Aalto University
School of Engineering

Joonas Jokela

CityGML building model production from airborne laser scanning

Thesis submitted for examination for the degree of Master of Science in Technology

Espoo, 07.01.2016
Supervisor: Professor Henrik Haggrén
Advisors: Doctor of Science Petri Rönnholm,
Master of Science Arttu Julin

Aalto University
School of Engineering

Aalto University, P.O. BOX 11000, 00076 AALTO
www.aalto.fi

Abstract of master's thesis

Author Joonas Jokela

Title of thesis CityGML building model production from airborne laser scanning

Degree programme Degree programme in Geomatics

Major/minor Photogrammetry and remote sensing **Code** M3006

Thesis supervisor Professor Henrik Haggrén

Thesis advisor(s) Doctor of Science Petri Rönnholm, Master of Science Arttu Julin

Date 07.01.2016 **Number of pages** 64 **Language** English

3D city models have become an important tool in many applications across different fields. Usually these 3D city models only represent the geometrical attributes of the city, which enables easy visualization of cities. Yet, different thematic queries, analysis tasks, and spatial data mining are out of the reach of models that only offer us information about their geometry. CityGML 3D city models bring an addition of semantic information to the models.

In this thesis, the process and different techniques of building reconstruction from airborne laser scanning are explained. CityGML standard will also be explained and what has to be done in order to go from 3D building models to CityGML. The main focus of this thesis was to study how well it is possible to automatically create CityGML 2.0 3D city models from data collected only by airborne laser scanning.

CityGML has five different levels-of-detail indicating the level of precision of the building. LOD1 and LOD2 were the most important levels for this thesis, and so it was tested how well different software were able to export reconstructed building models in the CityGML format with these precision levels. These exports were checked against the official specification of CityGML to see how well they met the requirements. It was also explained what more would be needed for the process and data, in order to produce higher quality models in LOD3. Two different test areas were chosen with different building and roof types. One area included detached houses, some partly covered with vegetation, and another area included mainly apartment houses.

The thesis shows that as of now, it is still quite challenging to automatically produce city models that are in line with the CityGML 2.0 standard. The model driven methods had problems when it came to building installations, such as chimneys. These could not be modelled with software that used model driven methods. Data driven methods on the other hand had problems when it came to the conversion from the building models to the CityGML format. Terrain and terrain intersection curve also turned out to be more difficult to model than anticipated. Most of the software used in this thesis were not able to automatically handle the addition of these elements. The elements were possible to add later on to the CityGML file but only with use of additional software tools.

Keywords ALS, laser scanning, city models, CityGML, building reconstruction, modelling

Aalto-yliopisto
Insinööritieteiden
korkeakoulu

Aalto-yliopisto, PL 11000, 00076 AALTO
www.aalto.fi

Diplomityön tiivistelmä

Tekijä Joonas Jokela

Työn nimi CityGML rakennusmallien tuottaminen ilmalaserkeilauksesta

Koulutusohjelma Geomatiikka

Pää-/sivuaine Fotogrammetria ja kaukokartoitus **Koodi** M3006

Työn valvoja Professori Henrik Haggrén

Työn ohjaaja(t) Tekniikan tohtori Petri Rönnholm, Diplomi-insinööri Arttu Julin

Päivämäärä 07.01.2016 **Sivumäärä** 64 **Kieli** Englanti

3D kaupunkimalleista on tullut tärkeä työkalu eri alojen käyttämissä sovelluksissa. Yleensä näitä 3D kaupunkimalleja käytetään vain kaupunkien geometristen attribuuttien mallintamiseen visualisointitarkoituksiin. Kuitenkin erilaiset temaattiset kyselyt, analyysitehtävät ja spatiaalinen tiedonlouhinta ovat pelkästään geometriaa esittävien mallien ulottumattomissa. CityGML 3D kaupunkimallit ottavat huomioon lisäksi myös semanttisen tiedon.

Tässä työssä selitetään rakennusten rekonstruointiprosessi ilmalaserkeilauksesta sekä esitellään erilaisia rekonstruointitekniikoita. Myös CityGML standardi esitellään sekä se, mitä 3D rakennusmalleille pitää tehdä, jotta ne saataisiin CityGML muotoon. Tämän työn pääpiste oli, kuinka hyvin on mahdollista automaattisesti luoda CityGML 2.0 muotoisia 3D kaupunkimalleja pelkästään ilmalaserkeilaamalla kerätystä aineistosta.

CityGMLssä on viisi erilaista yksityiskohtatasoa, jotka kertovat, kuinka tarkasti rakennus on mallinnettu. Näistä LOD1 ja LOD2 olivat tämän työn kannalta oleellisimmat. Tämän vuoksi sitä, kuinka hyvin ohjelmista saadaan ulos rakennusmalleja CityGML muodossa näillä tarkkuusvaatimuksilla, testattiin. Saatuja tuloksia verrattiin virallisiin CityGML vaatimuksiin, jotta saatiin selville, kuinka hyvin vaatimukset täyttyivät. Myös se käytiin läpi, mitä muutoksia tarvittaisiin, jotta malleista saataisiin korkeamman, LOD3, tason malleja. Valittiin kaksi erilaista testialuetta, joilla oli erilaisia rakennus- ja kattotyyppejä. Toisella alueella oli omakotitaloja, joista jotkut olivat osittain kasvillisuuden peittämiä ja toisella oli pääsääntöisesti kerrostaloja.

Työstä käy ilmi, että vielä tällä hetkellä automaattinen CityGML 2.0 standardin mukaisten kaupunkimallien tuottaminen on haastavaa. Mallipohjaisilla menetelmillä oli vaikeuksia rakennusten pienien osien, kuten savupiippujen, suhteen. Näitä ei pystytty mallintamaan ohjelmilla, jotka pohjautuivat mallipohjaisiin menetelmiin. Toisaalta tietopohjaisilla menetelmillä oli ongelmia, kun ne muunnettiin CityGML formaattiin. Maanpinnan sekä maanpinnan ja rakennuksen leikkauksen mallintamisessa oli odotettua enemmän ongelmia. Useimmat tässä työssä käytetyt ohjelmat eivät pystyneet automaattisesti näitä mallintamaan. Kuitenkin, ne oli mahdollista lisätä jälkikäteen käyttämällä muita ohjelmia.

Avainsanat ALS, laserkeilaus, kaupunkimallit, CityGML, rakennusten rekonstruointi, mallinnus

Acknowledgements

I would like to thank FM-International Oy FINNMAP for the opportunity to write my thesis with them and giving me so much freedom with it. I would also like to thank my advisors of this thesis, Arttu Julin and Petri Rönnholm for all the useful comments during the process. Special thanks to Professor Henrik Haggrén for supervising the thesis.

Finally, I would like to thank my family and all my friends who have helped and supported me with this thesis and throughout my studies.

Espoo 07.01.2016

Joonas Jokela

Table of Contents

Abbreviations

ADE	Application domain extension
ALS	Airborne laser scanning
DGNSS	Differential global navigation satellite system
DSM	Digital surface model
DTM	Digital terrain model
GML	Geography markup language
GNSS	Global navigation satellite system
IMU	Inertial measurement system
INS	Inertial navigation system
LiDAR	Light detection and ranging
LOD	Level of detail
nDSM	Normalized digital surface model
OGC	Open Geospatial Consortium
PPP	Precise point positioning
PRF	Pulse repetition frequency
RANSAC	Random sample consensus
TIC	Terrain intersection curve
TIN	Triangulated irregular network
TOF	Time-of-flight
UML	Unified modeling language
VRS	Virtual reference station
WebGL	Web-based graphics library
XML	Extensive markup language

1 Introduction

1.1 Background

Laser scanning has become an important tool for the surveying of the world around us. As the technology has moved forward, laser scanning has become more and more available for different purposes. As laser scanning can be done from aerial, terrestrial, or even a land-based mobile platform, it can be applied to a variety of different kind of projects. In this thesis the focus will be on how the data collected with airborne laser scanning can be used to produce city models that are in line with the CityGML standard.

Airborne laser scanning has the benefits of being a relatively inexpensive way to produce 3D information of the surroundings from large areas in a short period of time. This allows us to produce dense point clouds of a city environment quickly and effectively. If these point clouds are dense enough, it is possible to create building models from them. These building models can then be used as a virtual 3D city model. Accuracy is an important aspect with 3D city models, as it opens up a way to reliably simulate scenarios in virtual reality. Airborne laser scanning suits to the task of producing data for 3D city models because of its high accuracy. (Vosselman & Maas 2010).

3D models have become an important tool in the modern day planning work. New applications and uses are discovered constantly for these models. 3D city models are no exception in this trend. Different types of uses for these models have been discovered in the last decade that were not possible before due to the lack of accurate city models.

