite or have a vertex located outside the set. As the framework of this extraction is a roof surface, the geometry of the latter was used to solve it.

Furthermore, the option of the Voronoi diagram fully fulfills the second criterion, namely automation. Within its theoretical framework, this option inherently adapts itself to the point density (using each point's nearest neighbors) and thus does not even allow settings for external tuning.

The main disadvantage found for the Voronoi diagram method is that it requires more points to compute. While the other methods require only the points labeled as 'deviation', creating the Voronoi diagram requires non-labeled points too. This led to a heavier computational load, but not to an extent endangering the feasibility of the research. As this was not a criterion for this part, it was not further considered.

IMPLEMENTATION This chapter has successfully adapted and implemented the chosen Voronoi diagram method. While a fair part was done using the existing *scipy.spatial* library, specific parts required adapting it to the needs (e.g. the case of infinite cells).

Furthermore, the implementation also performed data storage, even if in a limited way from a semantic point of view. The options for a more complete semantic storage were explored and choices to be made were formulated. Also, the storage of additional attributes such as height criteria was performed, mainly to demonstrate the potential for future research as discussed in part 7.3.

5 DATA FUSION WITH HYPERSPECTRAL IMAGERY

5.1 BACKGROUND: INSTRUMENTATION USED IN THE AIRBORNE PRISM EXPERIMENT

The hyperspectral aerial imagery used for this study was acquired within the *Airborne Prism Experiment* (APEX), a Belgian-Swiss research project with the support of the European Space Agency. The study by Priem and Canters (2016) of which the potential was discussed in part 1.2.2 used images acquired by the same sensor but at a different flying height (3650 m instead of 7000 m above sea level). Pixel size is, therefore, higher, resulting in a resolution of approximately 3.5-3.8 instead of 2m. The flight itself was performed on 17 September 2014 around noon for the *Swiss Federal Laboratory for Material Science and Technology* (EMPA, Kuhlmann et al. (2016)). The main purpose is, therefore, atmospheric research and more specifically air pollution analysis but a side-application of this data might be land cover classification. A total of three flights with a swath of 3500m each was performed, with an overlap of 600m between datasets (see figure 5.1).

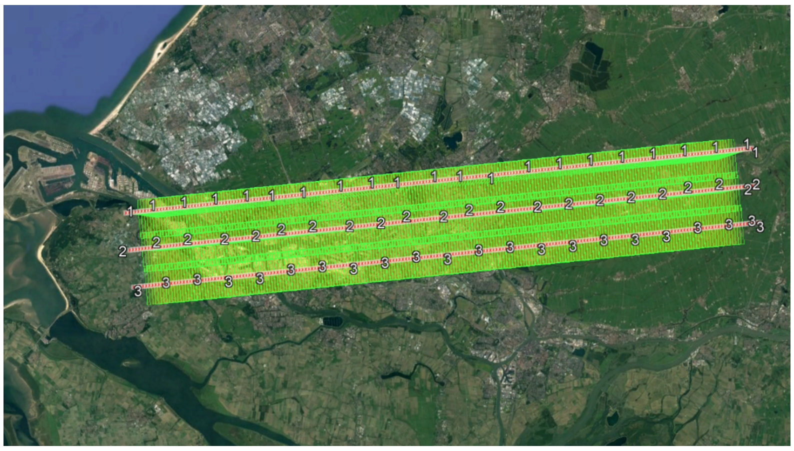


Figure 5.1: Overview of the flight lines of the hyperspectral acquisition performed by the APEX on 17 September 2014 in Rotterdam. Background imagery: (c) google earth

As the name suggests, a central part of the measuring device is a prism (for an illustration, see figure 5.2). After light entered the measuring device through a slit-shaped ground imager and a collimator, it hits the prism which splits the wavelengths into two groups. On one side, the visual and near-infrared (VNIR) radiation (380-970 nm) is directed to a Charge Coupled Device (CCD) sensor while the short-wave infrared (SWIR) radiation (940-2500 nm) is sent to a Complementary Metal-Oxide-Semiconductor (CMOS) sensor (Schaepman et al., 2015). Both CCD and CMOS are image sensor technologies that are also used in commercial photography equipment (e.g. smartphones, cameras).

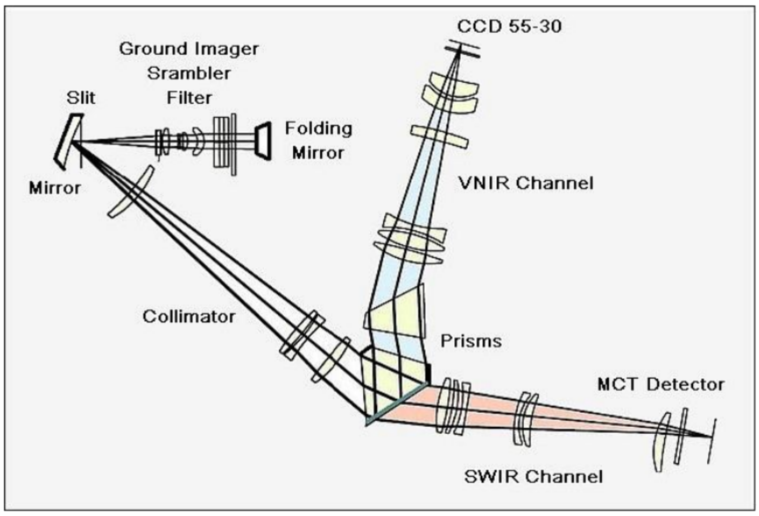


Figure 5.2: Optical design used in the apex spectrometer(ESA Earth Observation Portal Directory, n.d.).

The CCD sensor used is commercially available and produced by e2v technologies (n.d.). It has a size of 1252*1152 pixels with a pitch of 22.5 μm, and a fill-factor (the surface share that is photosensitive) of 100% (ESA Earth Observation Portal Directory, n.d.). Among the available sensors, only 1000*334 (maximum number of bands) are effectively used - the remaining ones being used for black current calibration (Schaepman et al., 2015).

In contrast, the CMOS sensor of the Saturn series used is custom made by French company *sofradir* using a HgCdTe detecting module. Also, it requires an active cooling system to work in optimal conditions (Nowicki-Bringuier and Chorier, 2009). The pitch size is 30 μm and the field has a size of 1024*256 pixels of which 1000*199 are effectively used (Schaepman et al., 2015). The fill factor for this sensor is not mentioned in publications, the only indications found is that the technology used by sofradir with the HgCdTe detecting module allows "sharp diodes with high fill factor" (Tribolet and Chorier, 2002). Moreover, more recent publications about the successor of the Saturn model indicates that a fill factor of 80% is now achieved with a pixel pitch of 15 μm (Fieque et al., 2012).

	Sensors used in the APEX experiment	
	E2V technologies CCD 55-30	*Sofradir* Saturn
Measured radiation	380-970 nm	940-2500 nm
Technology	CCD	CMOS
Number of pixels	1252*1152	1025*256
Effectively used pixels	1000*334	1000*199
Number of spectral bands	334	199
Pixel pitch size	22.5 μm	30 μm
Pixel fill factor	100%	'high'

Table 5.1: Comparison of the VNIR and SWIR sensors used in the APEX experiment

While both of the systems are rather different in terms of technology, they have a common aspect: the usage of their two-dimensional surface for a single spatial and a single spectral dimension (as explained in figure 5.3). In fact, just before radiation hits the sensor surface, a system of lenses separates it so that each wavelength hits another row of 1000 pixels. Therefore, the rows are the spectral dimension - and their number corresponds to the maximum number of bands that can be acquired (199 in SWIR + 334 in VNIR).

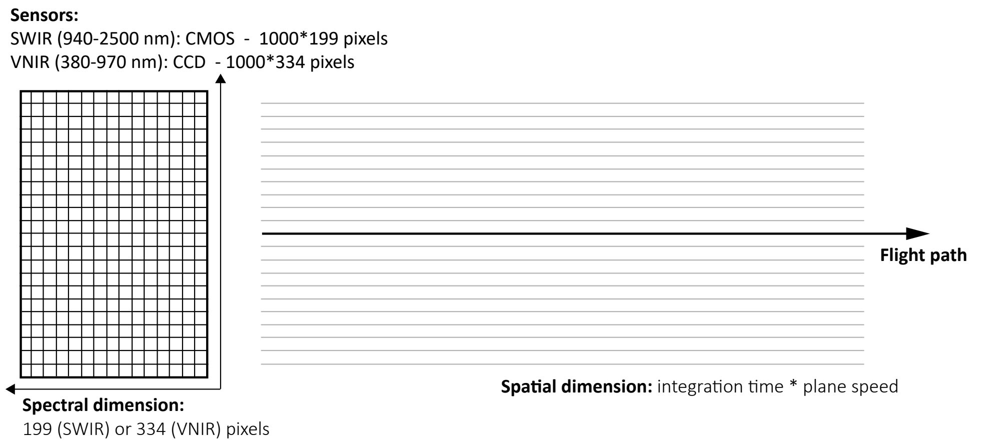

Sensors:
SWIR (940-2500 nm): CMOS - 1000*199 pixels
VNIR (380-970 nm): CCD - 1000*334 pixels
Flight path
Spatial dimension: integration time * plane speed
Spectral dimension:
199 (SWIR) or 334 (VNIR) pixels

Figure 5.3: Illustration of the different dimensionality involved during data acquisition with the APEX device.

The other dimension, the columns, the 1000 pixels of each pixel is the spatial one. Technically speaking, APEX instrumentation is acquiring spatial 'bands' with a fixed width (resulting from the combination of flying height and field of view). Spatially speaking, pixels only appear by sampling the signal temporally. The so-called 'integration time' can be set to a maximum of 34.5 ms. In combination with the displacement of the carrying airplane, this results in spatial information. With an integration time of 29 ms, almost square pixels are obtained when mounted on the propeller plane that is usually used for acquisition (Schaepman et al., 2015).

High fill factors are another important consideration during data fusion. In fact, this means that the sensor is sampling the observed surface as such and not just a part of it. If 'deviations' are present on the observed surface, they will impact the spectral values registered. This is in contrast with other, mostly active sensing methods (i.e. LiDAR) where the values represent much smaller surfaces.

The validity of this approach is a bit lower for the CMOS sensor/SWIR radiation as the fill factor is lower. Addressing this would require a detailed analysis including the exact geometry of the sensor pixels. As such an analysis is beyond the scope of this thesis and as the fill factor is "high", it will not be addressed here.

5.2 CONCEPTUAL FRAMEWORK: FROM PIXEL IMAGERY TO MESH

The hyperspectral aerial imagery that was obtained for this study has been subject to limited preprocessing beforehand (so-called level 1C as described by Schaepman et al. (2015)). Both spectral (atmospherical, as well as the keystone, smile and co-registration filtering) and spatial (georeferencing and orthorectification) corrections were performed by the APEX team using PARGE software. This pre-processing did not involve any interpolation as two products were obtained: the spatially raw but spectrally corrected data as well as the spatially corrected center points for each of the latter's pixels.

The aim of the study being to use the 'pixels' (which actually become a mesh) as the unit of validation, further considerations are necessary for data fusion.

First, the flight path of the aircraft is not totally regular (turbulence and wind induce small speed and axis variations) and the corrected center points, therefore, do not form a 100% regular raster. Thus, not all raw data pixels have the same shape and size. However, assuming that the displacement is equally distributed between any two sampling moments, a mesh can be reconstructed (as shown in figure 5.4). This is done by equally dividing the space between the spatially corrected center points (the neighborhood relations are implicitly given by the raw data).

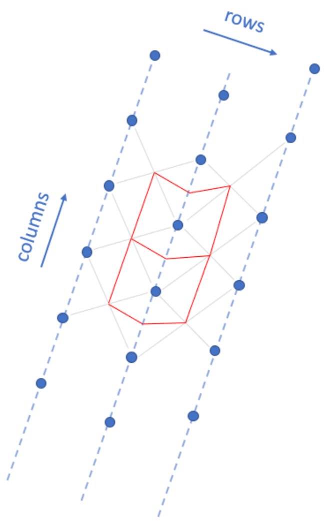


Figure 5.4: Approach allowing to generate a mesh using the orthorectified centerpoints of the 8 neighbouring pixels.

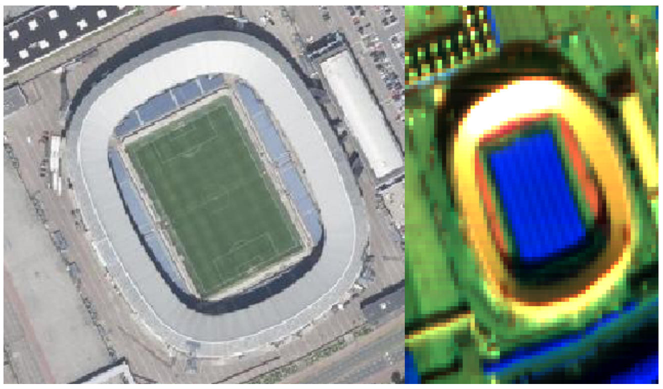


Figure 5.5: Illustration of the need for orthocorrection. On the left: nearly orthocorrect aerial image of the Feyenoord stadium (source: PDOK aerial imagery 2016). On the right: false color acapex imagery of the same stadium with the effect of projection (Red = 399-413 nm, Green = 1145-1155nm, Blue=2423-2432nm).

Second, the data obtained is orthorectified using a digital terrain model but not a digital surface model. Therefore, this is not a true 'orthographic' view (also see figure 5.5): some building facades are visible and the position of roofs is shifted (perspective phenomenon for non-oblique parts of the image). This phenomenon can be taken into account by adding correction factors to the pixel positions, using the height of the building (which can be extracted from the 3D city model). The mathematical approach is illustrated in figure 5.6 and results in the equations 5.1 and 5.2.

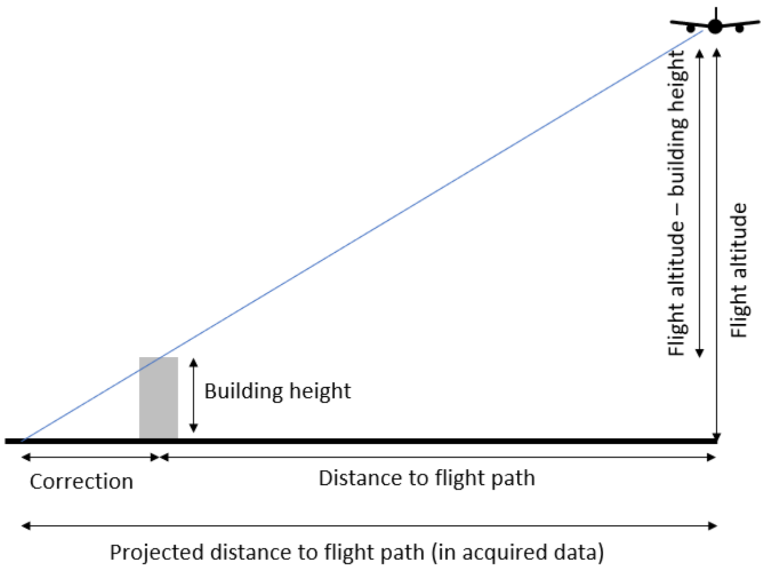

Flight altitude – building height
Flight altitude
Building height
Correction
Distance to flight path
Projected distance to flight path (in acquired data)

Figure 5.6: Trigonometric approach to the correction of perspective distortion in aerial images (production of orthophotos).

$$\frac{distance\,to\,flight\,path}{flight\,altitude - building_{height}} = \frac{distance\,to\,flight\,path + correction}{flight\,altitude} \quad (5.1)$$

$$correction = \frac{distance\,to\,flight\,path \times flight\,altitude}{altitude - building_{height}} - distance\,to\,flight\,path \quad (5.2)$$

One should note that the approach which has been implemented here is an approximation, mainly for two reasons:
- the location of the roof is simplified to its centerpoint.
- the mean height of the roof geometry is taken as reference height.

This is mainly for computational reasons: with these assumptions, a single correction factor can be applied to all candidate pixels to check if these ones intersect the given roof surface. A more precise approach would require dividing the roof surface into several samples and applying different corrections before checking the intersection with each of these samples, which is beyond the scope of this thesis. Furthermore, the estimated errors are acceptable as the calculations in the next paragraphs show. They build on the following facts:
- the field of view is 28 degrees across the track, resulting in a swath width of 3500.
- the maximum distance at which a building can be located from the flight line is thus 1750m (although in the specific study area, this maximum is lowered to 1450m due to the specific position and images overlapping 600m).

A realistic case is a building with a length of 80m perpendicularly to the flight path. With one end at the end of the swath width (where perspective distortions are biggest) and the other one 20m meters closer, and identical roof heights - the different correction factors would be: - at 1370m with a height of 10m: 1.97m
- at 1410m with a height of 10m: 2.02m
- at 1450m with a height of 10m: 2.07m
and thus the resulting positional error (difference) in such extreme cases: 5 cm.

Another factor that can lead to errors is the presence of height difference within a roof surface. For this purpose, for each roof surface present in the validation set (see part 6.1.1) the maximum difference between any point and the centroid was calculated. The resulting histogram in figure 5.7 shows that in about 59% (240/410) of the cases, the maximum difference does not exceed 25cm. In 85% of the cases, it does not exceed 2m and in 98% of the cases not 4m.

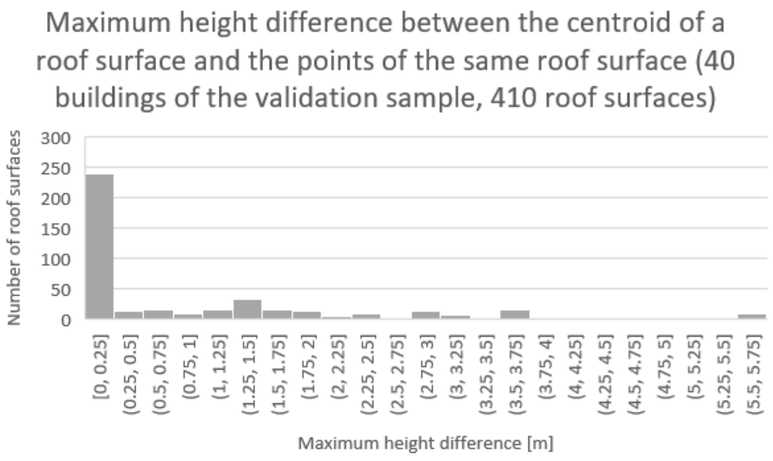


Figure 5.7: Histogram showing the distribution of the maximum difference between any point and the centroid, per roof surface.

Based on the equation shown earlier (see part 5.2) and the approach shown in figure 5.8, the impact of these maximum height differences was calculated using the following formula:

$$misplaced_{share} = \frac{error_{part} \times (ground\,sampling\,distance * 2 - error_{part})}{ground\,sampling\,distance^2} \tag{5.3}$$

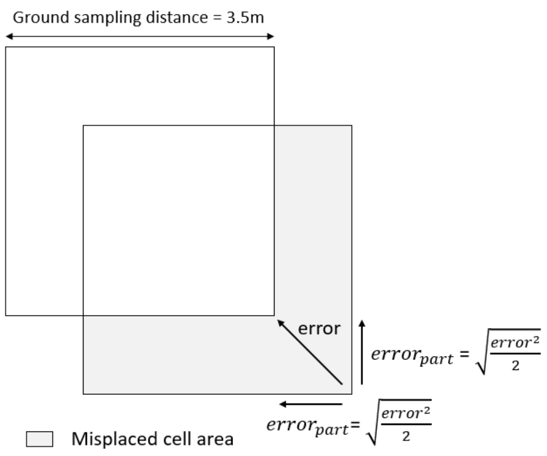

Ground sampling distance = 3.5m
error
$error_{part} = \sqrt{\frac{error^2}{2}}$
$error_{part} = \sqrt{\frac{error^2}{2}}$
Misplaced cell area

Figure 5.8: Illustration of the maximum impact of a potential inaccuracy on a cell of 3.5*3.5m.