Companies and cities can use these city models in order to visualize or to analyze data, giving them a lot of applications fields such as disaster management or urban planning. Before, most 3D models have had the problem that they are only made as purely graphical or geometrical models and because of this, lack semantic information. Therefore, these models could almost only be used for visualization purposes, but lack the ability to perform thematic queries, analysis tasks, and spatial data mining. (OGC 2012).

CityGML is XML-based format for the storage, representation, and exchange of virtual 3D models. CityGML aims to define basic entities, attributes, and relations of a 3D city model. CityGML can be used as a way to represent 3D city models with both geometric and semantic properties. It also allows the usage of said models over different applications. Standardization of these models is important since it would allow cost-efficient maintenance of 3D city models, leading to the possibility of using the same data in different application fields. (OGC 2012; CityGML).

In this work I will open up the workflow of making virtual 3D city models in CityGML format from airborne laser scanning data. I will also explain the basics of airborne laser scanning, and the process of extracting and reconstructing buildings from the laser point cloud data, as well as break down the concept of CityGML. Version 2.0 of CityGML is the most recent one as of now, and will be the focus of this study. In this study the word CityGML will refer to CityGML 2.0 if not mentioned otherwise.

It would be possible to use images in the process of building extraction and reconstruction alongside with laser data, but in this study the main focus will be just using laser point clouds in the process of reconstructing building models. Images could also be used for extracting

textures, but as it is not the main point of this work, models will be done without any additional texture.

1.2 Objectives of the study

The theoretical part of this thesis will open up the concepts of laser scanning, detection and reconstruction of buildings from point clouds, and the CityGML standard. The main focus will be in CityGML 2.0 as this is the most recent version of the standard, and thus, can be seen as the optimal result to which to aim when creating city models in CityGML format.

In the empirical part, the goal of the study is to see what the workflow from airborne laser scanning point cloud is all the way to a 3D city model structured based on CityGML standard. I will test the possibility of producing 3D building models in CityGML format as automatically as it is possible. The tests will be done with data collected in 2013 in Espoo region.

Two areas with different types of buildings were chosen; one area containing detached houses and another area containing apartment buildings. Different software was used in order to extract and reconstruct the buildings from the laser data. These reconstructed models were then exported into CityGML with semantic information added to them. These exports were then visualized and checked against the CityGML standard to analyze the successfulness of the models.

During the thesis I will search the answers to the following questions:

- Is it possible to produce models with ALS accurate enough to meet the requirements for CityGML LOD2?
- Can building installations, such as chimneys, be extracted from laser data automatically as CityGML format
- What would need to be done to produce LOD3 instead of LOD2?

1.3 Structure of the thesis

The thesis is as follows: The first chapter is the introduction to the topic as well as objectives of the study. The second chapter is about the principles of laser scanning, especially airborne laser scanning. In third chapter the reconstruction of buildings from laser data is presented and is basically divided into two between model driven and data driven methods. The fourth chapter is an introduction to the CityGML standard. The fifth chapter is the empirical part of the study where the used process to produce building models is explained and presented as well as the transformation into CityGML. Chapter six presents the results received from the study and finally, the seventh chapter is the summary and conclusion of the study.

2 Principles of airborne laser scanning

2.1 Introduction to laser scanning

To understand how laser scanning works, it is useful to know few properties about laser as well. Laser light is monochromatic, directional, and spatially coherent with a very narrow spectral width. These properties allow the laser beam to be focused on a single spot on the surface of an object (Baltsavias 1999c). Laser scanning allows the recording of hundreds of thousands of 3D points in a short span of time. These 3D points will form a point cloud where each point represents one distance measurement from scanner to the object. (Carter et al. 2012).

In order to work effectively, the laser scanning system needs a way to move the laser beam on the surface of the object. This is usually achieved through the use of mirror systems in the scanner. Different types of mirrors can be seen in figure 1. Different types of mirrors result in different kind of patters on the ground, which has to be taken into account in projects where constant point density is important. (Vosselman & Maas 2010).

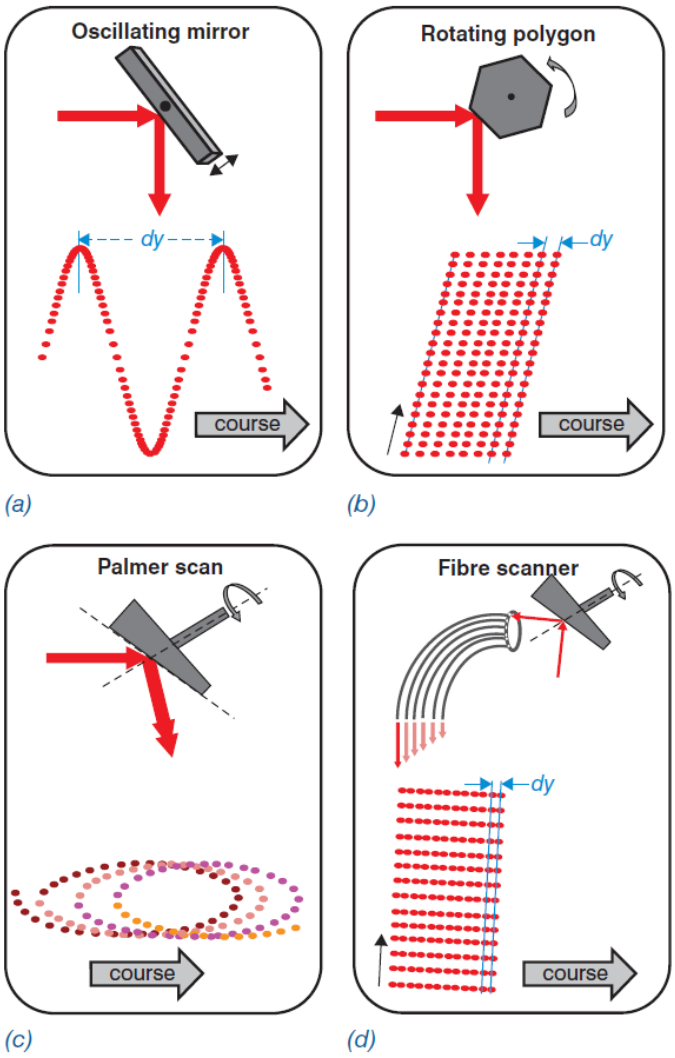

Figure 1. Examples of different mirror types and their patterns (Vosselman & Maas 2010).

In oscillating mirror systems, the moving mirror directs the laser beam across the swath creating a distinguished zigzag pattern on the surface. Point density varies on the scan line due to the mirror's accelerating and decelerating movement. (Vosselman & Maas 2010).

Oscillating mirrors have the advantage that the scan angle can be changed, meaning that along with scan rate, flying height, and laser pulse repetition frequency, the maximum point distance across- and along-track can be highly modified with changes to these variables. Because of their flexibility, oscillating mirrors are common in airborne laser scanners. Different mission requirements can be met by changing the configurations of the system. (Vosselman & Maas 2010).

Another type of mirror system uses a rotating polygonal mirror that generates points in only one scan directions. The advantages of this system are that the scan lines are parallel, and point density is more constant compared to the oscillating system. Palmer scanner system is constructed in a way that the mirror surface and rotation axis form an angle that does not equal to 90 degrees. The resulting pattern in airborne laser scanning is elliptical, and thus, might lead to some areas being hit multiple times by the laser pulse. This can help achieving denser point cloud than usual. (Vosselman & Maas 2010).

Glass fiber scanners work a bit differently than the previously explained scanners. In glass fiber scanner, a single laser pulse is emitted into neighboring glass fibers by a scanning mirror. A straight scan line can be achieved by arranging a number of glass fibers in a linear array. This system is extremely stable and scan angle can also be fixed. Point densities usually differ across and along the track, keeping in mind that the flying velocity affects these densities as well. (Vosselman & Maas 2010).

For the purpose of this thesis, laser scanning data is very suitable since its properties of fast data collecting, high density, high vertical accuracy, and low costs are advantageous when capturing 3D urban data (Wei 2008). Since this thesis focuses on the airborne laser scanning side of the modelling, in the rest of the thesis the term laser scanning is used for laser measurements done from the air.

2.2 Airborne laser scanning

Airborne laser scanning (ALS) is, as the name suggest, laser scanning that is performed from an aerial vehicle such as an airplane or a helicopter. It is used to capture the geometry and sometimes even information about the texture of a surface (Vosselman & Maas 2010). LiDAR is a term that is often used in the context of laser scanning and it is an acronym of the words light detection and ranging.

ALS is a fast and efficient technique for collecting 3D information from large areas in a relatively fast manner. As it is an active method, it can be used in basically any lighting conditions but ALS's maximum range performance is better when there is less background irradiance (Baltsavias 1999c). It is also quite flexible technique since the scanner can be attached to a fixed winged aircraft or helicopter depending on the project. Helicopters are typically used in smaller scale projects where maneuverability is an important factor of the project, for example when following power lines. Helicopters also allow scanning from lower altitudes than airplanes do. (Baltsavias 1999b).

Most laser scanning systems are capable of sending and detecting multiple echoes at the same time, which makes it possible to produce more accurate and denser point clouds. Usually the two first echoes contain about 90% of the total reflected beam. It can be assumed that the first echo belongs to the first object between the scanner and the terrain, so for example in a vegetated area it could be the canopy and presumably the last echo would be from the terrain. (Mallet & Bretar 2009). Some older scanners are restricted by their ability to send another pulse only after the previous one has returned. This limits the pulse repetition frequency (PRF) of the scanner leading to lower flying altitudes or less dense point clouds. (Pfeifer & Briese 2007).

Airborne laser scanning system usually consist of two parts; first part being the scanner system, which is used to measure the distance between a point on the ground and the scanner. The second part of the system is used to determine the exact position and orientation of the system. This is done with a combination of GNSS (global navigation satellite system) and IMU (inertia measurement unit). Figure 2 represents the fundamental principle of ALS. (Vosselman & Maas 2010).

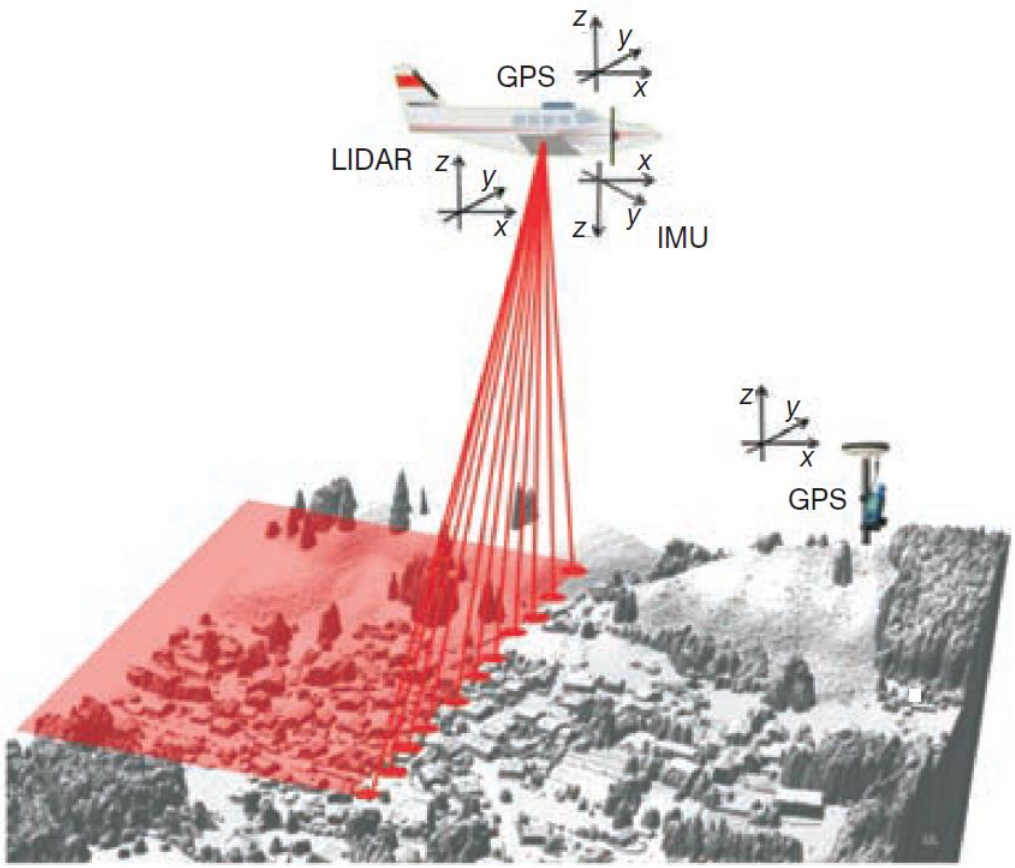

Figure 2. Basics of ALS (Vosselman & Maas 2010).

Important aspect that is always needed to take into account when doing airborne laser scanning is the pulse repetition frequency that directly correlates to the measured point density. The mean point density P_M on the ground can be calculated with the following equations. First we need to calculate the area covered by the aircraft in a certain time interval.

$$F = v * t * sw \quad (1)$$

In this equation v is the velocity of the aircraft, t is the time interval, and sw is the swath width of the scanner. We can also write the swath width as follows.

$$sw = 2h * \tan(\frac{\theta}{2}) \quad (2)$$

Where θ is the full scan angle of the scanner and h is the height above ground. Now it is possible to calculate the mean point density with the following equation.

$$P_M = \frac{PRF * t}{F} = \frac{PRF}{v * sw} = \frac{PRF}{2vh * \tan(\frac{\theta}{2})} \quad (3)$$

As can be seen from the equation 3, there are few variables that can affect the acquired point density. Highest point density is a combination of high pulse repetition frequency, low velocity, low flying altitude, and small scanning angle. Of course this equation is just a mathematical simplification of the scanning. In the actual scanning situation there are other factors to take into account, such as the scan pattern and surface topography. (Vosselman & Maas 2010). The point density should be adjusted according to the project specifications and the desired outcome. (Axelsson 1999).

Final accuracy of airborne laser scanning data is a sum of various aspects of the process. According to Vosselman and Maas (2010) the final accuracy is usually between 0.05m and 0.2m in height and 0.2m and 1.0m in position. As technology develops, these values will decrease. Obtained accuracy can worsen due to certain circumstances, such as errors in the GNSS and IMU calibration as well as the scanner assembly, errors caused by limited accuracy of the flight path restitution, or errors caused by the complexity of the object. Multipath reflections can also cause erroneous points in the data, lowering the overall accuracy.

2.3 Range measurements

Range measurements in airborne laser scanning are based on the fact that light travels with a known velocity. This constant can be used in order to calculate the distance between ground and the scanner. Measurements can also be done by studying the phase of the incoming continuous wave and calculating the travel time from it. (Vosselman & Maas 2010).

Backscattering properties of the surface has to be taken into account when measuring distances with laser scanner. The type of surface and the used wavelength of the laser greatly affect the number of observed echoes. Properties of the surface also affect the maximum range for a given wavelength. (Wehr & Lohr 1999).

The first technique utilized in laser scanning range measurements uses short pulses in repetitive manner to calculate the Time-of-flight (TOF). In it the velocity of light is considered to be a constant which can be used to calculate the time that it takes for a beam of light to travel to a surface and back. If we assume that ρ is the range to measure, then:

$$\rho = \frac{c}{n}\frac{\tau}{2} \quad (4)$$

Where c is the speed of light in a vacuum, τ is the time that it takes for the light to go from the scanner to the surface and back, and n is a correction factor which depends on the air temperature, pressure and humidity. (Vosselman & Maas 2010).

The time-of-flight can be detected in few different manners. One of these manners is peak detection where the detector uses a trigger pulse at the maximum point of the echo and then calculates the TOF from the time between maximums of the sent pulse and the echo. If there are more than one peak in the returning echo, then problems might arise in the calculations of the time delay. Another way to calculate the TOF is to use a trigger pulse at the moment when an echo reaches a preset fraction, which is usually 50%, of its maximum amplitude. This method is advantageous since it is relatively independent of an echo's amplitude. (Vosselman & Maas 2010; Mallet & Bretar 2009).

The second technique for range measurements can be done by comparing the emitted laser beam and observed backscattered beam. The phase difference ($\Delta\varphi$) between these two observations give us the time delay τ by

$$\tau = \frac{\Delta\varphi}{\pi} * \frac{\lambda_m}{c} \tag{5}$$

where λ_m is the wavelength of the amplitude modulation. With the solved τ, we can solve the equation 4 and get the value for the range. Due to range limitations of phase measurement techniques, it is usually not used in airborne laser scanning. (Vosselman, & Maas 2010).

2.4 Positioning

The laser scanner itself does not produce any position information about the echoes that it receives. They are simply range measurements and need extra information to determine the exact 3D coordinates for the point. GNSS and IMU are used in the airborne laser scanning process to measure the exact positions and orientations of the scanning system. These systems are integrated on the aircraft and the exact distances between them and the scanner must be known. (Vosselman & Maas 2010; Baltsavias 1999a).

Data from GNSS and IMU will be used to reconstruct the trajectories of the flight and later these trajectories can be linked to the range measurements from the scanner. Usually the GNSS has a sampling rate of 2 Hz and IMU 200 Hz. GNSS data is more accurate and is used for correcting the systematic drift effects of IMU and because of this it is important to not lose fix to satellites during the flight as IMU data may not be used over longer periods of time without the loss of accuracy. (Vosselman & Maas 2010; Carter et al. 2012). IMU data can be used to obtain navigation data in short time windows where a complete or partial GNSS outages happen (Rabbou & El-Rabbany 2014).