As a distance with regard to the flight line, the maximum distance of 1450m was taken into account and as a height, 10m was taken as reference. The results can be seen in 5.2. Here, the 5cm error from the positional error was taken into account too.

Maximum errors resulting from the approximate orthocorrection			
% of cells concerned	maximum height error	resulting maximum positional error	maximum % of area affected
58%	25cm	6 + 5 cm	5.4%
25%	200cm	42 + 5 cm	23%
15%	400cm	84 + 5 cm	43.6%

Table 5.2: Overview of the maximum errors resulting from the approximate orthocorrection using centroids only (resulting from the maximum height and position differences with regard to these centroids).

For the majority of cells (58%), the share of the area that might be misplaced is not higher than 5.4%. For 25%, this share does not exceed 23%. Only for the resulting 15%, it might be critically higher. One should keep in mind that these are maximum values that will only occur in local extrema. To truly reach these values, the following conditions must be aligned:
- the entire cell is affected by the maximum height difference with regard to the centroid.
- the cell is located at about 1450m from the flight line.
- the cell is located on a building with a height of 10m above the ground.
- the cell has a size of approximately 3.53.5, while in the dataset up to 3.83.8m occur.
- the error displacement is oriented diagonally with regard to the cell.

Considering that all these factors have been taken into account to calculate the most extreme scenarios, the values obtained are acceptable, and even negligible for a fair part.

5.3 IMPLEMENTATION: ORTHOCORRECTED GENERATION OF MESH CELLS

The workflow that was implemented in order to perform the fusion of spatial imagery from the hyperspectral imagery data is shown in algorithm 4. An important aspect to note is that the input file was not provided in the same coordinate system as all other files (here the coordinate reference system with code 32632 in the EPSG (European Petroleum Survey Group) classification was used, while all others used 28992). To perform this transformation, the Geospatial Data Abstraction Library (GDAL) for Python was used.

The steps of the algorithm can roughly be described as follows:

- First, the pixel centerpoints were retrieved from the input dataset (.bsq file) using the same GDAL library for python. As the initial set of pixels covers the entire area of all flight paths, this one is reduced to the study area of interest (see lines 2).

- Then, the shortest distance to the flight line was calculated and the average height of each roof surface was retrieved by obtaining the 3D centroid (line 5).

- Basing on these two pieces of information from the previous step, the orthocorrection vector is calculated (line 8).

- The bounding box of the roof surface is obtained and corrected using the orthocorrection vector (line 9). All centerpoints which fall inside the latter bounding box are then selected and a mesh with cells defined by 6 points is defined (line 10 until 12).

- The obtained mesh cells are then orthocorrected using the orthocorrection vector (line 14).

- Finally, the extent to which the mesh cell intersects with the roof surface is calculated (line 16). If the intersection exceeds a given threshold, the area of the mesh cell which is occupied by deviations is calculated (line 17) and stored along with the mesh cell itself (line 19 and line 20).

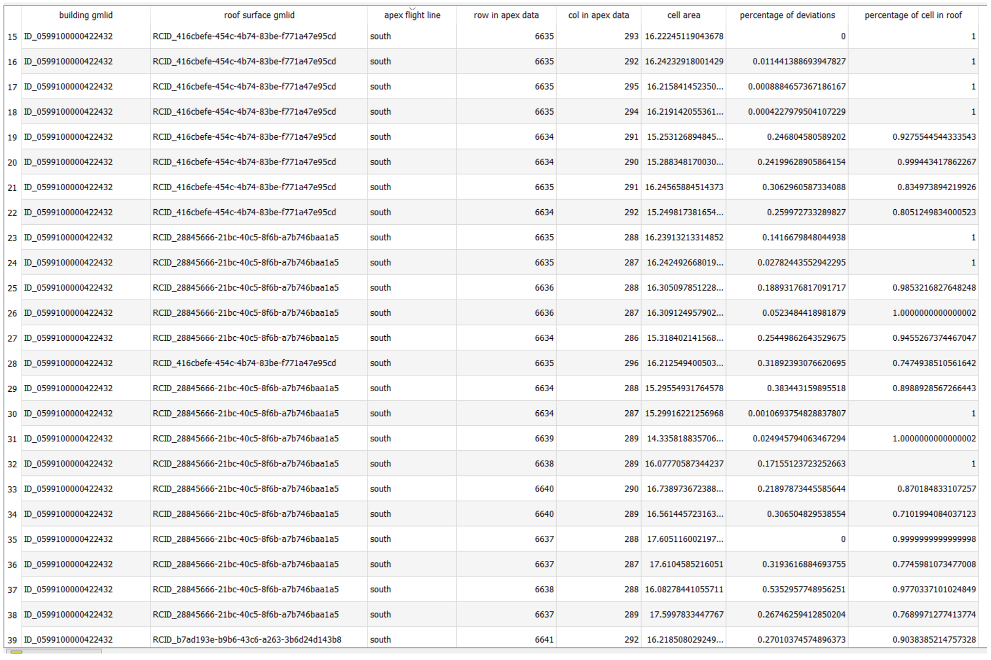
Algorithm 5.1: CELL GENERATOR ($Pixel\ centerpoints_{orthorectified}$, $flight\ line$, $model$, $area_{study}$, $threshold_{cell\ inside\ roof}$)

Input: A set of orthorectified centerpoints of the hyperspectral imagery pixels $Pixel\ centerpoints_{orthorectified}$; the flight path $flight\ line$ of the latter acquisition; a semantic 3D model of a city or neighborhood (e.g. in CityGML format) $model$; a 2D polygon $area_{study}$ of the latter city model extents; $threshold_{cell\ inside\ roof}$, a threshold for the cell inside roof criterion; a 2D

Output: $roof\ surface_{cell\ set}$, a list of cells belonging to the respective $roof_{surface}$, fulfilling the $threshold_{cell\ inside\ roof}$ criterion.

```
1  Initialization;
2  $orthorectified\ centerpoints_{study\ area} \leftarrow$
     $Pixel\ centerpoints_{orthorectified} \cap area_{study}$;
3  for $building \in model$ do
4      for $roof_{surface} \in building$ do
5          $roof\ surface_{centroid} \leftarrow$ centroid of $roof_{surface}$;
6          $distance_{flight\ line} \leftarrow |centroid - flight\ line_{closest\ point}|$;
7          Processing;
8          $\overrightarrow{orthorectification} \leftarrow$ calculated using $distance_{flight\ line}$ and
             $roof\ surface_{centroid}$ height;
9          $bounding\_box \leftarrow$ extents of $roof_{surface}(2D) + \overrightarrow{orthorectification}$;
10         $orthorectified\ centerpoints_{roof\ candidates} \leftarrow$
             $orthorectified\ centerpoints_{study\ area} \cap bounding\_box$;
11         for $centerpoint \in orthorectified\ centerpoints_{roof\ candidates}$ do
12             $mesh_{cell} \leftarrow 6 - verticemeshcellof$centerpoint$usingneighbors$;
13             for $vertex_{mesh\ cell} \in mesh_{cell}$ do
14                 $vertex_{mesh\ cell} \leftarrow vertex_{mesh\ cell} + \overrightarrow{orthorectification}$;
15             end
16             if $mesh_{cell} \cap roof_{surface}\ (2D) > threshold_{cell\ inside\ roof}$ then
17                 $deviation\ area_{mesh\ cell} \leftarrow area(mesh_{cell} \cap roof\ surface_{deviations})$;
18                 Storage;
19                 append $mesh_{cell}$ to $roof\ surface_{cell\ set}$;
20                 store $deviation\ area_{mesh\ cell}$ as an attribute of $mesh_{cell}$;
21             end
22         end
23     end
24 end
```

An example of the attribute table stored together with the mesh geometry in csv format can be found in 5.9. As one can see, the roof surface and *building ids* of the CityGML files are stored. For the sake of readability, the rather simple process of choosing between the two flight lines for the study area of the next chapter was not mentioned so far (calculation of the position of the roof centroid with regard to the line geometry which is equidistant to the flight lines). Therefore, the relevant flightline dataset is indicated (e.g.'south'), along with the row and column of the pixel. Finally, the area of the cell itself, the share of that area which is occupied by deviations and the percentage of the cell which is located inside the roof geometry are stored too.



	building gmlid	roof surface gmlid	apex flight line	row in apex data	col in apex data	cell area	percentage of deviations	percentage of cell in roof
15	ID_0599100000422432	RCID_416cbefe-454c-4b74-83be-f771a47e95cd	south	6635	293	16.22245119043678	0	1
16	ID_0599100000422432	RCID_416cbefe-454c-4b74-83be-f771a47e95cd	south	6635	292	16.24232918001429	0.011441388693947827	1
17	ID_0599100000422432	RCID_416cbefe-454c-4b74-83be-f771a47e95cd	south	6635	295	16.215841452350...	0.0008884657367186167	1
18	ID_0599100000422432	RCID_416cbefe-454c-4b74-83be-f771a47e95cd	south	6635	294	16.219142055361...	0.0004227979504107229	1
19	ID_0599100000422432	RCID_416cbefe-454c-4b74-83be-f771a47e95cd	south	6634	291	15.253126894845...	0.246804580589202	0.9275544544333543
20	ID_0599100000422432	RCID_416cbefe-454c-4b74-83be-f771a47e95cd	south	6634	290	15.288348170030...	0.24199628905864154	0.999443417862267
21	ID_0599100000422432	RCID_416cbefe-454c-4b74-83be-f771a47e95cd	south	6635	291	16.24565884514373	0.3062960587334088	0.834973894219926
22	ID_0599100000422432	RCID_416cbefe-454c-4b74-83be-f771a47e95cd	south	6634	292	15.249817381654...	0.259972733289827	0.8051249834000523
23	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6635	288	16.23913213314852	0.1416679848044938	1
24	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6635	287	16.242492668019...	0.02782443552942295	1
25	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6636	288	16.305097851228...	0.18893176817091717	0.9853216827648248
26	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6636	287	16.309124957902...	0.0523484418981879	1.0000000000000002
27	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6634	286	15.318402141568...	0.25449862643529675	0.9455267374467047
28	ID_0599100000422432	RCID_416cbefe-454c-4b74-83be-f771a47e95cd	south	6635	296	16.212549400503...	0.31892393076620695	0.7474938510561642
29	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6634	288	15.29554931764578	0.383443159895518	0.8988928567266443
30	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6634	287	15.29916221256968	0.0010693754828837807	1
31	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6639	289	14.335818835706...	0.024945794063467294	1.0000000000000002
32	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6638	289	16.07770587344237	0.17155123723252663	1
33	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6640	290	16.738973672388...	0.21897873445585644	0.870184833107257
34	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6640	289	16.561445723163...	0.306504829538554	0.7101994084037123
35	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6637	288	17.605116002197...	0	0.9999999999999998
36	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6637	287	17.6104585216051	0.3193616884693755	0.7745981073477008
37	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6638	288	16.08278441055711	0.5352957748956251	0.9770337101024849
38	ID_0599100000422432	RCID_28845666-21bc-40c5-8f6b-a7b746baa1a5	south	6637	289	17.5997833447767	0.26746259412850204	0.7689971277413774
39	ID_0599100000422432	RCID_b7ad193e-b9b6-43c6-a263-3b6d24d143b8	south	6641	292	16.218508029249...	0.27010374574896373	0.9038385214757328

Figure 5.9: Example attribute table showing the data stored along with the geometry of the mesh cells themselves.

5.4 CHAPTER SUMMARY AND LINK TO ASSUMPTIONS MADE

Additionally to the two datasets introduced in chapter 3, a third dataset was introduced here: the hyperspectral imagery. The latter was obtained from the team of the APEX in two forms. On one hand, a spatially raw imagery file and thus including projection and turbulence errors was used. On the other hand, the georeferenced and orthorectified centerpoints of each pixel were also used.

FROM PIXELS TO MESH CELLS In line with the criteria formulated in part 3, it was chosen not to create a new, resampled raster with the two files. Taking into account that the instrumentation used by APEX is a line scanner, a mesh with hexagonal cells was created (assuming that movements between lines are equally distributed). This allowed to perform a fusion of the raw data with the 'deviation' geometries produced in chapter 4 directly. By doing so, data losses and interpolation artifacts produced by raster resampling were avoided.

The practical usability of this method beyond the scientific scale of this study is still to be proven. In fact, the irregularity and the higher number of points (6 instead of 4) of the hexagonal mesh with regard to a strictly regular raster will lead to bigger data storage and heavier computations. As this is not a focus criteria for this chapter, exploring this impact is beyond the scope.

SIMPLIFIED ORTHOCORRECTION Another aspect that is out of the scope, but was tackled is the orthocorrection which aims at compensating projection errors in the data. This one was performed in a basic way with the aim to improve the specificity of the results computed in the subsequent chapter 6. The focus was therefore on an easy implementation - but the errors of the approximations made were nevertheless estimated in a scientific way and are deemed acceptable.

6 RESULTS AND VALIDATION

This chapter will explain the analysis that has been performed with the aim of validating the method. For this sake, a study area with a size of approximately 12 km^2 has been defined within the municipality of Rotterdam (for which a CityGML LOD2 model exists). This one has been chosen in order to include as many building typologies as possible: it covers high rise buildings on Kop van Zuid, big industrial hangars on the docks, apartment blocks, as well as semi-detached houses of different sizes (reaching from closed blocks to few attached bungalows). The extents of the study area can be seen in figure 6.1.

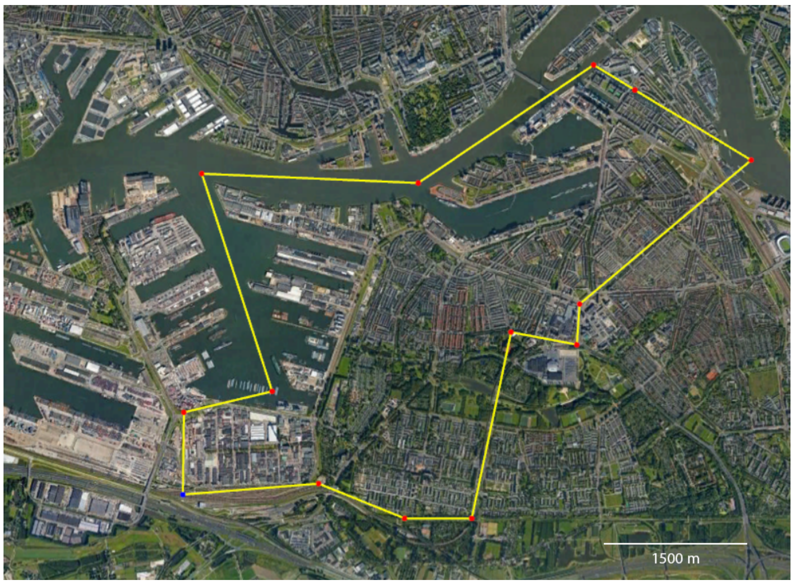


Figure 6.1: Overview of the study area chosen for the sake of validation. Background imagery: (c) google earth

This chapter will further elaborate on the approach that has been taken (e.g. selection of a subset, measuring method of the ground 'truth'). In a subsequent step, the results will be presented and discussed.

6.1 THEORETICAL FRAMEWORK

6.1.1 Validity

Ground truth

In order to ensure that the results are validated against a representative ground 'truth', linking back to the research question is necessary. For validation, mostly the third subquestion is of importance: *To which extent does such a method support the identification of clean pixels?*.

Whilst several steps have been explained in the previous chapters, the focus during validation will be on the suitability of pixels in hyperspectral imagery for material identification. As mentioned in chapter 1 in part 1.2.2, presence of several materials within a cell can be challenging for classification algorithms and should be either avoided (e.g. by excluding such cells) or addressed before classification (e.g. by applying a majority matrix or identifying gradients at transitions).

The unit of analysis is, therefore, the cell present in the hyperspectral imagery data of Rotterdam. The generation of such cells basing on pixels contained present in raw data has been discussed in chapter 5 'Data Fusion with Hyperspectral Imagery'. Basing on previous considerations, the following definitions of ground 'truth' [1] can be deduced:

A 'clean' cell is a cell filled by a single homogeneous material from a visual point of view. A 'non-clean' cell is a cell filled by several materials. A 'deviation' is a material that is different from the one covering the biggest share of the roof surface the cell lies in. The share of these 'deviations' is given by orthogonally projecting their surface on the cell and calculating the share.

An implication of this definition is that a 'clean' cell must necessarily be located inside a building's roof subsurface. In fact, cells that contain a part outside the roof surface can hardly be clean as the part lying outside might (often with a high probability) contain a different material (nevertheless, a tolerance with regard to this will be introduced in part 6.2.4). If different roof parts were made of the same material, all cells located inside the roof boundaries might be considered, but this can often not be guaranteed beforehand.

Another important aspect here is that this ground 'truth' is defined *"from a visual point of view"*. Therefore, only deviations that can be seen in the visible spectrum on aerial images will be considered in this step. This includes geometric deviations but not only as objects such as maintenance walking paths or gutters that might lie in the roof's plane will also be considered as deviations in the ground 'truth'. Furthermore, some very small objects (with a size inferior to 0.5 m^2) might be invisible in the aerial imagery due to limited resolution and are necessarily omitted. Some hard to differentiate objects (two materials which are of the same color and in the same plane, objects in the shadow of another) might also have been omitted. More details about aerial imagery data used can be found in part 6.1.2.

Validation sample

The aim of the validation is to estimate the suitability of the developed method for the selected study area in Rotterdam. For reliability purposes, a large part of the work is done manually, and the sample size was thus limited to 41 buildings [2]. An overview of the validation sample can be found in figure 6.1.
For the choice of the validation sample (and the subsample used in part 6.2.4), several criteria were established to ensure diversity:

- functional: community, offices, industrial and housing functions are all present (the two latter ones in a higher share, which reflect the distribution in the study area).

- among the housing function, low (less than 4 floors), medium (4 to 10 floors) and high (more than 10 floors) buildings are present - the biggest part being the medium buildings.

- quantity of deviations in the ground 'truth': for both the cell count and the roof surface, hard and easy cases were represented by 25 ± 6%, normal cases by 50 ± 3% (in terms of both roof surface in m^2 and number of pixels inside a roof surface).

- Building surfaces from 150 to 5600 m^2, containing from a few cells (a case with 0 cells was also included for the third validation level, using fuzziness) up to 81 cells

1 In reality, it is only a sample which is believed to be more valid than the one generated by the method. Therefore, 'truth' is used in quotation marks.

2 One might note that the definition of a building by the CityGML data of Rotterdam is not the most consistent as it follows cadastral parcels. Within this study, two groups of each about 5 semi-detached buildings have been grouped together - according to the definition used in the CityGML file, a total of 50 buildings were thus selected.

are present in the set. 81 is considered an acceptable maximum amount with regard to the total of 830 cells.

- A final criterion is geographic distribution: as shown in 6.2, the selected buildings are distributed all over the sample area. This is done in order to cover the spatial diversity, avoiding for instance selection inside a cluster built in the same period.