Usually measurements are done as differential GNSS (DGNSS) where ground stations are used in the calculations. DGNSS is used in order to compensate atmospheric disturbance and so to enhance the accuracy of the position information. In some cases, this is not possible

and the other option is to use precise point positioning (PPP). PPP uses precise clock and orbit information of satellites to calculate the position. The distance between the scanning system and the ground station should not be longer than 30 km but good accuracies can be achieved in longer distances as well (Vosselman & Maas 2010).

It is not always needed to set up or use actual ground stations. Virtual reference stations (VRS) can be used when there exists a GNSS reference station network that provides data to individual users. VRS consists of the reference station network, control center that gathers the data, and software that in real time corrects the GNSS measurements and signals. (Landau et al. 2002; Tötterström 2010). These virtual stations can be used in the GNSS calculations as the base stations.

In addition to the distance between the system and ground station, other factors can affect the accuracy of position measurements. Ground stations should be distributed well in the survey area, and there should be enough of them. The GNSS constellation also affects the accuracy of measured position (Baltsavias 1999a; Katzenbeisser 2003). In ideal case the available satellites would be scattered around the sky but in some cases it is possible that during the measurements they are grouped up in a small sector causing the constellation to worsen. This can be tried to solve by adjusting the elevation mask of the station.

In theory ALS can automatically produce correct X, Y, Z data, but this is not the case in practice. Due to errors in the data, there are always some shifts and tilts in the overlapping strips. These strips have to be adjusted to each other before the data can be used to its full potential. The process of adjusting the strips together is similar to the photogrammetric strip adjustment. Well defined tie points are searched between the strips, and these tie points are then used to solve the shifts and the tilts in the data. (Baltsavias 1999c).

2.5 Accuracy of measurements

Range measurement accuracy depends on several different factors that have to be taken into consideration when planning the scanning as errors might easily multiply the range errors to 3D coordinates. Pfeifer and Briese (2007) write in their article that using GNSS to calculate the flight path of the aircraft, can give precision of ± 10 cm in each coordinate direction and become a limiting component in the ALS precision process.

Baltsavias (1999a) lists in his article different sources for possible errors in the range measurements. In TOF techniques errors can be caused by the accuracy of detecting the same relative position on the transmitted and received pulse. Time differences between systems can also cause a minor loss of accuracy (e.g. between GNSS and the scanner).

In phase measurements the errors can be caused by the frequency of the tone or modulation, measuring the phase in the emitted and received pulse, noise, problems with the wave modulation oscillator, or turbulence and variations in the index of refraction. (Baltsavias 1999a).

Loss of accuracy can also be caused by effects from the mirror systems. These problems can be caused by reflection of light to the sensor from other source than the target, attenuation of flight, scattering of light because of dust etc., or even slowing of the light from passing through a window. (Baltsavias 1999a).

The reflective abilities of the surface can also affect the quality of the produced point cloud. For example, black surfaces, such as the roof of a building, can have a relatively weak reflection. Also shiny surfaces might cause problems with the reflected echoes as the reflection might be unidirectional giving no backscatter to the scanner. (Boehler et al. 2003).

3 Automated reconstruction of buildings from point clouds

3.1 Main principles of building extraction

In order to create a 3D city model, it is necessary to have the tools and skills for extracting individual buildings from the collected data. From the collected ALS data, it is possible to create a digital surface model (DSM) that has the information about buildings. These models though include all of the data as one and do not differentiate between individual buildings, neither do they make a difference between the terrain and the buildings. To obtain a 3D city model, buildings have to be detected and extracted from the terrain. After this has been done, it is possible to calculate geometrical values like volume or roof surface area from the model for each building (Dorninger & Pfeifer 2008; Haala & Brenner 1997).

In order to be able to model buildings from a point cloud, these buildings need to be extracted from their surroundings. This process can be divided into two steps. In the first step the buildings are detected from the data, and the approximate outlines for the buildings are determined. In the second step the buildings are reconstructed geometrically. This step will result in 3D polyhedral models of the buildings. (Rottensteiner & Briese 2002). Dorninger and Pfeifer (2008) on the other hand list the steps in the process of generating accurate building models from laser scanning data as: building detection, extraction of the outlines of the buildings, reconstruction of roof shapes, model generation and regularization, and model quality analysis. They also mention that reliability of the building models can be increased by set of rules. These rules can be, for example that the walls of a building are vertical, or that a roof is a composition of planar faces.

The first step in the process of classification of buildings is to separate ground surface from tall objects, such as buildings and trees. After this has been done, the task will be to separate those points that belong to buildings from points that do not belong to ground nor buildings (Tarsha-Kurdi et al. 2007b). Small areas classified as buildings can be eliminated in the post processing of classification (Matikainen et al. 2010).

Because of spatial distribution characteristics of laser point cloud, the roof of a building or the face of it, and ground usually have no distinct topographical changes nor height jump features. These conditions make it possible to treat those parts as smooth planes. (Zeng 2008). ALS point clouds have the problem that they might have a poor lateral accuracy making it hard to model the walls of a building (Brenner 2000).

Figure 3 is an example of a spatial distribution of laser cloud points. The slope between points p1 and p2 can be considered small and so they can be assumed to be a part of the same plane. Whereas the point p3 from the building's vertical wall has a much bigger slope to the point p1, meaning that they do not belong to the same plane. The same rule applies to the vegetation point p4.

Buildings can be acquired through a manual work of extracting and reconstructing buildings from image data. Although the models acquired by this method can have high quality, it is very slow method and prone to humanly errors. Because of these reasons it is important that this process could be made automatic. (Wang & Hsu 2007).

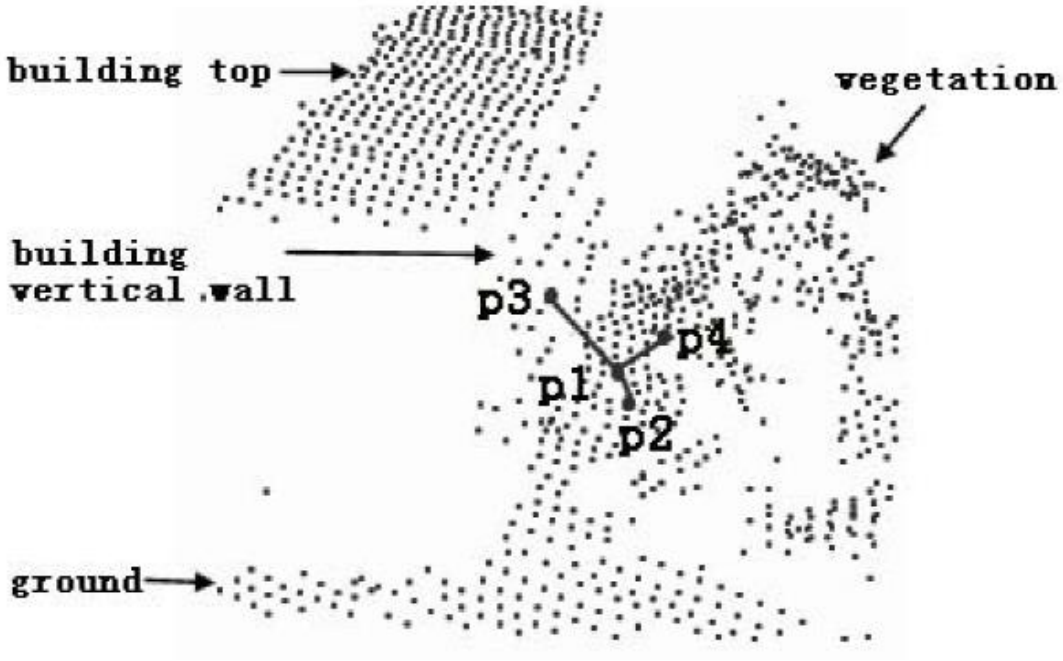

Figure 3. Example of spatial distribution of laser points (Zeng 2008).

The complexity of buildings can cause issues when trying to model them. In city areas, the buildings next to each other might have significant height differences. Although ALS data does not suffer from conventional shadows, it runs into problems with occlusion that can cause large number of data gaps in the point cloud. This problem is possible to fix by performing several flights over the problematic areas and collecting data from different angles. (Awrangjeb et al. 2014).

3.2 Automated building detection

The first step in the modelling process is to detect the buildings from the unorganized point cloud. Segmentation and clustering are ways to perform this action. In these, the points are organized into homogenous groups and these groups are then divided into classes (Pfeifer & Briese 2007).

The points in the laser point cloud can be classified into a variety of classes according to their properties. This can be done by the use of various macros or by using existing template models of the objects (Tomljenovic et al. 2015). Building detection is essentially a segmentation process where building point groups are detected and separated from the rest of the point cloud. (Wang & Shan 2009; Cheng et al. 2011).

In their study, Awrangjeb and Fraser (2014) divided the laser points to ground and non-ground points. They used the ground points to create a building mask where the black areas without points are the areas where buildings and trees are located. This way it is possible to find clusters of building and tree points from the data. These clusters of points can then be segmented in order to create planes. Planes from trees are usually smaller in size and randomly oriented and so, by studying these planes it is possible to separate trees from the

buildings. From the remaining planes, boundaries of individual buildings are possible to obtain.

Ground and non-ground points can also be separated by using morphological filters. Different size moving windows are used in this process, and for each window the lowest point is located and close by points, with higher value, are detected. These points get assigned a weight according to the probability of them being a ground point. After assigning weights to the points, thresholding can be used to distinguish between ground and non-ground points. (Morgan & Tempfli 2000).

Buildings and vegetation segments are possible to separate from each other by examining some of the properties of the segments. One of these properties is minimum building area, which means that segments smaller than the given value are considered vegetation and therefore eliminated. In the same way it is possible to use maximum building area as a constraint. Another property is how rapid the changes are in elevation. In vegetation areas changes should not be as extreme as the change from ground to the roof of a building. This can be used to eliminate vegetation clusters effectively from the data. (Morgan & Tempfli 2000).

In ALS data the buildings typically have two types of edges: jump edges and crease edges. Jump edges are identified as changes in the depth or height values in the data. These edges usually separate a building from another, or ground. Crease edges, on the other hand, are formed in places where two surfaces meet. These edges can be identified by a threshold in the angle between two surface normals at the intersection line. These edge types have different purposes in the segmentation process. Jump edges are usually used more when objects are being extracted or detected from the laser scanning data, and crease edges are used when objects are being reconstructed. (Wang & Shan 2009).

Basic idea of segmentation is that after it has been performed, all the points in a segment belong to a same shape (Haala & Kada 2010). In clustering the point groups are formed based on certain features of the points. These features can be for example a local roughness measure, or intensity measure of each point. Coordinates are then added as elements of the feature vector and groups are connected using them. (Pfeifer & Briese 2007).

Wang and Shan (2009) divide segmenting methods into two types: part-type segmentation and patch-type segmentation. In part-type segmentation the fundamental idea is to form visually meaningful simple objects by extracting geometric primitives such as planes, cylinder, and spheres. These simple objects can all be described with mathematical parameters. Hough transform, RANSAC (RANdom SAmple Consensus), and least square fitting are common techniques when working with part-type segmentation. Patch-type methods segment point clouds to homogenous regions as described earlier. In ALS data processing, patch-type methods are more commonly used.

Wang and Shan (2009) list some segmentation algorithms in their article. These algorithms can be divided into five groups.

1. Edge-detection method: In this method the laser data is converted into a range image, such as DSM, and edges are tried to extract from this. The problem with this is the loss of information when converting from 3D point cloud to 2.5D range image.

2. Surface-growing method: Seed points are identified from the data and then these seeds are gradually multiplied to the neighboring points based on the similarity measurements, such as proximity, and slope. It might be problematic to find good set of starting seeds and different seeds might lead to different segmentation results. Pfeifer and Briese (2007) mention that the similarity of a point can be measured either from the seed point to a new point, or from the previously accepted point to its new neighbors. Regions produced in this way are usually smooth or flat regions.
3. Scan-line methods: This method uses split-and-merge strategy. Scan lines are used from a range image and these lines are then segmented into line segments until perpendicular distance of points to their corresponding line segment is below a given threshold. Then similarity measures are utilized in order to merge these segments together. This method does not exist for point clouds that have not been structured.
4. Clustering Methods: As described earlier, these methods give each point a feature vector with some sort of geometric or radiometric measures. These feature vectors are then used in order to segment the point in feature space with clustering techniques, such as maximum likelihood.
5. Graph partitioning methods: Fundamental idea of these methods is that the points in the same segment are more closely connected to each other than to the points in other segments. Following this logic, the edges of the segments can be found in the areas where the connections between points are the weakest.

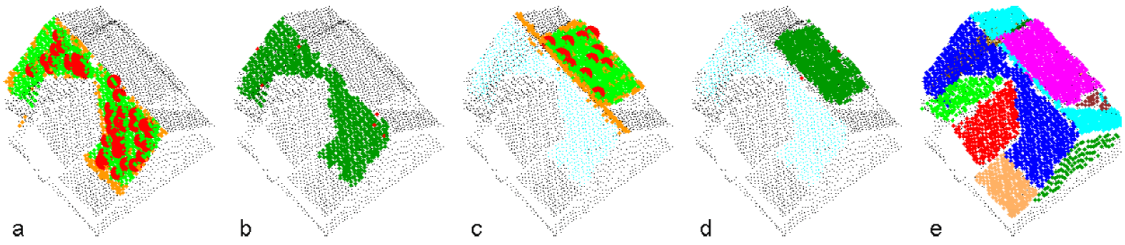

Figure 4. Example of the process of finding building segments with seed points (Dorninger & Pfeifer 2008).

In figure 4, there is an example of the steps that an algorithm, that can find segments of a building from starting seed points, goes through. In the pictures a and c, the red points represent the starting seed points, orange points are points accepted in object space, and green points are those accepted in feature space. Pictures b and d are the result of the algorithm where dark green points are accepted points, and red points are rejected points. Picture e represents the final segmentation of the building once the algorithm is done.

In their article Wang and Zhaoa (2008) used normalized digital surface model (nDSM) for the detection of buildings. Normalized digital surface model was created by subtracting digital terrain model (DTM) from DSM. From the created nDSM the heights of objects can be easily seen and with certain rules, objects that are not buildings can be eliminated. For example, by assuming that buildings have to be taller than 3.5 meters, objects that do not meet this requirement can be automatically removed from the data set.

After the points have been segmented, a neighborhood analysis can be performed for the segments to detect roof features like step edges or break lines. This information can then be used to construct the building. (Haala & Kada 2010).

3.3 Automated reconstruction of buildings

3.3.1 Model driven methods

Building reconstruction can be divided into two different categories: model driven methods and data driven methods (Dorninger & Pfeifer 2008; Tarsha-Kurdi et al. 2007c).

The idea behind model driven methods is to compare already existing models of roof forms to the points and calculating which one fits the best. This method is suitable for datasets with low point density. The model driven approach also ensures that the final roof shape is topologically correct one, problems might arise though when the roof shapes are complex ones with no equivalent in the model database. (Dorninger & Pfeifer 2008).

When buildings are of rather simple shape, they can be approximated by rectangular footprints and with standard roof shapes. This is usually the case in rural and suburban area but can also be applied in some degree to city areas (Haala & Kada 2010).

Hough transform is a popular method for the detection of objects with known mathematical expressions. Common practice when using Hough transform for the detection of roof planes is to use three parameters (θ, φ, ρ) to define the plane. These parameters represent the distance (ρ) of the plane to origin of the coordinate system, and the directional cosines of the plane (θ, φ). If these parameters reach a certain threshold, the observed plane is considered as a roof plane. To eliminate false positives from the found planes, additional constraint can be applied. (Sampath & Shan 2010).

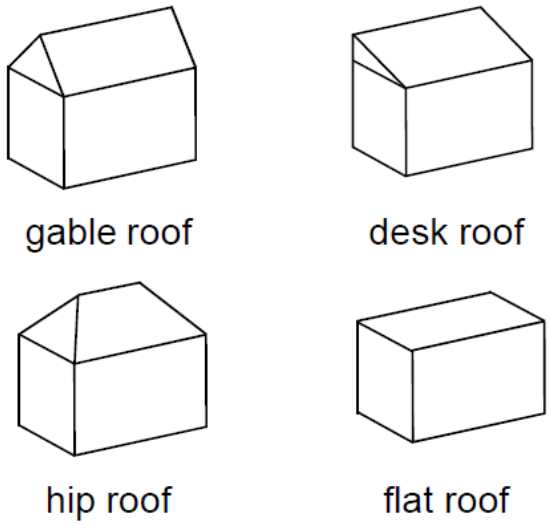

Figure 5. Examples of different roof styles (Brenner and Haala 1998).

Brenner and Haala (1998) used existing ground plans as a constraint for the search space for buildings. Depending on the plans and the desired model, it has to be taken into account that the plans might only represent the footprint of the building and not the whole roof area. In

their work they assumed that every building consisted of basic primitives and according to these primitives every building could be modelled. These primitives can be seen in figure 5.

Primitives are especially used when trying to model complex buildings. Complex buildings are first broken down into smaller parts of the buildings. Objective of this process is that these smaller parts can then be modelled with the use of the primitives. (Tarsha-Kurdi et al. 2007a). When the primitives are formed, their parameters are calculated to ensure that they fit the data optimally. The algorithms in model driven methods have to locate areas that will be compared against the models and also recognize which model to use in each case. The recognition step gives a number of approximate positions for each template and in the next step the parameters of the template are adjusted to the data. Number of the parameters to work with depend on the complexity of the primitives that are used and available transformations in the fitting process. (Schindler & Bauer 2003).