Also, buildings without any roof part containing a cell (e.g. small buildings or buildings with a rather high level of detail) were encountered during the selection of sample buildings. Such buildings were excluded from the validation sample as they do not have relevancy for validating the method (they are in fact out of the scope) here.

		used for validation types A, B, C and D		
name	**category**	**level**	**size [m²]**	**number of pixels 100% inside a roof surface**
Sommelsdijkstraat 45	community	normal	1400	14
Pannerdenstraat 25-21	industrial	normal	760	4
Wevershoekstraat 89	community	normal	1000	0
Slinge 290-386	housing medium	normal	1200	18
	offices	normal	1400	24
Klaverstraat 7-9	housing low	normal	1000	19
Plein 1953 95-156	housing high	hard	900	15
De la Reystraat 63-59	housing low	hard	550	7
Moerkerkestraat 128-130	housing medium	hard	190	1
Van Meelstraat 44	industrial	easy	875	26
		used for validation types A and B		
Pannerdenstraat 33-29	industrial	normal	1500	30
Mezenstraat 76-142	community	normal	2000	8
Waalhaven N.z.	industrial	normal	600	7
Sint-Janshaven 43-33	housing medium	normal	400	9
Gooilandsingel 7	community	normal	3000	42
Posthumalaan 100-122	offices	normal	1600	18
Brammertstraat 1	community	normal	2300	81
Ichthushof 200	offices	normal	4200	71
Wilhelminakade 405-677	offices	normal	5600	58
Bonaventurastraat 43-41	housing low	normal	151	2
Ooltgensplaatweg 1-23	housing medium	normal	930	9
Kortgenestraat 221	housing low	normal	200	2
Kokerstraat 16-14	housing medium	hard	3600	20
Ooltgensplaathof 13-124	housing high	hard	1000	9
Hillelaan 23-19	housing medium	hard	1630	70
Charloisse Kerksingel 35	community	hard	850	3
Laan op zuid 866-942	housing high	hard	1200	1
Pleinweg 171-177	housing medium	hard	865	9
Spuikade 13	industrial	hard	700	7
Willingestraat 4-8	offices	hard	1200	4
Geyssendorfferweg 58	industrial	hard	900	13
Dorpsweg 177-297	housing medium	hard	1200	1
Zuidhoek 59-51	housing medium	hard	530	8
Waalhaven O.z. 21	industrial	easy	1900	47
Pannerdenstraat 15	industrial	easy	1200	44
Brielselaan 121	industrial	easy	1500	29
Sluisjesdijk 129	idnustrial	easy	2400	32
Albert Plesmanweg 93-95	industrial	easy	2800	68
		used for validation types A only		
Antony Fokkerweg 15	industrial	easy	12800	615
Veerlaan 19-21	industrial	normal	8200	156

Table 6.1: Table giving an overview of the sample used for validation.

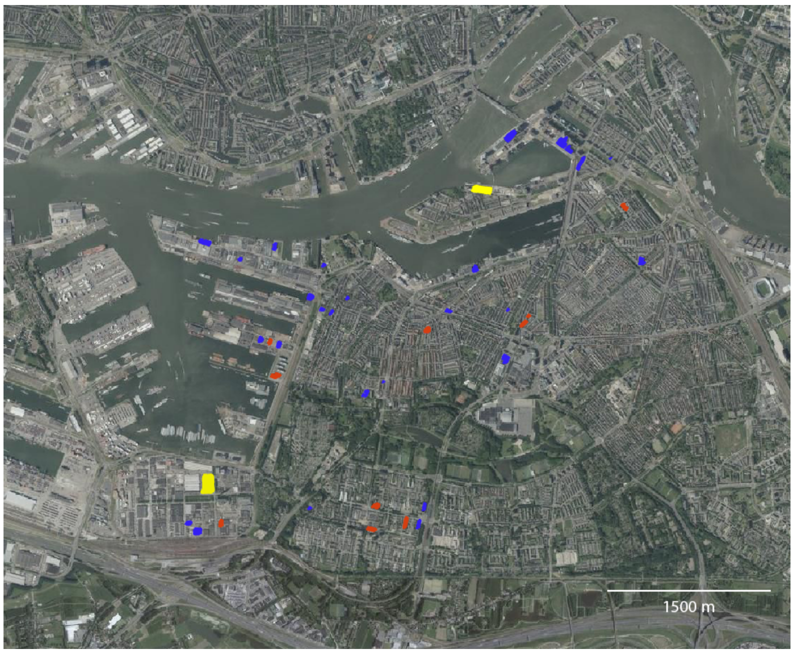


Figure 6.2: Map showing the buildings selected for the sample. The buildings in yellow were only used for the nominal validation at building level (due to having a much lager size, more details in part 6.1.2); buildings colored in red represent the subset that has been used for the validation part concerning fuzziness, for the nominal validation at cell level; both red and blue buildings were used (see part 6.1.1 for more details). Background imagery: (c) PDOK

Variables

As pointed out in the previous chapters, the method that was developed offers four variables (as mentioned in part 3.4.4):
- a threshold setting the distance tolerance (between the point cloud and the 3D city model) for the identification of seed points.
- a threshold setting the angle tolerance used during the region growing (to distinguish between planes deviating and matching with regard to the roof surface).
- the KNN setting used for the region growing
- the KNN setting used for the local normal vector calculation.

As each variable multiplies the number of versions by two or more, it was necessary to limit their number to two (due to time-wise budget constraints as stated in part 2.1). Among the variables, the two latter ones showed most conclusive results towards choosing a constant, as presented in part 3.4.4. Therefore, it was decided to use the identified constant of 10 for the KNN settings in the two last variables.

For each of the two variables left (explained in figure 6.3), two values were chosen, resulting in a total of 4 versions of the method (see figure 6.5):

- for the first variable, 20 and 40cm were chosen based on the exploration study in part 3.3.1. With 20cm, the first limits of the method were observed in part for roofs which have been modeled as flat but are in reality slightly tilted. For 40cm, such phenomena have not been observed but geometries that are lower might obviously be missed.

- for the second variable, 2 and 5 degrees were retained based on a pilot study. As can be seen in figure 6.4) 2 degrees was identified as a limit from which on noise in the point cloud starts being identified as 'deviations'. On another hand, higher thresholds have the tendency to result in smaller shapes, which might be more accurate. One can imagine that moving away from a 'deviation', the local plane becomes gradually closer to the one of the city model. If the threshold is higher, the tolerance is too and the region is expected to stop growing earlier. Basing on the result of the pilot (see figure 6.4), the second value of 5 degrees was therefore chosen.

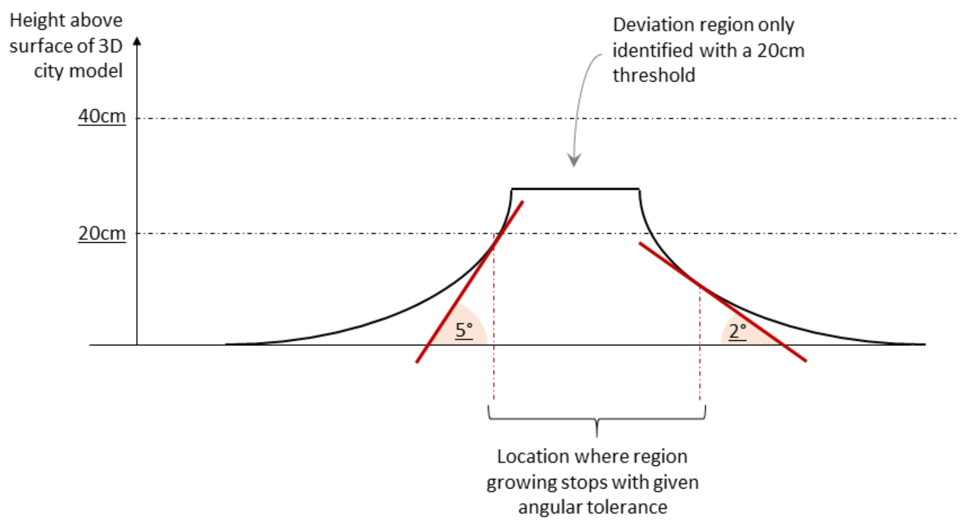

Height above surface of 3D city model
40cm
20cm
Deviation region only identified with a 20cm threshold
5°
2°
Location where region growing stops with given angular tolerance

Figure 6.3: Schema explaining the impact of settings for a 'deviation' of height 25cm. This one will only be detected with a height threshold of 20cm, not 40cm. In the case presented here, the 'deviation' will 'grow' stop growing depending on the angle tolerance used (earlier for 5 degrees than 2 degrees).

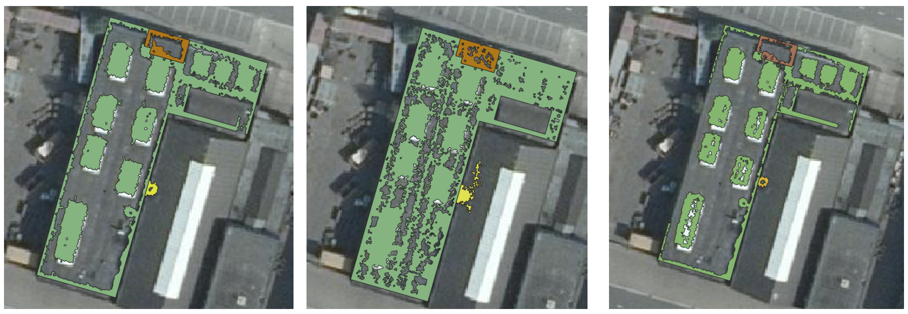


Figure 6.4: Example of the result obtained by applying the method to a roof with windows which are missing in the 3d model. Settings used here are 20cm (all three images) for the distance threshold and 2.56 (left), 1.13 (middle) and 5.73 (right) degrees for the angle threshold. Background imagery: (c) pdok

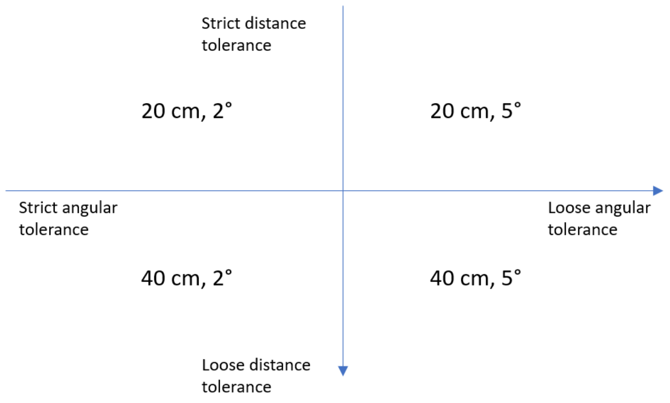


Figure 6.5: Illustration of the selected criteria resulting in four validation sets.

Types of validation

In order to cover different levels of detail, the validation was divided into three types, during each of which indicators have been generated:

- type A (see part 6.2.2): the most coarse level is the nominal validation at building level (see part 6.2.1). Here, a check whether one or several clean cells are present and whether at least one of the latter has been identified was performed. As cells are grouped by building, each building has the same contribution to the results, independently of the number of cells it contains.

- type B (see part 6.2.2): From the second level on, each cell is treated independently. For each of them, it was checked whether it is actually clean or not. By comparing the method and the ground 'truth', an estimation of the method's reliability was established.

- type C (see part 6.2.3): The third level of validation introduces fuzziness by also considering cells that are only partially enclosed by the roof surface. For this purpose, the thresholds of $\geq$90% and $\geq$70% inside the roof surface are used. As the number of cells hereby increases, this analysis was performed on a subset of 10 buildings.

- type D (see part 6.2.4): A fourth and final level of validation declines type C by switching to a rational data type. Using the same subset of 10 buildings, it looks at how clean each cell is (% of 'deviations') according to the method and compares it to the rate in the ground 'truth'.

One might note that the different levels of coarseness bring limitations with them. For the validation of type A at the building level, no distinction can be made between cases where the cells identified as clean are the same as in the ground 'truth' and cases where there is a difference. However, as the validation switches to the cell level, this kind of issue is covered in type B-D.

However, variations within cells might also be missed. In fact, as soon as there are two or more 'deviations' within the same cell, dynamics inside the cell will not be covered (for instance, for the nominal level: one 'deviation' is identified while another is missed; for the rational level: one is overestimated while the other one is under-estimated).

Performance indicators

As common for classifier validation, the results are presented in the form of error-matrices (see annexes 8.3, 8.4, 8.5, 8.6 and 8.7) and graphs (within this chapter). An additional indicator used is the estimation of cohen's kappa, the an estimate of cohen's kappa (khat) which estimates how much better than random attribution the classification is. According to Congalton and Green (2002), a khat value of more than 80% represents a strong agreement, 40 to 80% a moderate agreement and below 40% a weak agreement.

Furthermore, missing a clean pixel might be unproblematic if another one is present in the building (although it might improve reliability or allow identification of more materials). In contrast, the same cannot be said for the labeling of a non-clean pixel as clean. In fact, non-clean (or mixed) pixels might confuse classification algorithms and result in identification of non-existing materials (see part 1.2.2). Therefore, the commission error of the class 'clean' (the ratio of cells that were wrongly added to the 'clean' class) will also be used as a performance indicator (for an example, see figure 6.6).

For validation of type D, additional indicators linked to the nature of the rational data become possible. Therefore, the average error and standard deviation will also be computed.

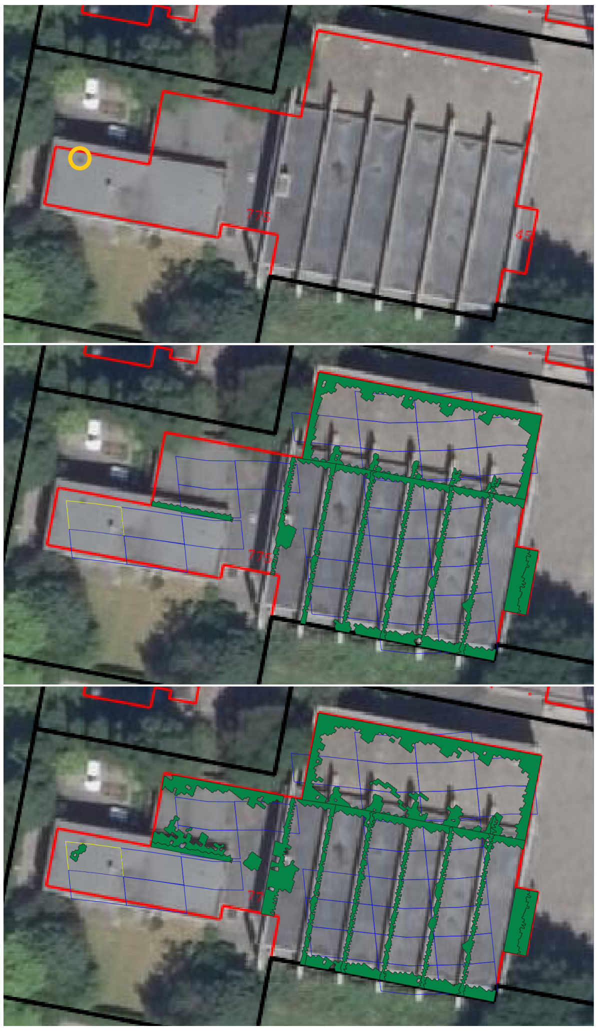


Figure 6.6: Example of a 'deviation' (identified in ground 'truth', left picture), that is not identified using the 40cm, 5 degrees setting (middle), leading to a commission error for the class 'clean' - in opposition to the 20cm, 2 degrees setting (bottom) where the cell is correctly classified as 'non-clean'. One might further note that in the same cell but at the bottom right, there is a 'deviation' which was identified by neither of the two methods. Background imagery: (c) PDOK, 2016

6.1.2 Reliability

An important consideration for reliability (the extent to which the validation can be repeated with the same results) is the distribution of building sizes and the resulting number of pixel sizes in the validation set. As has been pointed out in part 6.1.1, diversity is important during sample selection. However, diversity is not sufficient as several building types are present, and some might be over-represented. In fact, this might especially be problematic with buildings that have big roof surfaces as they contain considerably more pixels than smaller ones. For instance, it was observed that a big industrial building of 12 800 m^2 contains 615 full cells (1 cell per 20 m^2 of roof surface, the real size of the cells being +/- 16m^2) while a smaller industrial building of 700 m^2 contains only 7 full cells (1 cell per 100 m^2 of roof surface).

As buildings with big roof surfaces contain over-proportionally more cells, it is not sufficient to look at the buildings' footprint size. Instead, the actual number of cells should be taken into account too. Another implication is that the size of the biggest building of the set is limited by the size of the set itself. For the validation performed here, about 41 buildings have been selected, resulting in 917 cells.

Adding too big buildings (e.g. with 615 cells), would introduce a considerable bias and has therefore been avoided (yellow buildings in figure 6.2, conclusions for big buildings are nevertheless drawn in part 6.4).

Furthermore, some practical considerations have been taken for evaluating the presence and the size of deviations:

- Several aerial imagery datasets were used. The first one is composed of the yearly acquired and open aerial images of the Dutch government (2016, 2017 and 2018 [3]). These ones have a resolution of 25cm and different flight lines. An advantage of the latter is that nearly every building is sufficiently close to one of the three flight lines. This allows proceeding without orthocorrection.

- A second aerial imagery dataset that was used is the imagery provided by google earth acquired on 10/1/15 [4]. This data was used to support the first ones as it offers better contrast, often allowing better recognition of details.

- For the few cases where all imagery showed too big displacements of the roofs, correct roof reference positions were provided by a cadastral map [5] showing parcel and elevation borders.

- in the case of edges, the thickness was first estimated (see figure 6.7). This was done by measuring directly on imagery and in some cases in the AHN point cloud too. In that way, only one dimension, the length of the edge still has to be measured in order to determine the surface of the edge.

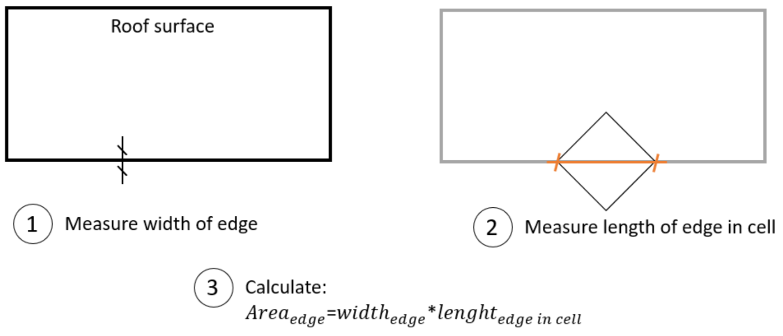

Roof surface
1 Measure width of edge
2 Measure length of edge in cell
3 Calculate:
$Area_{edge}=width_{edge}*lenght_{edge\ in\ cell}$

Figure 6.7: Schema showing the approach used in the case of roof edges. Beforehand, the constant width was measured so that in the subsequent steps only one dimension was left to measure.

6.2 RESULTS

6.2.1 Nominal validation at building level (type A)

The first validation was conducted by checking at the building level, whether at least one clean pixel is present and whether at least one of these clean pixels has been identified. For this sake, all cells belonging to one building (independently of the subsurface) were grouped together. This results in each building having the same weight in the final result, independently of the size or number of cells belonging to it. The error matrices showing the full results can be found in annex 8.3.

As the result in figure 6.8 shows, the khat is higher for the versions using a distance threshold of 40cm (e.g. a khat of 66 vs. 47 % for 5 degrees and 39 vs. 26% for 2 degrees). Furthermore, a better performance can be observed for 5 than for 2 degrees (a khat of respectively 66 vs. 39 % and 47 vs. 26 % for 40 and 20cm). The

3 obtained from https://www.pdok.nl/introductie/-/article/luchtfoto-pdok
4 As google earth does not provide any details for aerial images, the only information available about the data is the acquisition date
5 obtained from *https : //www.pdok.nl/introductie/ − /article/basisregistratie − kadaster − brk−*

third indicator, the share of buildings that were wrongly predicted as having at least one clean pixel does not show variations as it is stable around 10% (see figure 6.9).