Tarsha-Kurdi et al. (2007a) state in their article that parameters used in model driven methods can be divided into two types: those defining the building ground or footprint, and those describing the building space. The first type of parameters define the building footprint position, orientation, and dimensions as well as the façades equations. The second type of parameters define the building roof plane equations.

Brenner et al. (2001) describe the workflow of modeling a simple L-shaped building in their article. First step would be for the algorithm to select two rectangles to cover the building area. Then hipped roofs would be selected on each of those rectangles and to estimate eaves and ridge heights. Last step would be to merge both of the primitives together.

Advantages of model driven methods are their ability to produce geometrically correct models without visual deformations. Their computing speed is also significantly faster than with data driven methods. The biggest disadvantage is the need of having an extensive library of models to which the point clouds are compared against. (Tarsha-Kurdi et al. 2007a).

3.3.2 Data driven methods

Data driven methods try to model a primitive or a complex building with the use of complex operations. They attempt to model buildings by using the point cloud as initial data and try to analyze it as a unity without relating it to any set of parameters. (Tarsha-Kurdi et al. 2007a).

Data driven methods reconstruct the roof from parts found by different segmentation algorithms. In ideal case these parts are the faces of the roof. The main problem in these methods is the ability to find the start and end points of the intersections between different segments. (Dorninger & Pfeifer 2008).

The workflow of data driven methods can be divided into following steps: acquisition, segmentation, building detection, feature extraction, and 3D reconstruction (Elberink 2008). Data driven approaches provide more universal models. In order to get these models, automatic segmentation of planes is highly important task. Most used methods to this task involve procedures as region growing, Hough transform, and RANSAC. (Karsli & Kahya 2008).

Segmentation methods can also be split into two classes based on the ways that they represent homogenous regions: edge/boundary-based, and surface-based methods. The basic idea of edge-based methods is to outline boundaries and detect edges, and according to these detections, segment the points inside the boundaries. Surface-based methods use local geometries as the indicator of similarity for the points. Points with spatially similar features and surface properties are merged together and boundaries are formed of these newly formed regions. (Wang & Shan 2009; Sampath & Shan 2010).

Edge-based methods run into problems when all edges do not form explicit segments due to outliers and incomplete edges in the data (Sampath & Shan 2010). When doing the segmentation part of data driven methods, the segmentation parameters should be chosen carefully as to avoid under- and over-segmentation. Over-segmentation means that an object feature is represented by more than one segment, and under-segmentation means that a segment is covering multiple object features. Of these two, under-segmentation is a bigger problem since it will be harder to reconstruct the object later on. (Elberink 2008).

One way to detect edges from the data is to use RANSAC approach. It randomly and iteratively uses a point in the data set to determine parameters for the model. The obtained parameters are then tested against the rest of the data set. The process is continued until the number of data points that fit the parameters is above a given threshold. After this the parameters are re-estimated with the new data set. This process has to be done iteratively because one round only gives one set of parameters. (Sampath & Shan 2010; Tarsha-Kurdi et al. 2007c).

Another technique for the detection of edges is called Hough transform. It is an algorithm that essentially works as a voting process, where each feature point votes for all the possible patterns passing through that point. The pattern that receives the most votes will be considered as the correct pattern. This method is robust to noise and discontinuities in the patterns. These patterns can be for example straight lines, circles, or ellipses. (Karsli & Kahya 2008).

In the case of point cloud pattern recognition, Hough transform has to be extended to 3D in order to detect 3D planes. Also has to be noted that the Hough transform does not check if the founded points on the same plane actually make up a continuous plane. This can be checked with a TIN of all the laser points. Only those points in the detected plane are used which form a connected piece of the TIN of a minimum size. After this points that are assigned to the found planar face are removed from the parameter space before starting to look for the next plane. (Vosselman & Dijkman 2001).

There are two approaches to modelling building façades. The building contour polygon is to be identified either before the segmentation of the roof planes, or after. In the first case, line generalization algorithms have to be used to allow simplifying or segmenting the building contour according to its façades. In the second case the contour polygon is segmented automatically according to the roof segmentation. The main difference between these two cases is that in the first one, one façade is presented by several vertical planes according to the number of its adjacent roof planes, and in the second the façade is presented by only one plane under the assumption that the façade was previously well filtered. (Tarsha-Kurdi et al. 2007a).

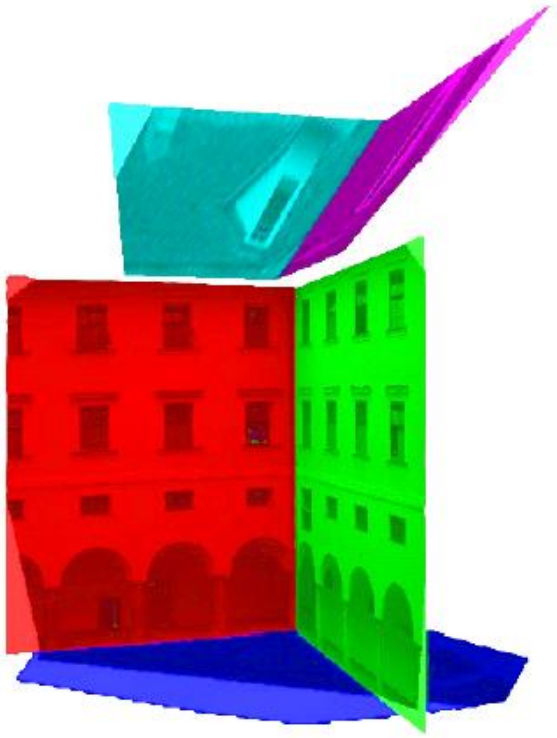

Figure 6. Example of a coarse model of a building (Schindler & Bauer 2003).

According to Schindler and Bauer (2003), the reconstruction process of a building can be divided into three parts. First one being the recovery of a coarse model, which consists of the main planes, this can be seen in figure 6. After this discontinuities, such as indentations and protrusions, are detected from the planes. And lastly the detected features are refined with the templates in the existing database.

Data driven methods can be seen to produce more accurate and robust models due to increasing point density of the laser scanners. This means that smaller roof details can be detected and different geometric constraints can be applied with more confidence. (Elberink 2008). Problems of data driven methods include the possibility of producing deformed models, and long computing times in some cases. (Tarsha-Kurdi et al. 2007a).

Tarsha-Kurdi et al. (2007a) list some reasons for possible deformations with the data driven methods.

- Usually modelling is based on the assumption that a building is composed of planes and lines. This is not usually true from a mathematical perspective, and thus, the equations for planes are merely approximations.
- Errors in the point clouds might cause problems. These errors can be caused by errors in position, or accuracy. Different artefacts, and multipath reflections are also possible sources of errors.
- The point cloud might not be evenly distributed causing irregularity in certain parts of the building.
- Point cloud density affects the risk of possible deformities in the modelling phase. The denser the cloud, the fewer the deformities.

- Interpolation of the point cloud can cause some effects to the models. It can allow the elimination of building façade points, in order to obtain regular point grid and to smooth out some of the errors. It can also cause unwanted effects if the sampling interval for the grid cells are very different to the density of points, or if there are empty areas in the point cloud.
- Noise can also cause some problems as well as some roof details, which make the segmentation harder.

4 3D city models and CityGML

4.1 3D city models

3D city models represent urban georeferenced spatial data and are a digital representation of the Earth's surface and objects belonging to the city area (Stadler & Kolbe 2007). These virtual city models usually consist of digital terrain models, building models, street-space models, and green-space models (Döllner et al. 2006). The models serve as platforms where 2D and 3D geodata, and georeferenced thematic data can be integrated. (Döllner & Buchholz 2005).

3D city models are an important tool in the analysis performed in many fields nowadays. Many tasks in urban planning require a model where buildings are separated from the terrain. These models can then be used to simulate different scenarios in a virtual environment. (Haala & Brenner 1997). Because of different simulation and analysis purposes, these models need to reflect the complexity of city objects and their interrelations (Stadler & Kolbe 2007). Semantical, topological, and geometrical correctness are important factors when considering the quality of 3D city models (Wagner et al. 2013).

In addition to the building models, these city models contain several other types of information such as information about terrain elevation, land use, vegetation, and roads. Main advance of models like these is that the information is stored in the same database which makes the use and interoperability of these models much easier. (Prieto et al. 2012).

In order to create 3D city models, it is important to create the geometry and appearance of city object with the complexity necessary to obtain realistic 3D models. But it would also be desirable to be able to add semantics to these models to make them understandable for computers. Semantics are possible to add to the models in a semi-automated way where elements of the 3D model get procedurally annotated. (Prieto et al. 2012).