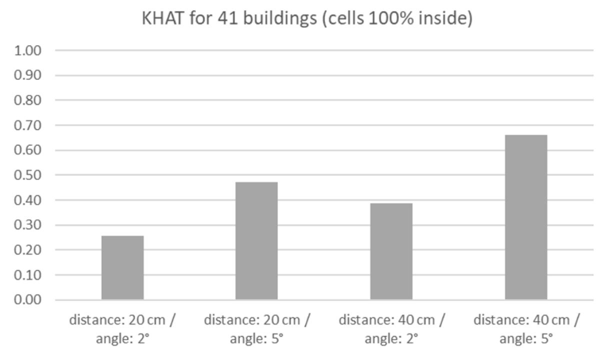


Figure 6.8: Bar chart showing the khat results of the nominal validation at building level (basing on data in annex 8.3).

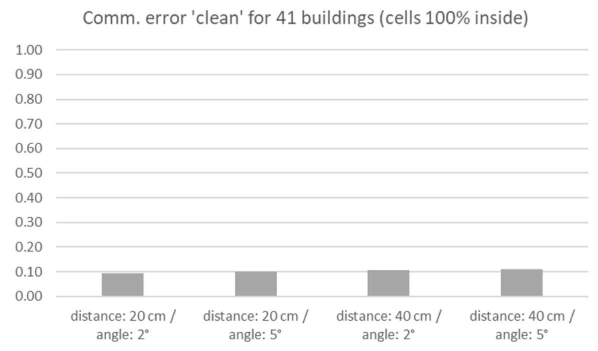


Figure 6.9: Bar chart showing the commission errors for the class 'clean' of the nominal validation at building level (basing on data in annex 8.3).

One might, however, note the shortcomings of such a coarse validation. In fact, it does not provide any information on the extent to which the identified pixels were indeed the clean ones. In order to get a better idea of the actual performance of the method, an analysis at cell rather than building level is necessary.

6.2.2 Nominal validation at cell level (type B)

A second validation was performed directly at cell level: for each cell, the prediction of the algorithm was checked with the ground 'truth'. If any 'deviation' within the cell was identified (by the algorithm or the manual observation for the ground 'truth'), the cell was labeled as being non-clean. The error matrices showing the full results can be found in annex 8.4.

As can be observed from the validation performed on a total of 830 cells, the khat (see figure 6.10) suggests that the method performs better with a threshold of 40 rather than 20cm, as the values are about 10% higher (respectively 70 vs. 58 % and 67 vs. 57 % for 2 and 5 degrees).

The observation of better performance for the 5 than for the 2-degree versions cannot be confirmed. Rather, data refutes it here (a khat of respectively 68 vs. 70 % and 57 vs. 59 % for 40 and 20cm).

Another important difference here is the commission error for the class 'clean' which considerably differs between the variable values (see figure 6.11). The lowest value is obtained for thresholds of 20cm and 2 degrees, with only 10% of cells identified as clean being non-clean in the ground 'truth'. The commission error for the class 'clean' is about 7% higher when rising the threshold to 40cm and worst for the version also using a 5-degree threshold.

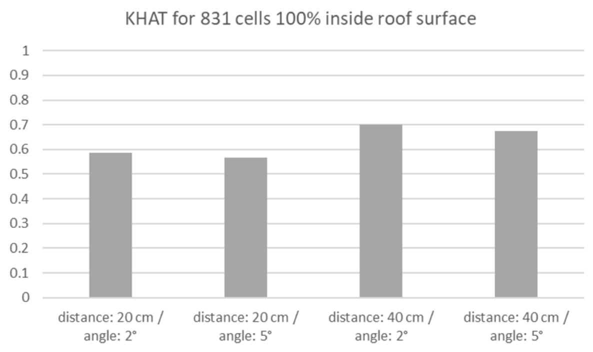


Figure 6.10: Bar chart showing the khat results of the nominal validation at cell level (basing on data in annex 8.4).

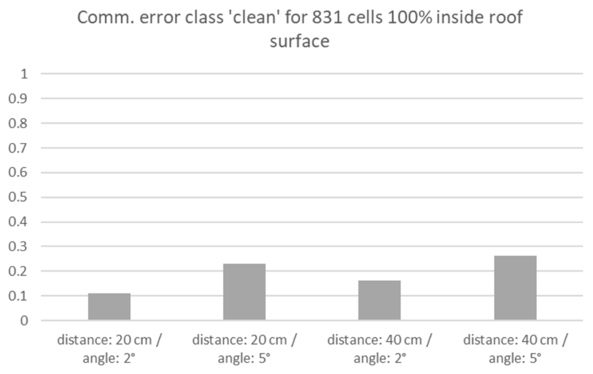


Figure 6.11: Bar chart showing the commission errors for the class 'clean' of the nominal validation at cell level (basing on data in annex 8.4).

6.2.3 Nominal validation: considering cells lying only partially inside a roof surface (type C)

This third validation introduces fuzziness by relaxing the cleanliness constraint used so far. The idea behind this is that classification algorithms with a limited tolerance with regard to 'deviations' might be used or, if needed, developed (for an example of research in that direction, see Guo et al. (2009)). Therefore, two assumptions were made - the first one is that algorithms might accept cells that are up to 10% outside the roof surface - and as a second assumption, the tolerance is raised to 30%.

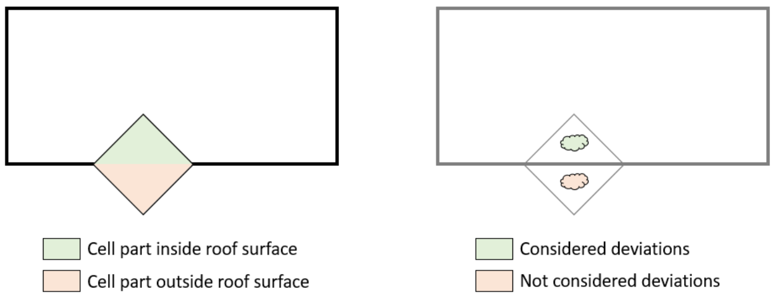

🟩 Cell part inside roof surface
🟧 Cell part outside roof surface
🟩 Considered deviations
🟧 Not considered deviations

Figure 6.12: Schema showing the case of cells that are not entirely inside a roof cell. In that case, only the 'deviations' in the part inside the roof cell are taken into account.

An important point for this part of the analysis is that only the cell's part which is inside the roof surface is considered for the method performance (see figure 6.12). The remaining up to 30% are not considered as they are unreliable: a similar material as inside the roof surface cannot be guaranteed, even if no deviations were identified in this remaining part. A pragmatic approach would be to de-facto label these up to 30% as 'non-clean'. However, this would induce bias as an additional factor, which is unrelated to the method to validate would be introduced. In order to avoid this bias, the parts outside the roof surface are simply omitted: the only aspect impacting the cleanliness (thus the results of the method) are the deviations in the part of the cell inside the roof surface.

As the typical study object of type C validation includes more cells than type A and B, it was limited to a subset of 10 buildings, which were also present in the previous validation set. One might note the growth of the subset's cells from 128 to 328 as the cell in roof surface constraint gets relaxed to 70% (see figure 6.2). Furthermore, type D will use the same subset but switch from nominal to rational data. Measuring the exact cleanliness share is more work-intensive than only identifying their position, giving another motivation to limit the extent of this approach.

detailed list				Number of pixels x% inside a roof surface	
name	category	level	size [m^2]	100%	>=70%
Sommelsdijkstraat 45	community	normal	1400	14	35
Pannerdenstraat 25-21	industrial	normal	760	4	15
Wevershoekstraat 89	community	normal	1000	0	38
Slinge 290-386	housing medium	normal	1200	18	45
Waalhaven O.z. 85	offices	normal	1400	24	60
Klaverstraat 7-9	housing low	normal	1000	19	48
Plein 1953 95-156	housing high	hard	900	15	35
De la Reystraat 63-59	housing low	hard	550	7	19
Moerkerkestraat 128-130	housing medium	hard	190	1	6
Van Meelstraat 44	industrial	easy	875	26	30

Table 6.2: Overview of the subsample used for the validation of type C and D (cells lying partially inside a roof and fuzziness).

The most interesting results are the evolution of the khat and commission error for the class 'clean'. For ease of understanding, separate bar charts have been plotted (figures 6.13 and 6.14). The full results of the type C validation performed for cells that are 100%, [90-100[% and [70-90[% inside the roof surface (respectively 128, 106 and 93 cells) can be found in annex 8.5.

One can observe that for the khat (figure 6.13), the differences between settings are smaller for the '[90-100[%' and '[70-90[%' cell groups than for the '100%' group. In fact, when moving to the '[90-100[%' and '[70-90[%', a convergence towards values between 0.4 and 0.6 can be observed.

A plausible explanation for this is the higher share of non-clean pixels among the cells that are partially outside a roof surface. An argument supporting this is that the variant with the highest commission errors for the class 'clean' observed in type B (40cm and 5 degrees), is also the one where the loss of overall accuracy and khat performance is biggest. Another argument in line with this is that the version with the lowest commission error for the class 'clean' observed in type B (20cm and 2 degrees) is showing the strongest increases in khat (figure 6.13).

Another interesting observation is the commission error for the class 'clean' (figure 6.14). The values for the '100%' cell group are similar for all settings and located between 11 and 16%. For the '[90-100[%' and '[70-90[%' cell groups higher values can be observed for all settings, located between 0.45 and 0.6. The only exception is the one obtained with the setting 20cm and 2 degrees which are lower, located between 0.3 and 0.4.

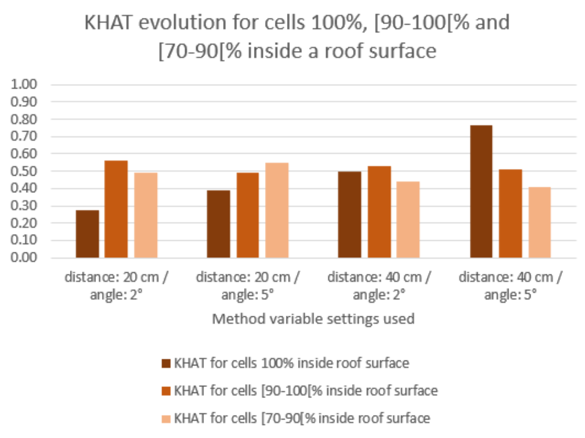


Figure 6.13: KHAT evolution for cells 100%, [90-100[% and [70-90[% inside a roof surface. For all cases with a setting of 20cm, the khat tends to increase when loosening the cell in roof surface criterion. For the 40cm, 2 degrees settings, the trend is rather stable. In contrast, the 40cm, 5 degrees settings result in a decreasing trend.

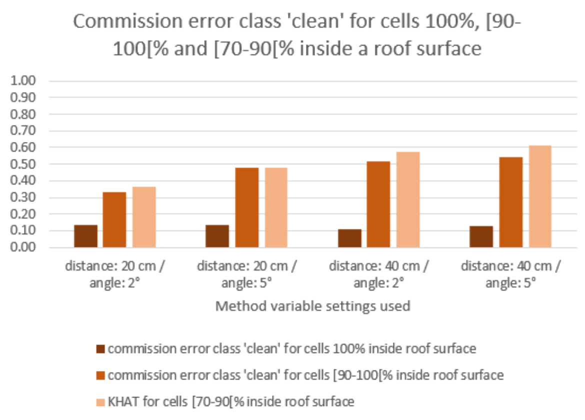


Figure 6.14: Commission error for the class 'clean' evolution for cells 100%, [90-100[% and [70-90[% inside a roof surface. With all settings, an increasing trend can be observed. The increase is however less pronounced for the 20cm and 2 degrees setting.

6.2.4 Rational validation: extending type C by introducing fuzziness (type D)

For the same set of 331 cells which are 70% or more inside a roof surface, the following indicators concerning the cleanliness estimation have been calculated:

distance: 20 cm / angle: 2°		distance: 20 cm / angle: 5°	
average difference [m²]	-0.08	average difference [m²]	0.01
standard deviation [m²]	0.13	standard deviation [m²]	0.30

distance: 40 cm / angle: 2°		distance: 40 cm / angle: 5°	
average difference [m²]	-0.06	average difference [m²]	0.01
standard deviation [m²]	0.11	standard deviation [m²]	0.08

Table 6.3: Overall performance statistic concerning the estimation of 'deviation' shares.

The numbers that are shown in figure 6.3 suggest that there is a trend to overestimate the size of 'deviations'. In fact, the average difference is negative in 3 out of 4 cases, the smallest absolute being found with the loosest settings, namely 40cm and 5 degrees.

For the standard deviation too, the 40cm and 5 degrees setting performs best, reaching a standard deviation of 8%. The other settings deliver less good results - which are reaching up to 16% for the 20cm versions. However, the standard deviations being relatively big with regard to the differences between the averages found, no solid conclusions can be drawn from this analysis. Moreover, the reliability of the ground 'truth' also decreases by switching to rational data - estimating it would ideally require field measurements at the location.

In order to estimate the usability of the method to discriminate between different levels of cleanliness, error matrices with additional, rational categories have been calculated. These ones can be found in appendix 8.6. The overall accuracy and khat are highest with a setting of 40cm and 5 degrees, reaching respectively 62% and 47%. Similarly to the previous validations,the 20cm, 2-degrees version (see annex 8.6) performs worst from the overall accuracy and khat point of view.

For the matter of comparison, the nominal validation performed at the beginning of this part was aggregated to include all cells that are at least 70% inside a roof surface in one matrix which can be found in appendix 8.7.

Comparing the results in annex 8.6 with the ones in annex 8.7, it becomes clear that the classification into rational categories requires higher performance than into nominal categories.

In fact, as can be seen in figures 6.15 and 6.16 for all four versions of the method, a decrease in both overall accuracy and khat performance can be observed. This shows that while the method can to some extent differentiate between clean and non-clean cells, the extent to which it can do so is lower.

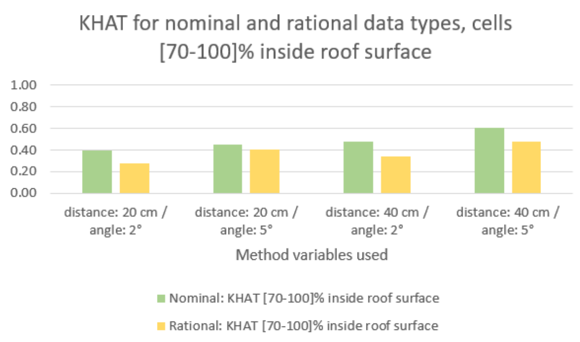


Figure 6.15: KHAT for nominal vs. rational data types, cells [70-100]% inside roof surface (level D).

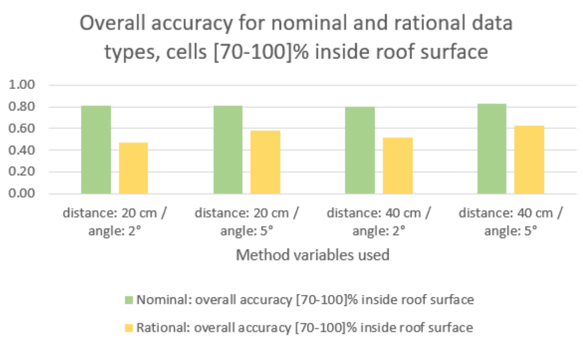


Figure 6.16: Overall accuracy nominal vs. rational data types, cells [70-100]% inside roof surface (level D).

6.3 ANALYSIS

Overview of the relevant observations

The first validation, type A (see part 6.2.1), has shown that the overall accuracy (and khat) statistics obtained with the 40cm variants are superior to the ones obtained with 20cm. Furthermore, versions using a setting of 5 degrees perform better than the ones using 2 degrees on the khat statistic.
The difference between 40cm and 20cm variants is confirmed by type B (see part 6.2.2), although the magnitude of the differences is smaller. However, the difference observed for the khat between 2 and 5-degree variants is rejected. A new finding introduced by type B is that commission errors for the class 'clean' are 5 to 15% lower for the strictest version using settings of 20cm and 2 degrees.

Type C (see part 6.2.3) was conducted on a subset of the previously used buildings. The good performance of the 20cm and 2 degrees with regard to commission errors for the class 'clean' can also be found in that subset. Therefore, this observation is confirmed for cells that are only partially inside a roof surface and often include an edge.

Type C has further shown that the overall accuracy and khat differences between 20 and 40cm versions, as observed in types A and B are reduced when including cells that are only partially inside a roof surface. While the performance of the 20cm versions tends to increase, the performance of the 40cm versions tends to stay stable or decrease. It has been shown that a plausible hypothesis to explain this is the higher share of non-clean pixels among the cells that are partially outside a roof surface.

Validation type D (see part 6.2.4) focused on rational data by looking at the degrees of cleanliness for all cells that are at least 70% inside a roof surface (of the subset used for type C too). From an overall statistical point of view, the 40cm and 5 degrees version seems to perform best as the absolute average difference is lowest and the standard deviation observed is 8%. However, it is impossible to draw reliable conclusions as the standard deviations reach up to 30%, with all average differences locate between 0 and -10%.

The second part of type D validation was dedicated to the performance of the method for differentiating between different levels of cleanliness. Here, the observation of types A and B, that the overall accuracy is higher for 40cm than for 20cm settings is confirmed (again, the KHAT confirms this too). Overall, the observation that performance is lower than for previous validations (which were still using nominal data types) was made.

Recommendations basing on the validations

Finally, it can be said that two observations were consistent through the different types of validation performed:

- the better overall performance (for the overall accuracy and khat) of the 40cm settings, compared to the 20cm settings.

- the most strict settings (20cm and 2 degrees, lowest tolerances) deliver the lowest commission errors for the class 'clean' (thus the lowest share of wrongly identified 'non-clean' cells).

Also, the methods perform better for differentiating between 'clean' and 'non-clean' cells (nominal approach) than for differentiating between degrees of cleanliness (rational approach).

The 40cm setting is the most loose one and delivers better overall results, while the strictest settings (20cm and 2 degrees) result in the lowest commission error for the class 'clean'. Depending on the application, two approaches shown in table 6.4 can be suggested:

	approach 1	approach 2
aim	identify material presence	quantify materials
strategy	avoid wrongly identifying non-clean pixels	identify as many clean pixels as possible
recommendation	use a strict setting (e.g. 20cm, 2 degrees)	use a loose setting (e.g. 40cm)
limitation	some buildings (e.g. small ones) might be missed	some low-quantified materials might be absent

Table 6.4: Table giving an overview of the two different approaches which depend on the final goal.

Approach 1 - identification of material presence. If the aim is simply to identify where given building material is present, a single cell of a given roof might be sufficient. One might, for instance, want to discover the roofs with a recyclable material and needs to perform on-site checks anyways (e.g. to check how the material was attached). In that case, it might be more wasted effort to contact a building owner, arrange a meeting and finally discover the material is not present than to miss a share of the buildings that have the material. In such a situation (which shows similarities to a 'gold-digger' approach), using the most strict settings (20cm and 2 degrees) will be beneficial. One should note that this should also be related to the building size. In fact, the buildings of interest should still deliver at least one clean cell with this approach (possibly, cells lying partially outside the roof might be considered too - then again, the 20cm 2 degrees setting delivers the lowest commission error for these too).

Approach 2 - a contrasting approach is the quantification of materials. In contrast to the first one, the goal is to get a good estimation of the quantity (for instance inm^2 of a given material for a given/building or area). In such a situation, it is necessary to differentiate as well as possible between 'clean' and 'non-clean' pixels. While including too few 'clean' pixels would lead to an under-estimation, including too many 'non-clean' pixels would lead to wrong results too (as the classifier might confuse the mix of materials with other materials). Therefore, the overall accuracy and the khat are the best indicators in this situation - basing on the results a setting of 40cm and either 2 or 5 degrees (validations performed here do not allow to make a statement on which angle tolerance to use). Just as for the first approach, this recommendation is valid with or without considering cells lying partially outside the roof surfaces.