With semantic and topological aspects added to the model, it would allow new kind of procedures to be performed with the models. Thematic queries, analysis tasks, and spatial data mining are examples of the new possibilities if semantics were to be added. (Döllner et al. 2006).

Kolbe (2009) mentions in his article that semantic models require extra work compared to normal models. Because of this, it should be noted that from economic point of view it only makes sense to build these models if the customers are able to get something new out of them or if the same data can be used by different customers within multiple applications.

4.2 CityGML

CityGML is an international standard by the Open Geospatial Consortium (OGC). It is an open data model and it is based on XML format. CityGML's data model is based on the ISO 19100 standards family and it is an application schema for Geography Markup Language (GML). CityGML is used to store and exchange 3D city models. (OGC 2012; Gröger & Plümer 2012; Kolbe 2009). Example of CityGML format can be seen in figure 7.

```
<?xml version="1.0" encoding="utf-8"?>
<CityModel xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns="http://www.opengis.net/citygml/2.0"
    xmlns:xAL="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0" xmlns:xlink="http://www.w3.org/1999/xlink"
    xmlns:gml="http://www.opengis.net/gml" xmlns:dem="http://www.opengis.net/citygml/relief/2.0"
    xmlns:bldg="http://www.opengis.net/citygml/building/2.0"
    xsi:schemaLocation="http://www.opengis.net/citygml/building/2.0 ../../CityGML/building.xsd http://www.opengis.net/citygml/relief/2.0 ../../CityGML/relief.xsd">
    <gml:name>Simple 3D city model LOD2 without Appearance</gml:name>
    <gml:boundedBy>
        <gml:Envelope srsDimension="3" srsName="urn:ogc:def:crs,crs:EPSG::25832,crs:EPSG::5783">
            <gml:lowerCorner>458868.0 5438343.0 112.0</gml:lowerCorner>
            <gml:upperCorner>458892.0 5438362.0 117.0</gml:upperCorner>
        </gml:Envelope>
    </gml:boundedBy>
    <cityObjectMember>
        <bldg:Building gml:id="GML_7b1a5a6f-ddad-4c3d-a507-3eb9ee0a8e68">
            <gml:name>Example Building LOD2 </gml:name>
            <bldg:function codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_function.xml">1000</bldg:function>
            <bldg:yearOfConstruction>1985</bldg:yearOfConstruction>
            <bldg:roofType codeSpace="http://www.sig3d.org/codelists/standard/building/2.0/_AbstractBuilding_roofType.xml">1030</bldg:roofType>
            <bldg:measuredHeight uom="#m">5.0</bldg:measuredHeight>
            <bldg:storeysAboveGround>1</bldg:storeysAboveGround>
            <bldg:storeyHeightsAboveGround uom="#m">3.0</bldg:storeyHeightsAboveGround>
            <bldg:lod2Solid>
                <gml:Solid>
                    <gml:exterior>
                        <gml:CompositeSurface>
                            <!-- Ground Slab -->
                            <gml:surfaceMember xlink:href="#GML_d3981803-d4b0-4b5b-969c-53f657594757"/>
                            <!-- Wall South -->
                            <gml:surfaceMember xlink:href="#GML_1d350a50-6acc-4d3c-8c28-326ca4305fd1"/>
                            <!-- Wall North -->
                            <gml:surfaceMember xlink:href="#GML_d3909000-2f18-4472-8886-1c127ea67df1"/>
                            <!-- Wall East -->
                            <gml:surfaceMember xlink:href="#GML_6286ffa9-3811-4796-a92f-3fd037c8e668"/>
                            <!-- Wall West -->
                            <gml:surfaceMember xlink:href="#GML_5cc4fd92-d5de-4dd8-971e-892c91da2d9f"/>
                            <!-- Roof North -->
                            <gml:surfaceMember xlink:href="#GML_ec6a8966-58d9-4894-8edd-9aceb91b923f"/>
                            <!-- Roof South -->
                            <gml:surfaceMember xlink:href="#GML_b41dc792-5da6-4cd9-8f85-247583f305e3"/>
                        </gml:CompositeSurface>
                    </gml:exterior>
                </gml:Solid>
            </bldg:lod2Solid>
```

Figure 7. Example of the CityGML syntax (Taken from CityGML example data set available on http://www.citygml.org/index.php?id=1539).

OGC (2012) defines the aim of CityGML to be a common definition for the basic entities, attributes, and relations of 3D city models. Liukkonen (2015) states in his study that different municipalities in Finland have come across the problem of the lack of a common standard regarding to the 3D city models. Yet a lot of cities are looking into building 3D city models for different applications.

CityGML differs from conventional ways of representing 3D city models with its semantic properties. These semantic properties allow users to perform actions which are not possible without the metadata that CityGML provides. In addition to the geographical properties of the city, CityGML offers the knowledge of semantic and thematic properties, taxonomies, and aggregations of the objects in the model. These additions allow users to perform new kind of simulations and analysis of the 3D city models (OGC 2012; Gröger & Plümer 2012; Kolbe 2009).

In semantic 3D city models, the objects are divided into multiple parts due to a logical criterion which reflect the structures given to the model or that can be observed in the real world. For example, a building can be divided into different parts due to their different roof types. (Kolbe 2009).

CityGML allows its features to have properties that are not spatial. For example, features can be assigned properties that are not visual, such as infrared radiation, noise pollution, or something else arbitrary, observable data. It is also possible for objects to be linked to corresponding objects in external datasets, such as a cadastral database. (OGC 2012)

Features are organized into smaller subsections called modules. These modules are, for example building module, or vegetation module. It is then possible to arbitrarily combine these modules as needed. These combinations are called profiles. (Gröger & Plümer 2012).

Modules in CityGML reflect the spatial, appearance, and theme characteristics of an object (Buyukaslih 2013).

How the geometrical features are presented in CityGML is based on GML 3.1.1. GML has a lot of interpolation options for surfaces and curves that CityGML lacks. In CityGML all coordinates of the outer boundary and those of the optional interior boundary (for example a window) has to be on the same plane, also only straight lines are allowed. These restrictions are made for compatibility reasons. Nonlinear structures are approximated by planar surfaces. (Gröger & Plümer 2012).

Well-known boundary representation is used to describe features geometrically. This means that volume features, like buildings, are represented as solids that are formed by bounding surfaces. These surfaces need to fit perfectly meaning that they do not leave gaps between them nor do they overlap or such. This kind of representation has the benefit that the volumes of objects or part of them are easy to calculate. For objects that cannot be represented as solids, surfaces or line geometries are used. (Gröger & Plümer 2012).

Figure 8 has an example of the representation of objects and how they are linked to each other in CityGML. Solids 1 and 2 share a surface su1 (blue). This means that the wall now consists of two parts: su1 and su2. The relations between these objects is represented as a relation diagram on the right.

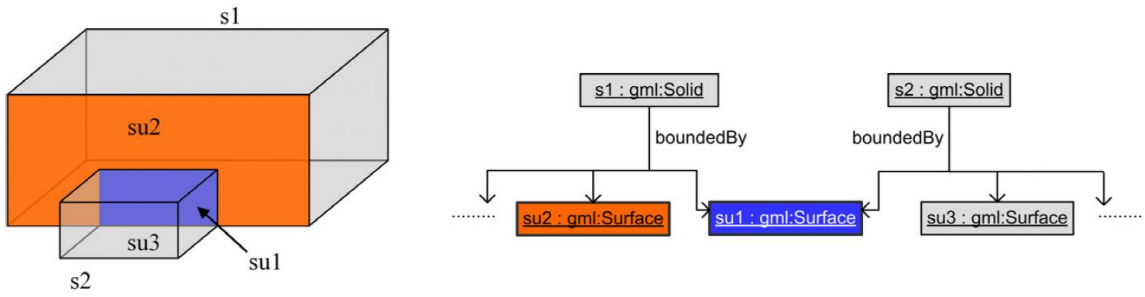

Figure 8. Example of representation of topology in CityGML (Gröger & Plümer 2012).

The most important types of objects that one could encounter in virtual 3D models have had their class definitions written into the CityGML data model. The CityGML data model is thematically divided into a core module and thematic extension modules. The basic concepts and components of the CityGML belong to the core module, and because of this they have to be implemented by any conformant system. Extensions are used to cover specific thematic fields of virtual 3D city models. In CityGML these thematic extension modules are: Appearance, Bridge, Building, CityFurniture, CityObjectGroup, Generics, LandUse, Relief, Transportation, Tunnel, Vegetation, WaterBody, and TexturedSurface. Modules and their schema dependencies can be seen in figure 9. (OGC 2012).