6.4 FURTHER QUALITATIVE OBSERVATIONS

During the validation, some additional observations were made next to the core presented in the preceding sections. These ones are of a rather qualitative nature and will shortly be addressed in this section.

Limits of this validation with regard to big buildings

This validation was conducted with the aim to give a reliable estimation of the performance of the algorithm on a representative set of the study area (with the constraints elaborated in part 6.1.1 and 6.1.2). The fact that this subset cannot simply be generalized should be kept in mind when drawing conclusions basing on the results. In fact, two types of buildings were not covered by the validation:

The first one is buildings that are relatively small or, at least have relatively small roof surfaces, which thus do not contain a single cell entirely. Such buildings can be identified without the method proposed in this thesis and were therefore not of interest for the validation. One such building was nevertheless included as it is deemed to increase representativeness for the type C and D validations where partially included cells were studied too.

This suggests that the results obtained in this part are also dependent on the cell size/ground resolution of the hyperspectral imagery. If a smaller cell size was available (such as used in other studies, e.g. in Brussels (Priem and Canters, 2016), the resolution was about 2 by 2m), the set of buildings, as well as the ratio between the 'deviation' size and the cell size would change. It can, therefore, be expected that the validation results would change too (as the ratio clean/non-clean cells of the ground 'truth' would probably be different).

The second type of omitted buildings is rather big buildings with few (and thus big) roof surfaces. Including them beyond type A validation would have led to an not-acceptable share of the cells belonging to a single building (and thus jeopardizing the reliability). The results for such a big building, an industrial hall located on the 'Antony Fokkerweg 15' were nevertheless computed and can be found in figure 6.5.

Antony Fokkerweg 15, 20cm, 2 degrees - cells 100% inside				
		truth		
		clean	not clean	**total**
	clean	578	9	587
prediction	not clean	21	7	28
	total	599	16	615
	overall acc.	**0.95**	**comm. errors 'clean'**	**0.02**
	khat	**0.29**		
Antony Fokkerweg 15, 20cm, 5 degrees - cells 100% inside				
		truth		
		clean	not clean	**total**
	clean	585	9	594
prediction	not clean	14	7	21
	total	599	16	615
	overall acc.	**0.96**	**comm. errors 'clean'**	**0.02**
	khat	**0.36**		
Antony Fokkerweg 15, 40cm, 2 degrees - cells 100% inside				
		truth		
		clean	not clean	**total**
	clean	596	9	605
prediction	not clean	3	7	10
	total	599	16	615
	overall acc.	**0.98**	**comm. errors 'clean'**	**0.01**
	khat	**0.53**		
Antony Fokkerweg 15, 40cm, 5 degrees - cells 100% inside				
		truth		
		clean	not clean	**total**
	clean	599	7	606
prediction	not clean	0	9	9
	total	599	16	615
	overall acc.	**0.99**	**comm. errors 'clean'**	**0.01**
	khat	**0.71**		

Table 6.5: Example results for a big building (Antony Fokkerweg 15) of 12800 m^2 enclosing 615 cells.

It is quite striking that the overall accuracy and commission error scores are better than those encountered in the validations. In fact, on a big building that has only very few 'deviations', the algorithm performs rather good.

This clearly illustrates that the method developed and validated is more suited for some type of buildings than for others. Therefore, the validation performed in this chapter should be re-used with care and only after checking the extent to which the sample used matches the objective with which the method is used.

Importance of input geometry quality

In some cases of the validation, 'deviations' were detected at unexpected locations. After a closer look, it appeared that such cases were often related to insufficient accuracy in the 3D city model (observed in three cases in the sample). A good example can be found in figure 6.17 where a roof has erroneously been modeled as fully flat. This shows that the method requires an accurate and precise input geometry to perform correctly. If there is a mismatch, the spots with insufficient quality will be identified instead of the ones with truly diverging geometries. In such a case, the method would check the quality of the input dataset, rather than the presence of spectral variations.

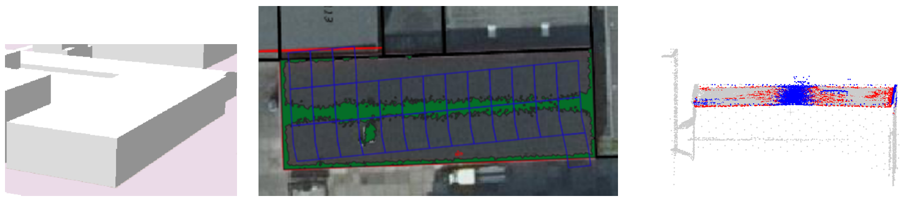


Figure 6.17: Example of a building (Van Meelstraat 44) that has erronously been modelled with a fully flat roof in the 3D city model of Rotterdam (image on the left). 'Deviations' identified in the middle of the roof (blue in the point cloud in the middle, green on the aerial image on the right), do not represent a material 'deviation' but simply the peak of the slightly gabled roof surface (visualisation in the middle and right obtained for 20cm and 5 degrees settings).

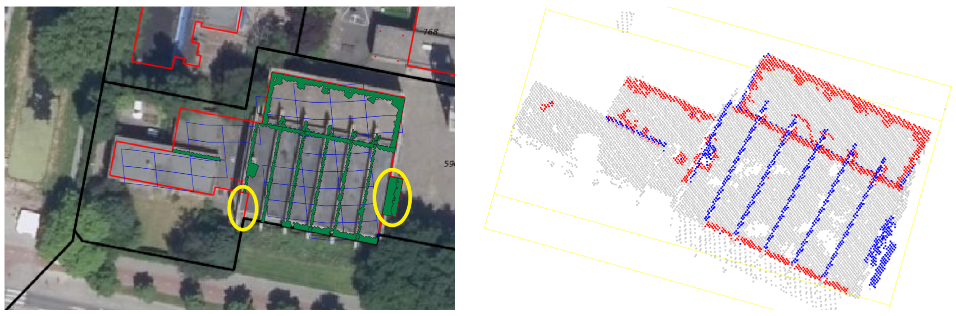


Figure 6.18: Example of a building with wrong cadastral footprints (Sommelsdijkstraat 45), leading to a wrong 3D city model biasing the results of the method. On the left, one can see the footprints represented by the red lines. The yellow circle on the left indicates a part of the building that was missed. The 'deviation' in green (a beam) suddenly stops where the building stops while it clearly continues on the aerial image. The yellow circle on the right shows a 'deviation' of the same surface that is split in two. The reason for this can be found in the point cloud on the right. The outer right cluster of blue points (seeds identified due to their high distance from the roof surface above the entrance) is split into two parts: one lies on the roof of the building while the other lies on the roof of the entrance. This suggests that the shape of the 3d city model might also be inaccurate on that side.

Another issue illustrating the importance of the accuracy of the 3D city model is incorrectly positioned and sometimes even missing parts. As shown in the example in figure 6.18, these ones are most likely the result of inaccurate footprints used to create the 3D city model. This can lead to points being wrongly used as seed points (because they lie far above a wrong roof surface) and to points being omitted as they are outside the provided roof boundaries.

Occlusion and small irregular surfaces

Another requirement with regard to the quality of input information results from the need for a minimum number of total points (per roof surface, not only 'deviations)) in order to handle a surface. This is related to the *scipy.spatial.voronoi* library requiring at least 4 initial points to generate a Voronoi diagram (theoretically 2 points are sufficient, but as this library uses simplexes, a minimum of 4 points is imposed).

In some situations, this minimum number of points is not fulfilled. Mainly two factors contributing to this have been identified:
- occlusion in the AHN point cloud (roof surfaces covered by water and thus rather absorbing than reflecting the laser signal, or geometric occlusion at locations off the flightpath).

- particularly small surfaces in the 3D city model, which show a normal point density but still contain less than 4 points.

While the problem with regard to the number of points might be solved by using a higher density point cloud (i.e. flying each flight line in two directions, as done for the point cloud provided by the city of Rotterdam (see part 3.2.1), there is a second issue brought by some of the small surfaces.

In fact, some of the small surfaces are actually no real-world objects but arise as 3D surface reconstruction artifacts. Such surfaces are probably necessary to close a 3D city model when the modeled (main) surfaces do not meet at the right location. An example of such a surface can be found in 6.19. The problem that arises with such surfaces is that, as the region growing algorithm operates within roof surfaces and uses the k-nearest neighbor approach, growing very far from the seed can occur (the closest point might be relatively far away). As such closure surfaces are not always accurate (especially with regard to their orientation), 'deviations' 'filling in' such roof surfaces can sometimes be observed (see figure 6.19).

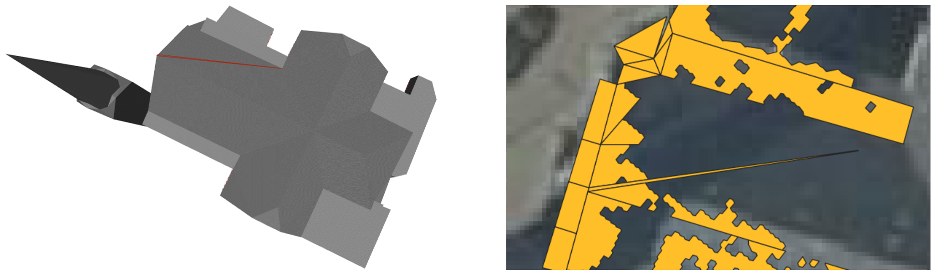


Figure 6.19: Example of a small 'closure' surface occuring in the 3D city model (left, building is located at Portlandstraat 57). The irregular shape can lead to erroneous results as the region growing method 'fills up' and labels the entire shape as 'deviation'.

Roof edge identification

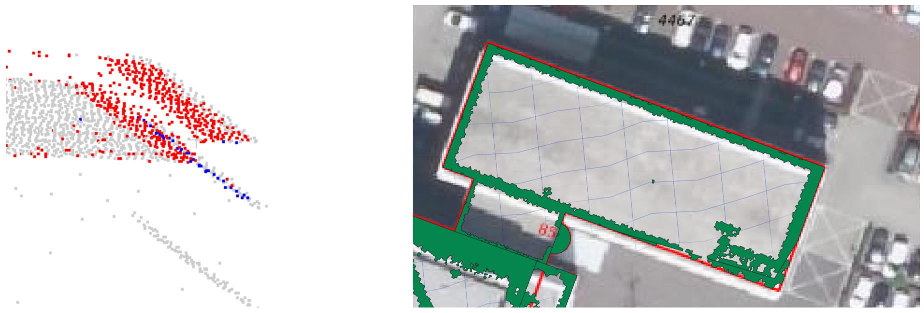


Figure 6.20: Example (Waalhaven O.z. 85) of a building where the method (visualizations with setting 20cm and 5 degrees) leads to an overestimation of the roof edges. As can be seen on the left, the region growing algorithm grows beyong the actual edge object (points in blue are the seeds, points added by region growing are colored in red). The result can be seen on the right. While the edge on the aerial image is estimated at 80cm, the method identifies an edge of about 140cm.

A final observation that was made concerns the edges of roofs. Among the 10 buildings of the subset, edges were entirely identified in 6, partially identified in 3 and entirely missed in only 1 case. This shows that progress can still be made on that level, although it might be challenging as roof edges (e.g. gutters) do sometimes lay in the same plane as the adjacent roof surface. Also, it was regularly observed that roof edge 'deviation' regions grow bigger than the actual roof edge (extending the observations of part 3.3.1). This might be related to the fact that the roof cover surface often changes its inclination already at some distance before the actual edge, or because of the usage of KNN clusters for the estimation of the local plane (for an example see fig 6.20).

6.5 CHAPTER SUMMARY AND LINK TO ASSUMPTIONS MADE

This chapter has performed a quantitative validation of the method developed in the previous chapters. For this sake, a validation area in the south of Rotterdam was chosen, elaborating the representativeness criteria formulated in part 2.1:
- the representativeness in terms of building functions (housing, industrial, community facilities, offices) which is deemed representative for cities with industrial activity.
- the availability of input data, as described earlier in part 3.2 and 5.1.
- budget constraints (one person, i.e. the researcher, and limited time) which apply to any research projects and limited the validation set to 40 buildings here.

The budget constraint also led to limiting the number of variables to two by using constants for the KNN settings, as suggested earlier in part 3.4.4. For the two variables left, a pilot study was performed with one building in order to identify a range of usable values. The maximum and minimum values of the latter ranges were then used as values, resulting in a total of four validation sets (or versions of the method).

VALIDATION IN FOUR WAYS Validation itself was performed in four different ways with the building (once at level B) or the mesh cell from hyperspectral imagery (three times, at level B, C and D) as the unit of analysis. First, only cells located 100% inside a roof surface were considered for nominal validation at building and cell level (A and B). In a second part (C and D), cells lying up to 30% outside a roof surface were considered too, for a subset of 10 buildings only. On one hand, this allowed observing the performance evolution when loosening the cell inside the roof surface criterion. On the other hand, a rational validation (D) was performed on top of the nominal one (by validating the area of the deviations). This one allowed exploration towards potential tolerance for spectral variations and showed that requirements to fulfill by the method in such case increase.

FINDINGS As performance indicators mainly the khat and the commission errors for the class 'clean' were used. Overall, this allowed to identify two different approaches with regard to the thesis:
- if the aim is to accurately quantify materials (thus to find as many clean cells as possible), the first indicator is most important. Basing on the results, it is recommended to use rather loose threshold settings for such endeavor.
- in contrast, the aim is limited to finding the presence of materials (thus to minimize the number of non-clean pixels wrongly classified as clean), the second indicator is most important. Looking at the outcome, it is recommended to use rather strict threshold settings.

LIMITATIONS AND QUALITATIVE OBSERVATIONS Using these findings, it should be kept in mind that they are subject to limitations. In fact, some types of buildings such as particularly big ones were not covered in the validation. Before reusing the results obtained, one should, therefore, ensure that the area of study is sufficiently similar to the set of samples used here.

Moreover, a number of qualitative observations were made next to quantitative validation. These ones stress the fact that input quality matters. By observing results, three topics of attention were identified: non-flat roofs modeled as flat ones, errors in the footprints used as inputs for the 3D models, and presence of irregular 'closing' surfaces.

7 CONCLUSIONS, LESSONS LEARNED AND OUTLOOK

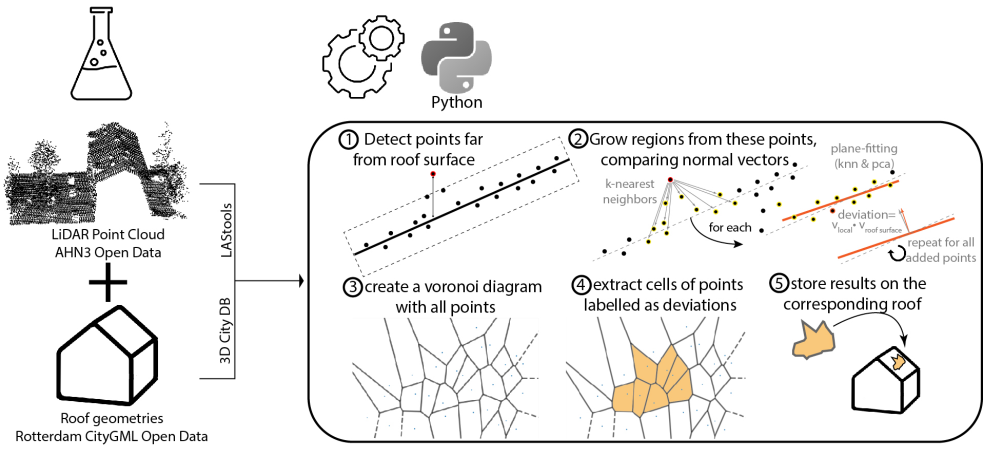
This final chapter will present the conclusions that can be drawn from the research that was presented in the previous chapters. First, the research questions stated in part 1.4 will be answered. The second step will formulate recommendations for data suppliers from the intermediate user perspective of this thesis. Finally, a number of topics of future research will be discussed and related to the developed method.

7.1 ANSWERS TO THE RESEARCH QUESTIONS

In this first subsection, the research question *How can a CityGML LOD2 model be semantically enriched in order to improve material classification performed on roof surfaces?* formulated at the start of the thesis (see part 1.4) will be answered, starting with the subquestions:

Suitability for deviation detection

1. Which method is suitable to identify 'deviations' of LiDAR point clouds compared to LOD2?



Python
LiDAR Point Cloud
AHN3 Open Data
Roof geometries
Rotterdam CityGML Open Data
LAStools
3D City DB
① Detect points far from roof surface
② Grow regions from these points, comparing normal vectors
k-nearest neighbors
for each
plane-fitting (knn & pca)
deviation= $v_{local} \cdot v_{roof\ surface}$
repeat for all added points
③ create a voronoi diagram with all points
④ extract cells of points labelled as deviations
⑤ store results on the corresponding roof

Figure 7.1: Illustration of the method developed for the identification of 'deviations' of LiDAR compared to LOD2

IDENTIFIED METHOD Within this thesis, a method that is able to identify 'deviations' from a geometric point of view has successfully been developed and implemented (for a visual overview, see figure 7.1). This method consists of globally three steps which have been described in the pseudo-code algorithms 1, 3 and 2:

- first, 'deviation' seeds are identified by selecting points exceeding a vertical distance from the roof surface.

- second, these seeds are extended by a region growing approach. This one involves the fitting of local planes using PCA and comparison of the orientations of the locally computed and the roof surface's normal vectors.

- third, the 'deviation' regions (sets of points) are converted into geometry by projecting the points on the roof surface and generating a Voronoi diagram.

The method developed is adaptable as two settings are used: a distance and an angle deviation threshold (see part 3.4.4 for more details). These ones can be fine-tuned to external factors such as point cloud noise or quality of the 3D city model.

GEOMETRIC LIMITATIONS It should nevertheless be noted that the method does only provide limited support for buildings that are complex from a 3D geometric point of view. Depending on the definition of the region of interest, only limited support for roofs with intrusions can be provided. Also, some particular complex cases with roof geometries located in the region of interest of another roof geometry might lead to inaccurate results.

7.1.1 Requirements with regard to data inputs

2. What are the requirements with regard to CityGML LOD2, LiDAR point clouds and hyper-spectral imagery data?

CITYGML LOD2 Relatively few quality requirements could be found in the tender document specifying the requirements for the CityGML model of Rotterdam (e.g. no accuracy requirement of the roof surfaces with regard to the point cloud, see part 3.2.2 and Gemeente Rotterdam (2016, n.d.)). Nevertheless, the 3D CityGML model of Rotterdam fulfills the requirements of the method rather well. While it is clear that a LOD1 model would not suffice, the precision of the LOD2 model provided is sufficient, showing that a rather expensive LOD3 model was not needed for this research.

In three cases of the 41 buildings analysed in chapter 6, the roof surfaces were wrongly located with regard to the point cloud (in one case a roof was modeled as flat while they are slightly sloped, see part 6.4). However, even in these cases, few clean pixels were still left and would have allowed the identification of material presence.

This observation is true for the settings with which the algorithm was validated. Therefore, in all the 38 other cases, the roof surfaces were located within 20cm of the points representing it. As the point cloud has up to 5cm of stochastic and up to 5cm of systematic errors, this can be seen as a rather good match. This is especially true as the point cloud used to make this observation is not the one from which the 3D city model was created.

Some more critical inaccuracies of the 3D city model were found and covered in part 6.4 and 6.4. Overall, they show that both the input but also the surface reconstruction quality matter and can strongly alter the results of the method developed in this research.

LiDAR POINT CLOUDS Although two point clouds were available, this research has shown that the AHN point cloud with 8 points/m^2 is of sufficient density to obtain 'suitable' results with the method (see the answer to sub-question 3 for more details). While the point cloud density is relatively low and occlusions exist, it still allowed the method to run correctly.

The *Nyquist-Shannon sampling theorem* [1] states that the size of an object that can reliably be detected (in one dimension) by a number of points n per m is given by:

$$\frac{2}{n} \tag{7.1}$$

If objects of 1 m^2 should be (reliably) detected, this reasoning translates to a requirement of 4 points/m^2 (for an object of 0.5 m^2 it translates to 8.2 points/m^2). As the point cloud distribution might locally be affected by occlusion, a margin should be applied (e.g. increasing the requirements by a given percentage, preferably by

1 for more information see (Claude and Shannon, 1934)

performing density variation measures on the point cloud directly). For a margin of 25%, the current point cloud density (6.4 points/m^2 after removal of 25%) should thus be able to (reliably) detect objects with a size of at least about 0.6 m^2.

The requirement in terms of density depends on the desired application: e.g. usage of 30 points/m^2 as acquired by the city of Rotterdam would allow detecting objects as small as 0.17 m^2 (incl. a density variation margin of 25%). One might, however, note, that where occlusion occurs, identification of 'deviations' becomes impossible. For this method, the usage of LiDAR point clouds acquired with two crossing flight lines is therefore recommended. The point cloud density should be adapted to the size of the objects to be detected.

Another important aspect is the pre-processing of the point cloud data. In fact, classification has to be performed with care as the confusion of objects such as chimneys with outliers can make the identification of 'deviations' impossible (as shown in part 3.2.1).

An alternative might be to use unclassified data but would come at the expense of heavier computation, especially during the cropping of the point cloud and the selection of the region of interest. A similarly heavier computation would also be induced by a higher density point cloud but might be compensated to some extent by thinning.

HYPERSPECTRAL IMAGERY Although the mesh cells resulting from the hyperspectral imagery data often resulted in at least one clean cell for a given building, whether this fulfills the requirements can only be said by looking at the application. Whether one desires to quantify materials (e.g. in m^2) or just identify their presence makes quite a difference (as mentioned in part 6.3).

In case the objective is to get nominal data on the presence of materials, the flight height of 7000m and the 4×4m resolution can be deemed sufficient. In the cases where rational data quantifying the materials is to be obtained, a 4×4m resolution leads to a considerable amount of cells that are excluded (e.g. which are only partially located on the roof and thus contain mixed information).

Obviously, the suitability of a 4×4m resolution also depends on the roof surface sizes of the buildings for which the data should be obtained. On one hand, even obtaining nominal data can be hard for small buildings (e.g. bungalows), while on the other hand, quantitative data can be reliably obtained for very big buildings (e.g. logistic halls).

LINKS BETWEEN REQUIREMENTS A number of links between the different dataset requirements exist. In these situations, the requirement must be formulated more strictly or loosely depending on the requirement formulated for the related dataset. In fact, the sum of height errors should not exceed the strictest distance threshold used for running the method. In the case of the validation, the sum (according to the dataset specifications) is 20 cm and thus at the limit with regard to the threshold used.

As mentioned in part 1.5.2, an alternative to LiDAR acquisition is the production of point clouds using dense image matching. However, these point clouds are generally of lower accuracy, which would increase the minimum usable height and angle thresholds. While point clouds from dense image matching are possible, their usage would limit the extent to which deviations can be identified (e.g. higher minimum height differences with regard to the roof surface - thus only detection of bigger objects).

A second link exists between the point cloud density (and the minimum deviation area to identify) and the area covered by the pixels of the hyperspectral imagery. In fact, the deviation areas to be identified should have a relevant size with regard to the *pepper and salt effect* (oversensibility) that might occur in spectral deviations. A critical threshold percentage for this effect was not known to the author at the time of writing. As an example, a single deviation of 0.2 m^2 only represents 1.25% of the area of a cell of 16 m^2.

Obviously, several objects with an area smaller than what can be detected reliably might exist and potentially be missed. However, if these 'deviation' objects are randomly distributed, the chances that at least one gets detected increases with their number. Also, as the area estimation uses the Voronoi diagram, the area of the latter identified objects should be overestimated, potentially compensating for the ones located in between points. Therefore, the point cloud density requirement should be defined with regard to the ground resolution of the hyperspectral imagery.

Another requirement that should be defined with regard to this is the positional accuracy of the point cloud (which, according to the specifications of the AHN used in this research is not higher than 50 cm). In fact, not only the positional accuracy of the mesh cells (resulting from the orthocorrection performed in part 5.2) but also of the point cloud itself can lead to a share of the cell area being misplaced. Here too, a critical threshold should preferably be known. As has been seen in part 5.2, a positional error of 11cm can already lead to 5.4% of misplaced cell area.

7.1.2 Suitability for identification of clean pixels

3. To which extent does such a method support the identification of clean pixels?

VALIDATION RESULTS The validation performed in chapter 6 was preceded by an in-depth study of the acquisition technology to establish the link between the imagery and geographic space (see part 5.2). The results show that the method delivers a moderate agreement with ground truth (defined by Congalton and Green (2002) as a khat from 40 to 80%)). With a distance threshold setting of 40cm (looser setting), better results are obtained than with 20cm (from an overall accuracy point of view). This shows that the method proposed in this thesis has the potential for the identification of clean pixels, but can still be enhanced, at least from an overall accuracy point of view.

A different perspective might be taken if the aim is to detect the material presence rather than the quantities (as mentioned in part 6.3). In that case, the commission error for clean pixels becomes more important. Focusing on this aspect, the proposed method performs moderately (commission errors as low as 10%), and best with the strictest setting of 20cm and 2-degree thresholds.

Another aspect that was studied is the possibility for hyperspectral imagery classifiers to become to some degree tolerant for non-clean pixels. In order to identify the extent to which this assumption is true, further research into mixed cells (such as Guo et al. (2009)) and the specific spectra of roof materials are required. In that case, pixels would have to be considered as rational instead of a nominal data type. The fact that the performance of the method is lower for such data confirms the higher requirements. While a potential does also exist here, even stronger improvements would thus be needed to support such data type.

VALIDATION LIMITATIONS The first, probably strongest limitation of the validation is the composition of the validation set. As has been shown by the qualitative observations on big buildings that could not be included (see part 6.4), the results obtained are rather specific for the buildings composing the set. The validity of the findings of the validation for other study sets is limited thereby.

A second limitation is that time-wise budget constraints only allowed the validation to be carried out with four sets, while the method has four variables. Therefore, it was chosen to set a constant for two of the variables and to allow two different values for the two others.

To determine the variable values used in the validation, reasonable minimum and maximum values were identified first. This was done by visual evaluation of computation results, performed on sample buildings. For the variables allowing two values (height and angle tolerance thresholds), the reasonable minimum and maximum values were selected. For the two other variables which were set to a constant, a reasonable minimum value was chosen (based on the criteria established in the deviation identification part: completeness and computational load).

If additional time-wise budget was provided, more conclusive results might have been obtained by testing more variable values. Retrospectively, it should nevertheless be noted that the value choice using the reasonable minimum and maximum values did lead to conclusive findings. In fact, the validation allowed to define two approaches, two situations in which either the 'strict' or the 'loose' settings are more suited.

IMPROVEMENTS NEEDED TOWARDS IMPLEMENTATION IN HYPERSPECTRAL IMAGERY CLASSIFICATION Furthermore, one has to note that the scope of the thesis was limited with regard to the definition of clean pixels. In practice, not only 'deviations' but also shadow resulting from the deviations and other geometries would have to be taken into account. This aspect was not considered in the research but is discussed in part 7.3.2. Another aspect which it is desirable to improve before implementation is the orthocorrection. As it was beyond the scope, this study satisfied itself with an approximate but acceptable implementation. If the method is to be used in hyperspectral imagery classification, it becomes necessary to perform a more exact orthocorrection (i.e. using a digital surface model and the image acquisition coordinates to solve collinearity equations).

7.1.3 Main question: Enrichment of a CityGML LOD2 with regard to material classification

Finally, the main question *How can a CityGML LOD2 model be semantically enriched in order to improve material classification performed on roof surfaces?* will be answered.

SUMMARY OF ANSWERS TO THE RESEARCH QUESTIONS As shown by answering the sub-questions, a LOD2 3D city model such as the one of Rotterdam can be semantically enriched by indicating the 2-dimensional locations of height 'deviations'. As the scope of the study focused on the data fusion with hyperspectral imagery, the height 'deviations' were not stored in the CityGML 3D city model but directly combined with the imagery information. This is, in fact, a form of semantic enrichment as the geometry of the city model is not altered - only additional 2D information (including geometric shapes) that can be used in combination with the 3D city model is generated and stored. Options for a semantic enrichment integrated with the CityGML standard were also explored (more specifically, the usage of *GenericCityObject* or *BuildingInstallation* classes) but not implemented.

The potential of this approach as an alternative to other ones (e.g. majority filter as presented in part 1.2.2) has been shown by the results obtained in chapter 6. This potential is stronger for a nominal approach (clean vs. non-clean pixels) than for a rational (degree of cleanliness of a pixel). Furthermore, the different validation showed the need to always keep the actual application in mind. In fact, whether one wants to identify the mere presence of materials or perform a quantification makes quite a difference in terms of settings identified as optimal. An additional aspect affecting results is the sizes and typologies of the validation building set. These aspects must definitely be included in any kind of specifications to treasure the value behind the numbers. If they are omitted, the results lose meaning and it becomes hard to draw conclusions for specific applications.

POSSIBLE VARIANT OF THE METHOD One should note that a simplified variant of the method developed in this thesis is possible. In fact, the conversion of the points into geometries might be skipped and the cleanliness of cells assessed by the number of (projected) 'deviation' points they contain. This is likely to allow faster computation, especially as it skips the conversion which requires the Voronoi diagram to be generated for all (labeled and non labeled) points of a roof surface.

However, two limitations can be noted here. First of all, skipping this step might become problematic in the case of globally or locally low point densities. In fact, the creation of the Voronoi diagram intrinsically adapts the output to the local point density. By doing so it also allows a point located close to the edge of a cell to affect neighboring cells. This characteristic would be lost if only counting the points located inside a cell. How critical this characteristic is depends on several factors discussed earlier in part 7.1.1.

A second limitation which is why this variant was not implemented here is that it is harder to validate. In fact, this would require to make a link between the point count and the area of a deviation. Using a constant value would be approximate as LiDAR point cloud density varies locally (distance from the flight line, occlusion). More precise would be the computation of a local point density to make the link more specific. However, as such a value is necessarily an average over a given area, it would still be less precise than the chosen method where the Voronoi diagram only takes into account the direct neighbors.

COHERENCE OF CHOICES AND ASSUMPTION FRAMEWORKS Overall, the guiding criteria, namely 'completeness' that was formulated in part 2.1 has been considered in all parts of the method (and within the limits of the scope). For each specific step, it was translated for the context and implemented accordingly. Considering that the validation did not only take into account geometric (as defined by the scope) but also other visual deviations, the positive results are proof that the 'completeness' criterion was taken into account correctly.

Among the secondary criteria used, different ones had to be chosen for the respective parts. While the overall aim was to make the realization of the research feasible, this had to be translated into several goals. On one hand, the computational load was important to ensure that results (during the different research phases) were obtained in a reasonable time. On the other hand, time-wise budget constraints (along with the need for a representative sample) only allowed a limited number of validation sets. For this reason, the number of input variables was limited, either by identifying reasonable constants (see part 3.4.4) or by implementing methods not requiring external input variables (see parts 4.4 and 5.4). This led to ambiguous choices concerning the secondary criteria. For instance, the computational load criterion was replaced by automation for the second and third step. This clearly shows that multiple goals existed beyond the primary one and a balance had to be found by making compromises.

7.2 RECOMMENDATIONS TO DATA SUPPLIERS

7.2.1 LiDAR acquisition and processing

Some formats in which point clouds are usually delivered (i.e. *.las* format) support metadata information such as the software used for acquisition and classification, but it is not a mandatory field (American Society for Photogrammetry & Remote Sensing, 2013). Furthermore, in the case of fully automated classification, such software can be proprietary - therefore not giving any insights on how the classification was performed. In some of the available software packages (e.g. TerraScan), the user is simply given a number of tools to create a classification workflow.

Sharing information about how the classification was performed is a tricky subject, but the issue encountered in this thesis (see part 3.2.1) shows that it can be critical. A reflection among practitioners' is therefore on how such information could be shared is therefore encouraged. In some cases, an attribute indication such as 'strict - medium - loose' might be sufficient while in others a detailed technical report describing the workflow might be more desirable.

7.2.2 CityGML structure

The CityGML format and especially its semantics have proven to be a vital element for research such as performed in this thesis. In fact, the storage of both geometric and thematic, aggregated and detailed information makes it much easier to work with a 3D city model.
Looking at the urban mining context in which this thesis is written, one can observe that there is no support for the storage of material quantities in CityGML yet. As discussed in part 3.1.2, some ADEs do support the storage of material information. However, to the knowledge of the author, no ADE does cover the actual quantification (e.g. in kgs or m^3) which might ultimately be the output of hyperspectral classifications (and should be retrievable both at the surface and aggregated at building level). One might argue that the CityGML format is in principle extensible by the usage of generic attributes. However, the inclusion of material information for the field of urban mining as a standard (of an ADE or of the CityGML format itself) is desirable to treasure interoperability.

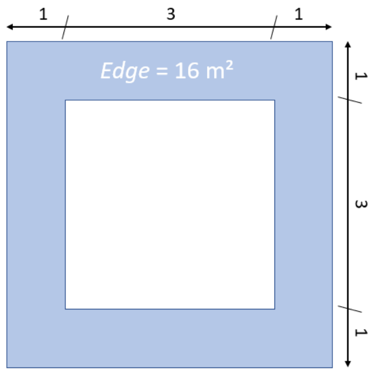


Figure 7.2: Schema showing the ambiguity of modeling roof edges with regard to the CityGML LOD standards. The edge here has a total attached area of 16 m^2 and should thus be modeled. But if it split in two parts, the attached area would be too low to be modeled while the edge might be nearly as relevant.

Another topic mentioned in this thesis which is not covered by the CityGML standard is the thickness of roof edges. While overhanging roof parts are explicitly addressed in chapter 6.2. of the standard (allowed from LOD2, mandatory from LOD3 on, Gröger et al. (2012)), this is not the case for roof edges. Especially in the case of flat buildings, these ones might have a considerable width (e.g. up to 1m) and show a sufficient height difference. Chapter 6.2. of the Open Geospatial Consortium standard (Gröger et al., 2012) does suggest for LOD2 to model roof objects with dimensions of at least 4×4m - however it is unclear if this therefore also applies to roof edges (e.g. a roof edge of 1m width and 50cm height difference on a roof surface of 6×6m, thus also covering a size of $16m^2$ - see figure 7.2). More precise addressing of such questions in future versions of the standard is thus strongly encouraged (instead of the attached area criterion, a width criterion might, for instance, apply to roof edges).

7.2.3 CityGML dataset of Rotterdam

Based on the qualitative observations made during the research (such as in part 6.4) , the following recommendations can be made for the CityGML dataset of Rotterdam:
- As the 3D city model is provided as open data, the technical specifications should

also be openly available. Additional information such as the acquisition date of the point cloud used, the custom specification used for LOD2 (see part 3.2.2) and the positional accuracy of roof surfaces (contained in Gemeente Rotterdam (n.d.) and Gemeente Rotterdam (2016)) should be indicated in a technical specification file for the users.

- If possible, results of the quality checks should be published too as they give a better idea of the end product quality than the specifications do. As a custom definition of LOD2 is used(see part 3.2.2), some cases where the definition was more than respected and other cases where it was not respected were encountered. The qualitative observations induce that it is hard to estimate the extents of this (although, the method of this thesis might be adapted to do so, see part 7.3.1). In some other cases such as the ship MS Rotterdam in 7.3, it is rather obvious that the model is of a LOD3 kind.
- Furthermore, it is unclear whether a tolerance was used for the modeling of flat roofs. The maximum slope/curvature for which a flat roof is accepted should be formulated in the technical specifications and shared with the user too.
- A final recommendation is that small surfaces that are modeling artifacts serving as a connection between two surfaces should be avoided. An example was given in part 6.4. Such artifacts can be problematic as it is hard for processing methods such as the one of this thesis to distinguish between such surfaces and real-world surfaces.

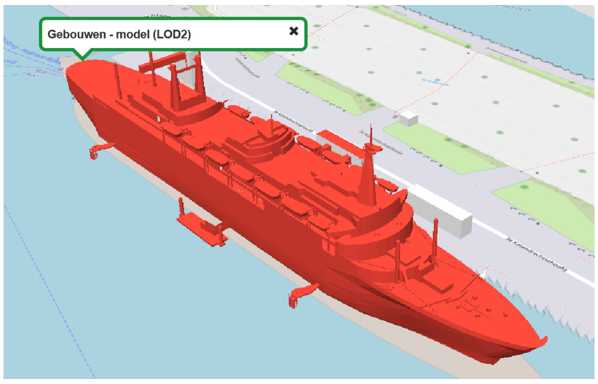


Figure 7.3: Example of a 'building' (actually a ship) that is labeled as LOD2 while it rather seems to fulfill LOD3 standards.

7.3 FUTURE RESEARCH

7.3.1 Automated quality checks

During a visit in March 2019 at the department in charge of *Rotterdam3D* within the municipality of Rotterdam, the topic of quality checks was discussed. In fact, at the moment of the visit - the quality checks were performed manually. A subset was selected, of which the 3D city model was displayed together with the point cloud. A human person did then estimate the extent to which the 3D city model complies with the requirements. While this process is not wrong as such, it is probably rather expensive.

The research performed in this thesis provides to some extent a framework for the automation of such quality checks. Roof 'deviations' which are in fact objects that were not appropriately modeled could be identified automatically - guiding the human in the quest for buildings that need improvements. In some cases of the validation in chapter 6, building models with a strong mismatch with the point cloud (e.g. wrong position of the roof surface) were observed. In these cases, a substantial part of the roof surface was identified as 'deviation'. Therefore, the share of the roof classified as 'deviations' could then be used as an indicator for the severeness of the case.

Nevertheless, the method proposed in this thesis might require modifications if used for this purpose. In fact, the roof edges which are only supported from LOD3 on according to the CityGML standard might trigger 'wrong errors'. As discussed with figure 7.2, edges might have the size of missed elements but still fall under LOD3.

7.3.2 More accurate estimation of shadow and solar potential

A recent study of Willenborg et al. (2018) has shown that the enrichment of semantic 3D city models with additional details can considerably improve the estimation of the solar potential for building roofs and walls. In this research, the approach differed from the one of this thesis as the 3D mesh used is already a geometric object, in contrast with the point cloud used as input here. Interestingly, the research resembles the part described in chapter 3 as it also uses first seed selection and subsequently a region growing.

A major difference with regard to this research is that the identification of shadow would require a third geometric dimension. While the first steps have been done in that direction by storing basic statistics (see section 4.3.4), a separate validation would be required too. A basic validation could be done using satellite or aerial images, preferably several ones taken with different sun positions. For a more advanced validation, site visits or highly accurate 3D models (e.g. Building Information Models (BIM)) would be required.

Also, at the current stage, only one height value has been stored for each 'deviation' region. Whether this approximation is sufficient depends on the application and the required level of detail. It might also be possible to group Voronoi cells by similar heights within a 'deviation' region - therefore supporting several values.
Another difference with regard to the research conducted in this thesis is that a more global approach would be required. In fact, while the visibility of roofs is generally not altered by objects external to the roof (with exception of some relatively low buildings with a tree growing above), the same cannot be said for shadow. For solar studies, neighboring trees and buildings need to be taken into account even if they are located outside the footprints of the building under study. Depending on the time of the day and the altitude of the sun, shadows are generally cast onto the surroundings of the object.

Moreover, future research on solar potential and shadow estimation would also be in line with the identification of clean pixels. In contrast with the validation performed in chapter 6, a clean pixel in the ground truth would then be defined as being a pixel that does neither contain several materials nor shadow.

7.3.3 Towards LOD3?

A final topic of future research is the extent to which this thesis' method can be adapted to build LOD3 models (thus containing objects below the custom detail size threshold set for LOD2). In order to do so, the following steps might be followed:

- first, the 2D deviation shapes that are currently obtained need to be rectified to match the shapes of real-world geometric objects. For this, line simplification algorithms might be used.

- second, the 2D shapes have to be projected on the roof surface. This area then has to be cut out so that the boundary surface can follow the shape created in the subsequent steps.

- third, the height attribute obtained in this thesis (see part 4.3.4) might then be used to vertically extrude the deviation geometry, hereby making a 3D object that extends the boundary surface.

One can note that this is a very basic approach that leads to the deviation being modeled as a rough 3D estimation. In fact, it resembles a bit to the LOD1 framework where a single height is available for each geometry - but then applied to detailed geometries within existing LOD2 models. In this case, the single height value implies that the top geometry of the 3D deviation is always coplanar to the roof surface it lies on (see figure 7.4 for an example). This approximation is often wrong (e.g chimneys) and more research would be needed to make such estimations more reliably (e.g. support several height values, check whether the increase in height values matches the one of the roof surface). In-depth research might also allow the support of deviations with several or non-flat top surfaces.

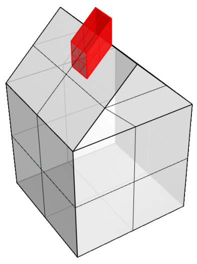


Figure 7.4: Example of a geometry that might be created with the results of this research, making a step from LOD2 to LOD3

.

BIBLIOGRAPHY

Agugiaro, G., Benner, J., Cipriano, P., and Nouvel, R. (2018). The energy application domain extension for citygml: enhancing interoperability for urban energy simulations. *Open Geospatial Data, Software and Standards*, 3(1):2.

AHN (2015). Besteksvoorwaarden inwinning landsdekkende dataset ahn 2014-2019.

American Society for Photogrammetry & Remote Sensing (2013). Las specification version 1.4–r13. In *"American Society for Photogrammetry & Remote Sensing"*.

Asaeedi, S., Didehvar, F., and Mohades, A. (2013). Alpha-concave hull, a generalization of convex hull. *arXiv preprint arXiv:1309.7829*.

Baccini, P. and Brunner, P. H. (2012). *Metabolism of the anthroposphere: analysis, evaluation, design*. MIT Press.

Bentley, J. L. (1975). A survey of techniques for fixed radius near neighbor searching. Technical report, Stanford Linear Accelerator Center.

Beth, R. T. (2016). Uav based hyperspectral imaging of river ecosystems. ULR https://www.researchgate.net/profile/Roberto_Beth/publication/318130142_UAV_Based_Hyperspectral_Imaging_of_River_Ecosystems/links/595b7bc2a6fdcc36b4dc282b/UAV-Based-Hyperspectral-Imaging-of-River-Ecosystems.pdf.

Beyer, K., Goldstein, J., Ramakrishnan, R., and Shaft, U. (1999). When is "nearest neighbor" meaningful? In *International conference on database theory*, pages 217–235. Springer.

Billen, R., Cutting-Decelle, A.-F., Marina, O., de Almeida, J.-P., Caglioni, M., Falquet, G., Leduc, T., Métral, C., Moreau, G., Perret, J., et al. (2014). *3D City Models and urban information: Current issues and perspectives*. edp Sciences Les Ulis, France.

Botsch, M., Kobbelt, L., Pauly, M., Alliez, P., and Lévy, B. (2010). *Polygon mesh processing*. AK Peters/CRC Press.

Claude, E. and Shannon, C. (1934). Communication in the presence of noise. *Proc. Inst. Radio Eng*, 371.

Congalton, R. G. and Green, K. (2002). *Assessing the accuracy of remotely sensed data: principles and practices*. CRC press.

De Berg, M., Van Kreveld, M., Overmars, M., and Schwarzkopf, O. (1997). Computational geometry. In *Computational geometry*, pages 1–17. Springer.

de Miguel, E., Fernández-Renau, A., Prado, E., Jiménez, M., de la Cámara, Ó. G., Linés, C., Gómez, J. A., Martín, A. I., and Muñoz, F. (2014). The processing of casi-1500i data at inta paf. *EARSeL eProceedings*, 13(1):30–37.

Demarchi, L., Canters, F., Cariou, C., Licciardi, G., and Chan, J. C.-W. (2014). Assessing the performance of two unsupervised dimensionality reduction techniques on hyperspectral apex data for high resolution urban land-cover mapping. *ISPRS Journal of Photogrammetry and Remote Sensing*, 87:166–179.

e2v technologies (n.d.). Ccd55-30 inverted mode sensor high performance ccd sensor technical specifications. https://www.teledyne-e2v.com/shared/content/resources/File/documents/Imaging%202017/CCDs%20-%20Full-Frame%20Spectroscopic%20&%20Scientific/CCD55-30/1.%20FI,%20AIMO/1274.pdf.

Edelsbrunner, H., Kirkpatrick, D., and Seidel, R. (1983). On the shape of a set of points in the plane. *IEEE Transactions on information theory*, 29(4):551–559.

Edelsbrunner, H. and Mücke, E. P. (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)*, 13(1):43–72.

ESA Earth Observation Portal Directory (n.d.). Apex (airborne prism experiment). URL https://earth.esa.int/web/eoportal/airborne-sensors/apex.

Fieque, B., Jamin, N., Chorier, P., Pidancier, P., Baud, L., and Terrier, B. (2012). New sofradir visir-swir large format detector for next generation space missions. In *Sensors, Systems, and Next-Generation Satellites Xvi*, volume 8533, page 853313. International Society for Optics and Photonics.

Frick, A. (2007). *Beiträge höchstauflösender Satellitenfernerkundung zum FFH-Monitoring-Entwicklung eines wissensbasierten Klassifikationsverfahrens und Anwendung in Brandenburg*. PhD thesis, Technische Universität Berlin, Fakultät VI - Planen Bauen Umwelt.

Gemeente Rotterdam (2016). Technische specificaties hoogtebestand gemeente rotterdam 2016.

Gemeente Rotterdam (n.d.). Programma van eisen 3d rotterdam.

Gilani, S. A. N., Awrangjeb, M., and Lu, G. (2016). Robust building roof segmentation using airborne point cloud data. In *Image Processing (ICIP), 2016 IEEE International Conference on*, pages 859–863. IEEE.

Gröger, G., Kolbe, T. H., Nagel, C., and Häfele, K.-H. (2012). Ogc city geography markup language (citygml) encoding standard, version 2.0. *OGC Doc*, (12-019).

Guo, Z., Wittman, T., and Osher, S. (2009). L1 unmixing and its application to hyperspectral image enhancement. In *Algorithms and Technologies for Multispectral, Hyperspectral, and Ultraspectral Imagery XV*, volume 7334, page 73341M. International Society for Optics and Photonics.

Heiden, U., Segl, K., Roessner, S., and Kaufmann, H. (2007). Determination of robust spectral features for identification of urban surface materials in hyperspectral remote sensing data. *Remote Sensing of Environment*, 111(4):537–552.

Kuhlmann, G., Hueni, A., and Brunner, D. (2016). High-resolution no2 maps of rotterdam and zürich retrieved from the apex imaging spectrometer. In *EGU General Assembly Conference Abstracts*, volume 18.

Lemmens, M. (2011). *Geo-information: technologies, applications and the environment*, volume 5. Springer Science & Business Media.

Müller, D. B. (2006). Stock dynamics for forecasting material flows—case study for housing in the netherlands. *Ecological Economics*, 59(1):142–156.

Moreira, A. and Santos, M. Y. (2007). Concave hull: A k-nearest neighbours approach for the computation of the region occupied by a set of points. *International Conference On Computer Graphics Theory and Applications*.

Nguyen, A. and Le, B. (2013). 3d point cloud segmentation: A survey. In *RAM*, pages 225–230.

Nowicki-Bringuier, Y.-R. and Chorier, P. (2009). Sofradir swir hyperspectral detectors for space applications. In *Sensors, systems, and next-generation satellites XIII*, volume 7474, page 747417. International Society for Optics and Photonics.

Nurunnabi, A., Belton, D., and West, G. (2012). Robust segmentation in laser scanning 3d point cloud data. In *Digital Image Computing Techniques and Applications (DICTA), 2012 International Conference on*, pages 1–8. IEEE.

Patouillard, L., Bulle, C., Querleu, C., Maxime, D., Osset, P., and Margni, M. (2018). Critical review and practical recommendations to integrate the spatial dimension into life cycle assessment. *Journal of Cleaner Production*, 177:398–412.

Priem, F. and Canters, F. (2016). Synergistic use of lidar and apex hyperspectral data for high-resolution urban land cover mapping. *Remote sensing*, 8(10):787.

Schaepman, M. E., Jehle, M., Hueni, A., D'Odorico, P., Damm, A., Weyermann, J., Schneider, F. D., Laurent, V., Popp, C., Seidel, F. C., et al. (2015). Advanced radiometry measurements and earth science applications with the airborne prism experiment (apex). *Remote Sensing of Environment*, 158:207–219.

Tanikawa, H. and Hashimoto, S. (2009). Urban stock over time: spatial material stock analysis using 4d-gis. *Building Research Information*, 37(5-6):483–502.

Thiel, F. (2016). Classifying anthropogenic and natural urban surface types using imaging spectrometer and lidar data. Bachelor thesis at Humboldt University Berlin, submitted in 2016.

Townsend, T., Powell, J., and Xu, C. (2007). Environmental issues associated with asphalt shingle recycling. *Construction Materials Recycling Association, US EPA Innovations Workgroup*.

Tribolet, P. and Chorier, P. (2002). Large infrared focal plane arrays for space applications. In *PROCEEDINGS OF THE INTERNATIONAL SYMPOSIUM ON SPACE TECHNOLOGY AND SCIENCE*, volume 23, pages 2247–2255. Citeseer.

Tryfona, M.-S. (2017). Bidirectional enrichment of citygml and multi-view stereo mesh models. Master's thesis, Delft University of Technology, Faculty of Architecture.

University of California Berkeley (non dated). 7.4 non-conventional classification algorithms. URL https://nature.berkeley.edu/~penggong/textbook/chapter7/html/sect74.htm.

Vosselman, G., Gorte, B. G., Sithole, G., and Rabbani, T. (2004). Recognising structure in laser scanner point clouds. *International archives of photogrammetry, remote sensing and spatial information sciences*, 46(8):33–38.

Vosselman, G. and Maas, H.-G. (2010). *Airborne and terrestrial laser scanning*. CRC.

Vreys, K., Iordache, M.-D., Biesemans, J., and Meuleman, K. (2016). Geometric correction of apex hyperspectral data. *Miscellanea Geographica*, 20(1):11–15.

Wang, J. and Shan, J. (2009). Segmentation of lidar point clouds for building extraction. In *American Society for Photogramm. Remote Sens. Annual Conference, Baltimore, MD*, pages 9–13.

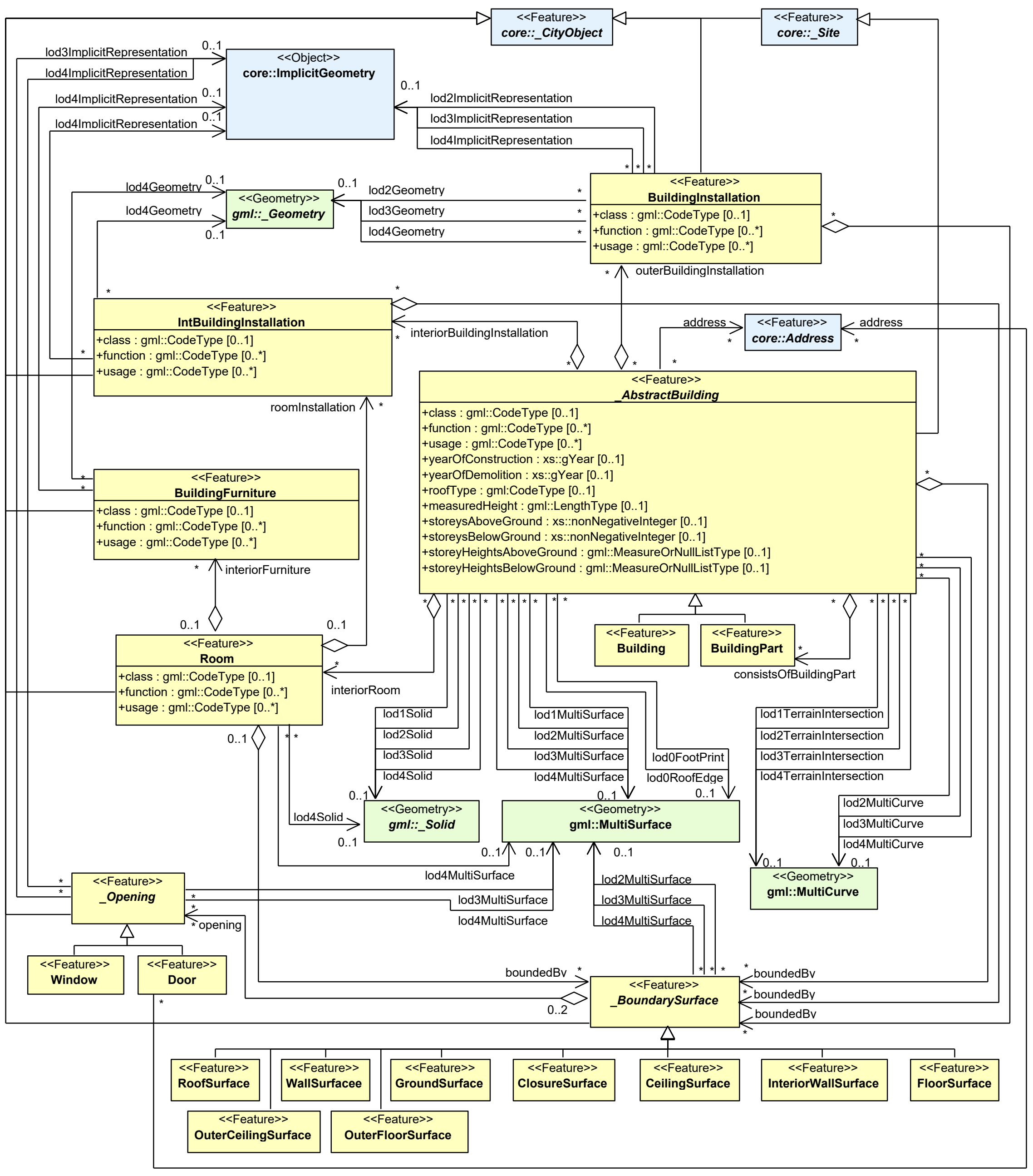
Willenborg, B., Pültz, M., and Kolbe, T. H. (2018). Integration of semantic 3d city models and 3d mesh models for accuracy improvements of solar potential analyses. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*.

Zhang, J. (2010). Multi-source remote sensing data fusion: status and trends. *International Journal of Image and Data Fusion*, 1(1):5–24.

Zhou, K., Gorte, B., Lind enbergh, R., and Widyaningrum, E. (2018). 3d building change detection between current vhr images and past lidar data. *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:2.

8 ANNEXES

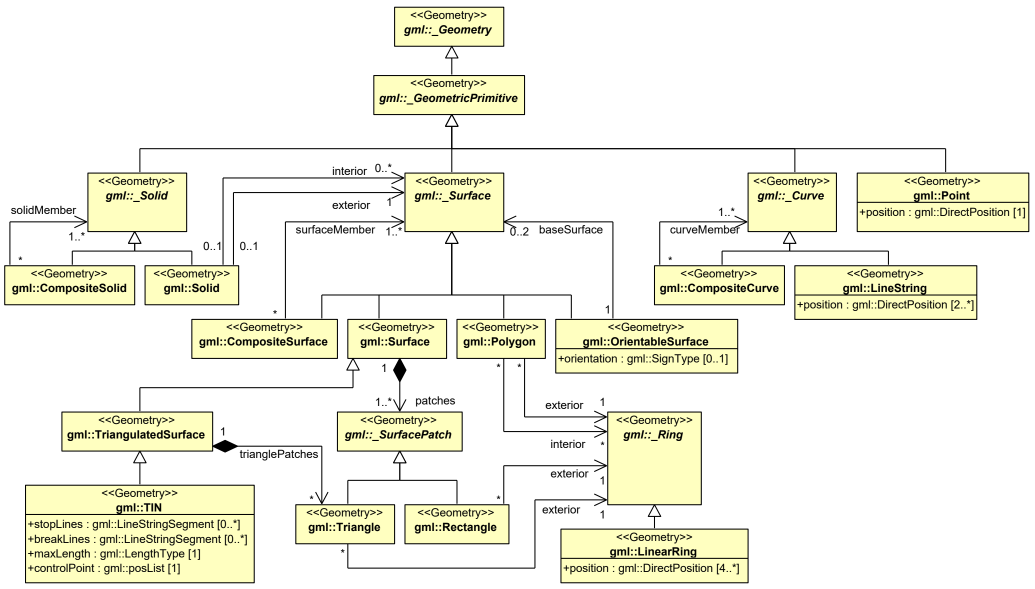
8.1 UML DIAGRAM OF CITYGML BUILDING CLASS.



<<Feature>> core::_CityObject
<<Feature>> core::_Site
<<Object>> core::ImplicitGeometry
lod3ImplicitRepresentation 0..1
lod4ImplicitRepresentation
lod4ImplicitRepresentation 0..1
lod4ImplicitRepresentation 0..1
0..1
lod2ImplicitRepresentation
lod3ImplicitRepresentation
lod4ImplicitRepresentation
* * *
<<Geometry>> gml::_Geometry
lod4Geometry 0..1
lod4Geometry 0..1
0..1
lod2Geometry *
lod3Geometry *
lod4Geometry *
<<Feature>> BuildingInstallation
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
* outerBuildingInstallation
<<Feature>> IntBuildingInstallation
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
* interiorBuildingInstallation
address * <<Feature>> core::Address address *
roomInstallation *
<<Feature>> _AbstractBuilding
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
+yearOfConstruction : xs::gYear [0..1]
+yearOfDemolition : xs::gYear [0..1]
+roofType : gml::CodeType [0..1]
+measuredHeight : gml::LengthType [0..1]
+storeysAboveGround : xs::nonNegativeInteger [0..1]
+storeysBelowGround : xs::nonNegativeInteger [0..1]
+storeyHeightsAboveGround : gml::MeasureOrNullListType [0..1]
+storeyHeightsBelowGround : gml::MeasureOrNullListType [0..1]
<<Feature>> BuildingFurniture
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
* interiorFurniture
0..1 0..1
<<Feature>> Room
+class : gml::CodeType [0..1]
+function : gml::CodeType [0..*]
+usage : gml::CodeType [0..*]
interiorRoom
<<Feature>> Building
<<Feature>> BuildingPart
consistsOfBuildingPart
lod1Solid
lod2Solid
lod3Solid
lod4Solid
lod1MultiSurface
lod2MultiSurface
lod3MultiSurface
lod4MultiSurface
lod0FootPrint
lod0RoofEdge
lod1TerrainIntersection
lod2TerrainIntersection
lod3TerrainIntersection
lod4TerrainIntersection
lod2MultiCurve
lod3MultiCurve
lod4MultiCurve
0..1 * *
lod4Solid
0..1
<<Geometry>> gml::_Solid
0..1
<<Geometry>> gml::MultiSurface
0..1 0..1
0..1 0..1 0..1
<<Geometry>> gml::MultiCurve
0..1 0..1
lod4MultiSurface
lod2MultiSurface
lod3MultiSurface
lod4MultiSurface
lod3MultiSurface
lod4MultiSurface
<<Feature>> _Opening
* opening
<<Feature>> Window
<<Feature>> Door
boundedBy *
0..2
<<Feature>> _BoundarySurface
boundedBy
boundedBy
boundedBy
<<Feature>> RoofSurface
<<Feature>> WallSurfacee
<<Feature>> GroundSurface
<<Feature>> ClosureSurface
<<Feature>> CeilingSurface
<<Feature>> InteriorWallSurface
<<Feature>> FloorSurface
<<Feature>> OuterCeilingSurface
<<Feature>> OuterFloorSurface

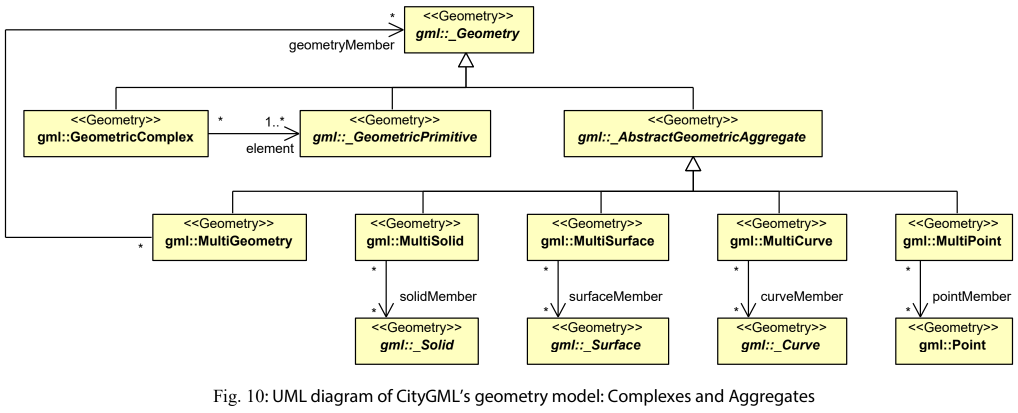
Figure 8.1: UML diagram of the CityGML building class (Gröger et al., 2012).

8.2 UML DIAGRAMS OF THE CITYGML GEOMETRY MODELS.



<<Geometry>> gml::_Geometry
<<Geometry>> gml::_GeometricPrimitive
<<Geometry>> gml::_Solid
<<Geometry>> gml::_Surface
<<Geometry>> gml::_Curve
<<Geometry>> gml::Point
+position : gml::DirectPosition [1]
solidMember
1..*
*
<<Geometry>> gml::CompositeSolid
<<Geometry>> gml::Solid
interior 0..*
exterior 1
0..1
0..1
surfaceMember 1..*
0..2 baseSurface
1..*
curveMember
*
<<Geometry>> gml::CompositeCurve
<<Geometry>> gml::LineString
+position : gml::DirectPosition [2..*]
<<Geometry>> gml::CompositeSurface
<<Geometry>> gml::Surface
<<Geometry>> gml::Polygon
<<Geometry>> gml::OrientableSurface
+orientation : gml::SignType [0..1]
1
1
1..* patches
exterior 1
interior *
<<Geometry>> gml::TriangulatedSurface
<<Geometry>> gml::_SurfacePatch
<<Geometry>> gml::_Ring
trianglePatches
<<Geometry>> gml::TIN
+stopLines : gml::LineStringSegment [0..*]
+breakLines : gml::LineStringSegment [0..*]
+maxLength : gml::LengthType [1]
+controlPoint : gml::posList [1]
<<Geometry>> gml::Triangle
<<Geometry>> gml::Rectangle
exterior 1
exterior 1
<<Geometry>> gml::LinearRing
+position : gml::DirectPosition [4..*]

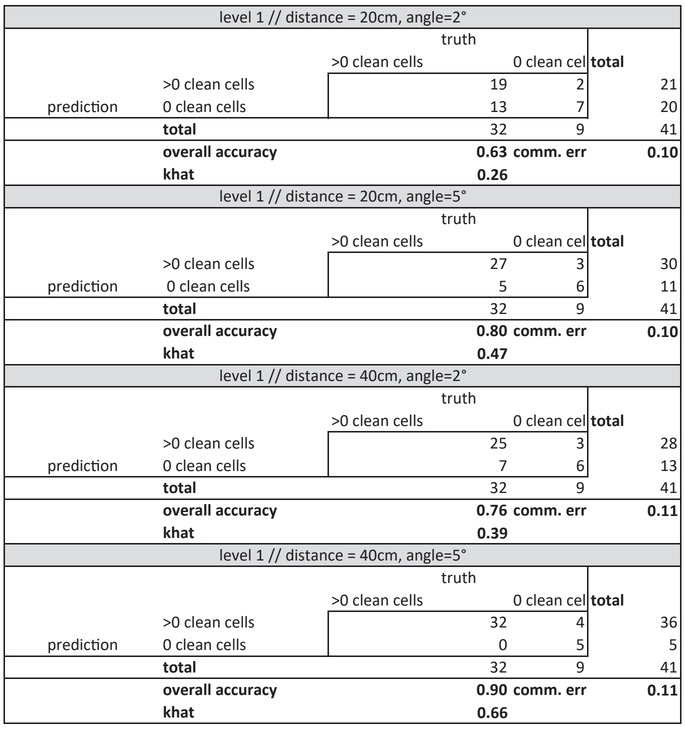
Figure 8.2: UML diagram of the CityGML geometry model (Gröger et al., 2012).



<<Geometry>> gml::_Geometry
geometryMember
*
<<Geometry>> gml::GeometricComplex
* 1..*
element
<<Geometry>> gml::_GeometricPrimitive
<<Geometry>> gml::_AbstractGeometricAggregate
<<Geometry>> gml::MultiGeometry
<<Geometry>> gml::MultiSolid
<<Geometry>> gml::MultiSurface
<<Geometry>> gml::MultiCurve
<<Geometry>> gml::MultiPoint
solidMember
surfaceMember
curveMember
pointMember
<<Geometry>> gml::_Solid
<<Geometry>> gml::_Surface
<<Geometry>> gml::_Curve
<<Geometry>> gml::Point
Fig. 10: UML diagram of CityGML's geometry model: Complexes and Aggregates

Figure 8.3: UML diagram of the aggregates of CityGML's geometry model (Gröger et al., 2012).

8.3 RESULTS OF LEVEL A VALIDATION



level 1 // distance = 20cm, angle=2°

		truth		
		>0 clean cells	0 clean cel	**total**
	>0 clean cells	19	2	21
prediction	0 clean cells	13	7	20
	total	32	9	41
	overall accuracy	**0.63**	**comm. err**	**0.10**
	khat	**0.26**		

level 1 // distance = 20cm, angle=5°

		truth		
		>0 clean cells	0 clean cel	**total**
	>0 clean cells	27	3	30
prediction	0 clean cells	5	6	11
	total	32	9	41
	overall accuracy	**0.80**	**comm. err**	**0.10**
	khat	**0.47**		

level 1 // distance = 40cm, angle=2°

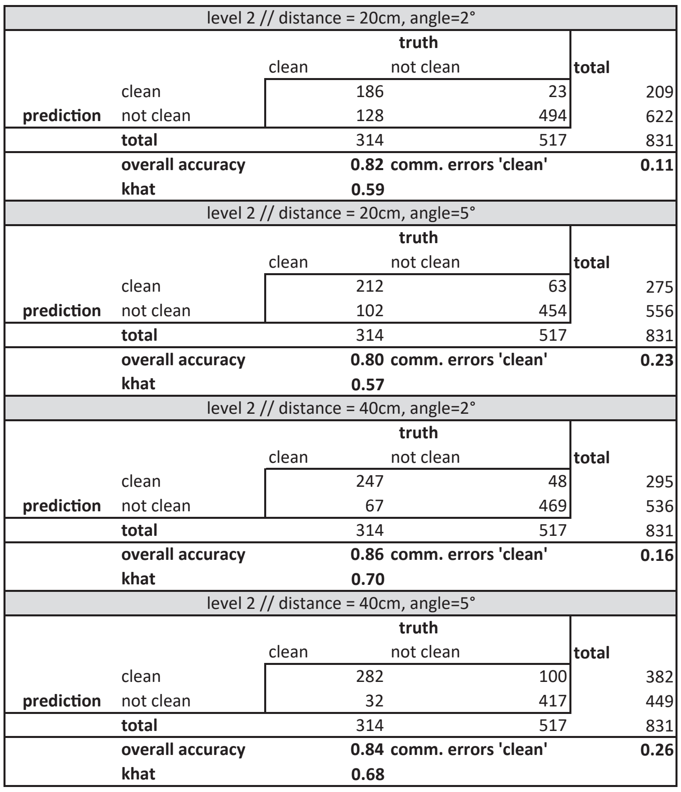
		truth		
		>0 clean cells	0 clean cel	**total**
	>0 clean cells	25	3	28
prediction	0 clean cells	7	6	13
	total	32	9	41
	overall accuracy	**0.76**	**comm. err**	**0.11**
	khat	**0.39**		

level 1 // distance = 40cm, angle=5°

		truth		
		>0 clean cells	0 clean cel	**total**
	>0 clean cells	32	4	36
prediction	0 clean cells	0	5	5
	total	32	9	41
	overall accuracy	**0.90**	**comm. err**	**0.11**
	khat	**0.66**		

Figure 8.4: Error matrices resulting from the level A validation

8.4 RESULTS OF LEVEL B VALIDATION



level 2 // distance = 20cm, angle=2°

		truth		
		clean	not clean	**total**
prediction	clean	186	23	209
	not clean	128	494	622
	total	314	517	831
	overall accuracy	**0.82**	**comm. errors 'clean'**	**0.11**
	khat	**0.59**		

level 2 // distance = 20cm, angle=5°

		truth		
		clean	not clean	**total**
prediction	clean	212	63	275
	not clean	102	454	556
	total	314	517	831
	overall accuracy	**0.80**	**comm. errors 'clean'**	**0.23**
	khat	**0.57**		

level 2 // distance = 40cm, angle=2°

		truth		
		clean	not clean	**total**
prediction	clean	247	48	295
	not clean	67	469	536
	total	314	517	831
	overall accuracy	**0.86**	**comm. errors 'clean'**	**0.16**
	khat	**0.70**		

level 2 // distance = 40cm, angle=5°

		truth		
		clean	not clean	**total**
prediction	clean	282	100	382
	not clean	32	417	449
	total	314	517	831
	overall accuracy	**0.84**	**comm. errors 'clean'**	**0.26**
	khat	**0.68**		

Figure 8.5: Error matrices resulting from the level B validation

8.5 RESULTS OF LEVEL C VALIDATION

[20cm, 2 degrees] cells 100% inside roof				
		truth		
		clean	not clean	**total**
prediction	clean	13	2	15
	not clean	36	78	114
	total	49	80	129
	overall acc	0.71	**comm. errors 'clean'**	0.13
	khat	0.28		

[90-100[% inside roof				
		truth		
		clean	not clean	**total**
prediction	clean	10	5	15
	not clean	7	84	91
	total	17	89	106
	overall acc	0.89	**comm. errors 'clea**	0.33
	khat	0.56		

[70-90[% inside roof				
		truth		
		clean	not clean	**total**
prediction	clean	7	4	11
	not cle	7	75	82
	total	14	79	93
	overall	0.88	**comm. errors 'clean'**	0.36
	khat	0.49		

[20cm, 5 degrees] cells 100% inside roof				
		truth		
		clean	not clean	**total**
	clean	19	3	22
prediction	not clean	30	77	107
	total	49	80	129
	overall acc	0.74	**comm. errors 'clean'**	0.14
	khat	0.39		

[90-100[% inside roof				
		truth		
		clean	not clean	**total**
	clean	11	10	21
prediction	not clean	6	79	85
	total	17	89	106
	overall acc	0.85	**comm. errors 'clea**	0.48
	khat	0.49		

[70-90[% inside roof				
		truth		
		clean	not clean	**total**
	clean	11	10	21
prediction	not cle	3	69	72
	total	14	79	93
	overall	0.86	**comm. errors 'clean'**	0.48
	khat	0.55		

[40cm, 2 degrees] cells 100% inside roof				
		truth		
		clean	not clean	**total**
	clean	24	3	27
prediction	not clean	25	77	102
	total	49	80	129
	overall acc	0.78	**comm. errors 'clean'**	0.11
	khat	0.50		

[90-100[% inside roof				
		truth		
		clean	not clean	**total**
	clean	15	16	31
prediction	not clean	2	73	75
	total	17	89	106
	overall acc	0.83	**comm. errors 'clea**	0.52
	khat	0.53		

[70-90[% inside roof				
		truth		
		clean	not clean	**total**
	clean	11	15	26
prediction	not cle	3	64	67
	total	14	79	93
	overall	0.81	**comm. errors 'clean'**	0.58
	khat	0.44		

[40cm, 5 degrees] cells 100% inside roof				
		truth		
		clean	not clean	**total**
	clean	41	6	47
prediction	not clean	8	74	82
	total	49	80	129
	overall acc	0.89	**comm. errors 'clean'**	0.13
	khat	0.77		

[90-100[% inside roof				
		truth		
		clean	not clean	**total**
	clean	16	19	35
prediction	not clean	1	70	71
	total	17	89	106
	overall acc	0.81	**comm. errors 'clea**	0.54
	khat	0.51		

[70-90[% inside roof				
		truth		
		clean	not clean	**total**
	clean	12	19	31
prediction	not cle	2	60	62
	total	14	79	93
	overall	0.77	**comm. errors 'clean'**	0.61
	khat	0.41		

Figure 8.6: Error matrices resulting from the level C validation

8.6 RESULTS OF LEVEL D VALIDATION (RATIONAL)

[20cm, 2 degrees] cells [70-100]% inside roof				
		truth		
		clean	not clean	**total**
prediction	clean	30	11	41
	not clean	50	237	287
	total	80	248	328
	overall acc	**0.81**	**comm. errors 'clean'**	**0.27**
	khat	**0.40**		

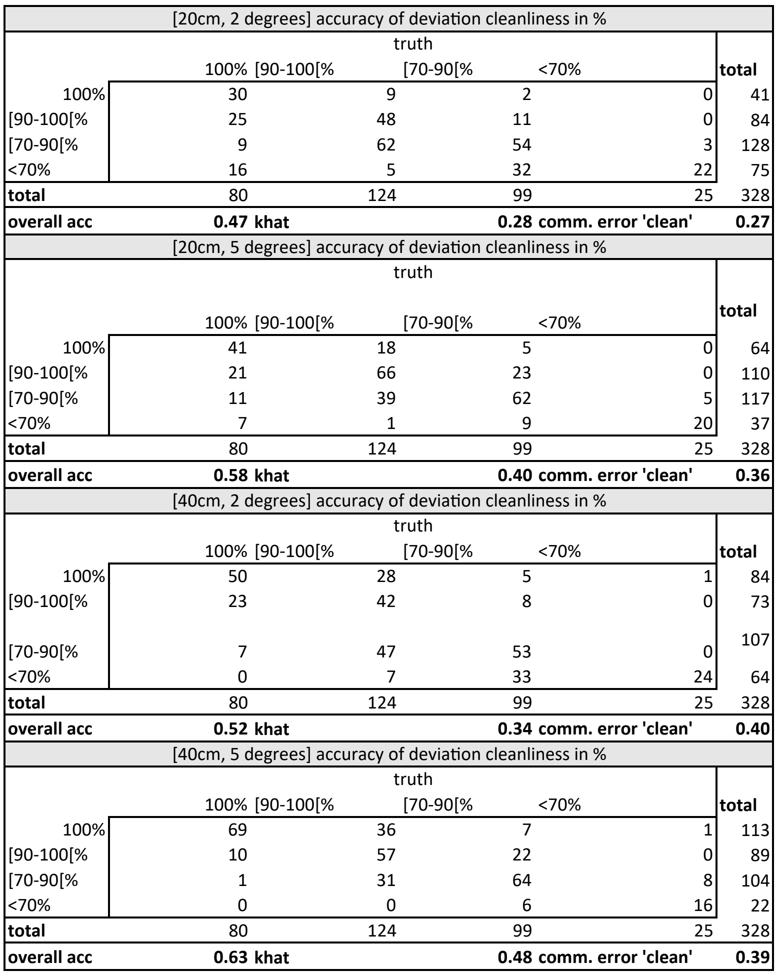
[20cm, 5 degrees] cells[70-100]% inside roof				
		truth		
		clean	not clean	**total**
	clean	41	23	64
prediction	not clean	39	225	264
	total	80	248	328
	overall acc	**0.81**	**comm. errors 'clean'**	**0.36**
	khat	**0.45**		

[40cm, 2 degrees] cells[70-100]% inside roof				
		truth		
		clean	not clean	**total**
	clean	50	34	84
prediction	not clean	30	214	244
	total	80	248	328
	overall acc	**0.80**	**comm. errors 'clean'**	**0.40**
	khat	**0.48**		

[40cm, 5 degrees] cells[70-100]% inside roof				
		truth		
		clean	not clean	**total**
	clean	69	44	113
prediction	not clean	11	204	215
	total	80	248	328
	overall acc	**0.83**	**comm. errors 'clean'**	**0.39**
	khat	**0.60**		

Figure 8.7: Error matrices resulting from the level D validation (nominal).

8.7 results of level d validation (nominal)



[20cm, 2 degrees] accuracy of deviation cleanliness in %

	truth				
	100%	[90-100[%	[70-90[%	<70%	**total**
100%	30	9	2	0	41
[90-100[%	25	48	11	0	84
[70-90[%	9	62	54	3	128
<70%	16	5	32	22	75
total	80	124	99	25	328
overall acc	**0.47**	**khat**	**0.28**	**comm. error 'clean'**	**0.27**

[20cm, 5 degrees] accuracy of deviation cleanliness in %

	truth				
	100%	[90-100[%	[70-90[%	<70%	**total**
100%	41	18	5	0	64
[90-100[%	21	66	23	0	110
[70-90[%	11	39	62	5	117
<70%	7	1	9	20	37
total	80	124	99	25	328
overall acc	**0.58**	**khat**	**0.40**	**comm. error 'clean'**	**0.36**

[40cm, 2 degrees] accuracy of deviation cleanliness in %

	truth				
	100%	[90-100[%	[70-90[%	<70%	**total**
100%	50	28	5	1	84
[90-100[%	23	42	8	0	73
[70-90[%	7	47	53	0	107
<70%	0	7	33	24	64
total	80	124	99	25	328
overall acc	**0.52**	**khat**	**0.34**	**comm. error 'clean'**	**0.40**

[40cm, 5 degrees] accuracy of deviation cleanliness in %

	truth				
	100%	[90-100[%	[70-90[%	<70%	**total**
100%	69	36	7	1	113
[90-100[%	10	57	22	0	89
[70-90[%	1	31	64	8	104
<70%	0	0	6	16	22
total	80	124	99	25	328
overall acc	**0.63**	**khat**	**0.48**	**comm. error 'clean'**	**0.39**

Figure 8.8: Error matrices resulting from the level D validation (rational).

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.