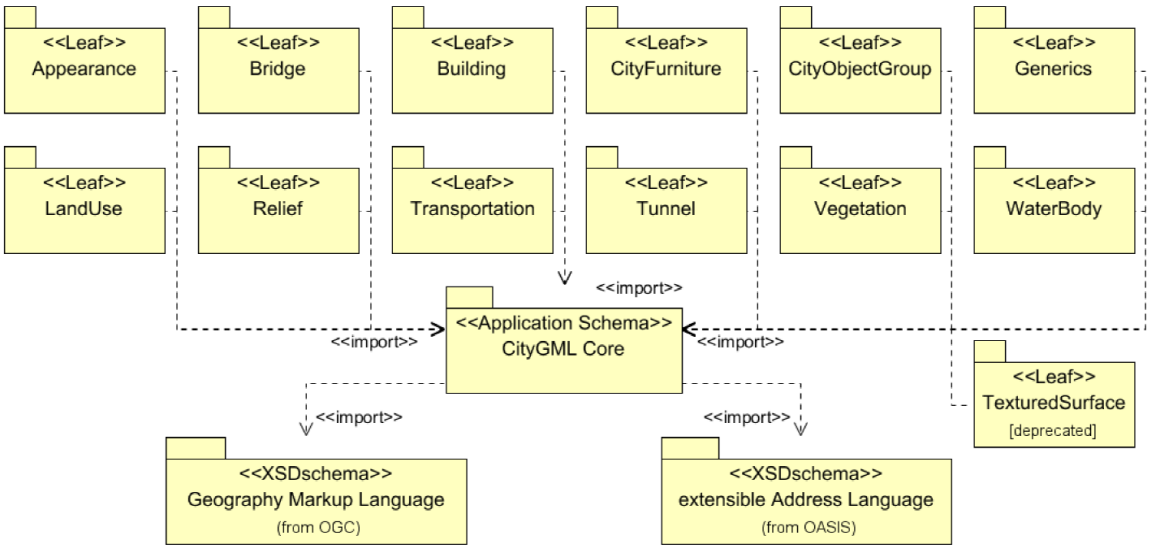

Figure 9. UML package diagram describing the modules of CityGML (OGC 2012).

Since actual cities are quite complex areas, it is impossible to specify classes for every detail in a city. In CityGML, objects that have yet to be modelled within the specification, can be represented with the use of generic objects and attributes. CityGML also allows new additions to the CityGML data model with the use of Application Domain Extension (ADE). The difference between ADE and generic objects and attributes is that additions done with ADE have to be defined separately to an extra XML schema definition file with each having their own namespace. This XML schema is used to import these additional CityGML modules. (OGC 2012; Mao 2011).

CityGML also has a possibility of representing objects that can be considered minor and of equal shape. These objects can be things like trees, or traffic lights. These shapes are represented as prototypes multiple times at different locations. In figure 10 there are some examples of the possible prototypes. Locations where these shapes are located have a reference to the prototype, a base point in the world coordinate system, and a transformation matrix. (OGC 2012).

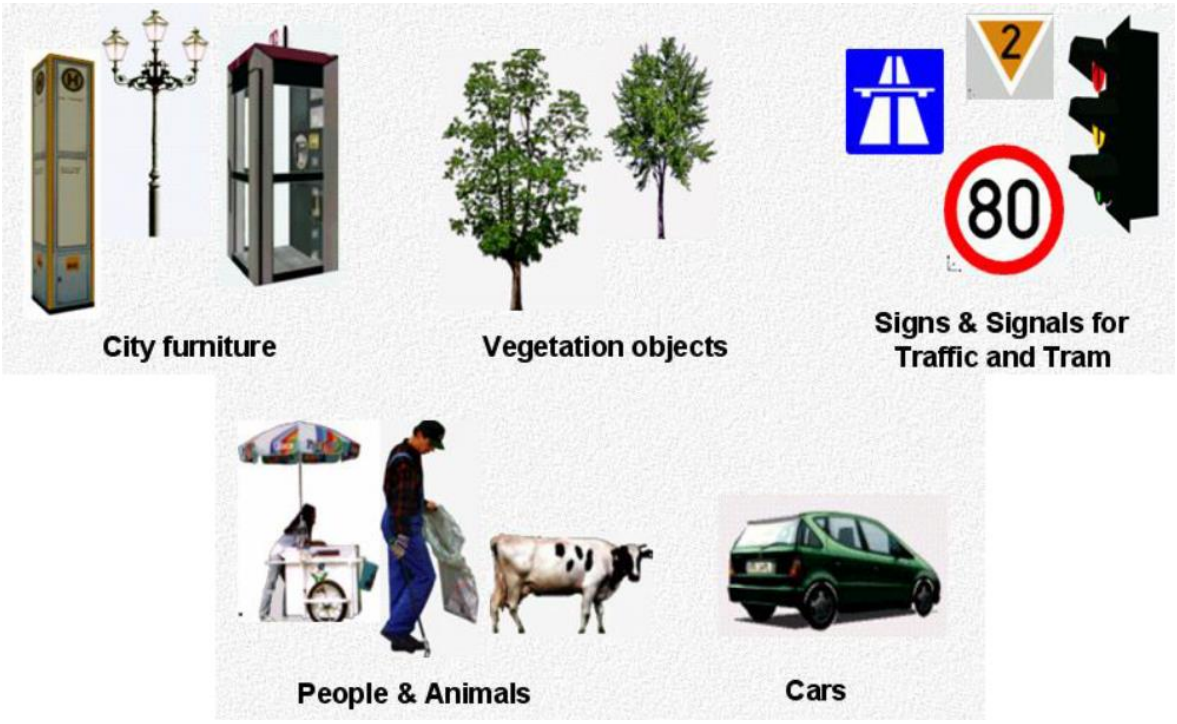

Figure 10. Examples of the possible shapes of prototypes (OGC 2012).

CityGML also has its downsides. CityGML can become large and difficult to handle, especially when modelling bigger cities. These models can become highly complex as well especially since the standard is constantly evolving and new additions can be brought into it easily. (Kolbe 2007).

4.3 Levels of detail in CityGML

Objects in the CityGML model can be represented on five different accuracy levels. These levels are called levels-of-detail (LOD). Higher levels are more accurate and have a higher structural complexity than the lower ones and therefore can be used in variety of different ways. CityGML allows the same object to be represented in different LODs at the same time, making it possible to run analysis on different degrees of resolutions (OGC 2012; Kolbe 2009). Example of a building modeled in LOD3 can be seen in figure 11.

Figure 11. Example of a building modeled in LOD3 (OGC 2012).

Gröger and Plümer (2012) list some characteristics about the LOD concept. They say that LODs make data integration and interoperability easier because features in same LODs can be integrated more easily than those of different LOD. Another point is that each LOD has its purpose, meaning that different degrees of accuracy can be used for different kind of applications. They also mention that since features can be in different LODs at the same time, it means that tools can be used to select dynamically the best LOD for the current task.

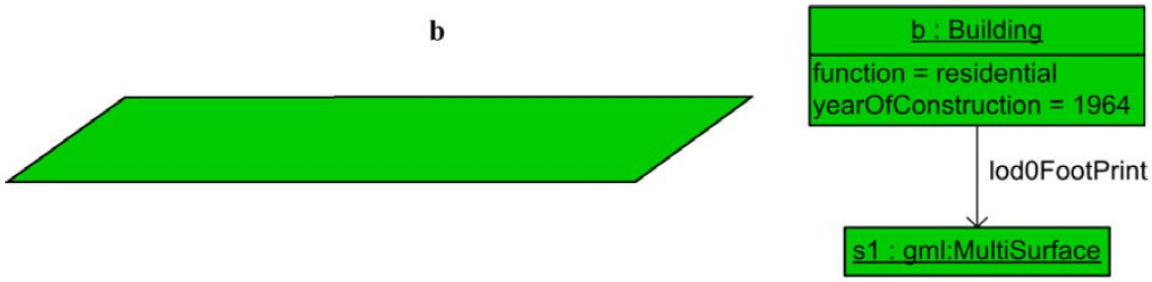

Figure 12. Example of a building in LOD0 (left) and CityGML feature structure as UML instance diagram (right) (Gröger & Plümer 2012).

The levels are labeled as LOD0 to LOD4, LOD0 being the coarsest level and LOD4 the level with highest degree of information. LOD0 can be defined as 2.5D DTM (Arefi et al. 2008). It is possible after this to place an aerial image or a map on the DTM. Using only DTM for the construction of LOD0 means that buildings in LOD0 are simply footprints or roof edge polygons as seen in figure 12.

LOD1 is a simple block model where the buildings and such are represented as blocks without any roof structures. It can be represented as a solid or as multi surface. Building can be divided into smaller parts called BuildingParts. This is beneficial in the sense that different parts of the buildings might have different attributes to themselves, such as varying height or year of construction. In the figure 13, the building consists of two parts with different kind of attribute values. The Terrain Intersection Curve (TIC) depicts the line where the building intersects with the terrain. This helps in the process of integrating the building to the terrain. (OGC 2012; Kolbe 2009; Gröger & Plümer 2012).

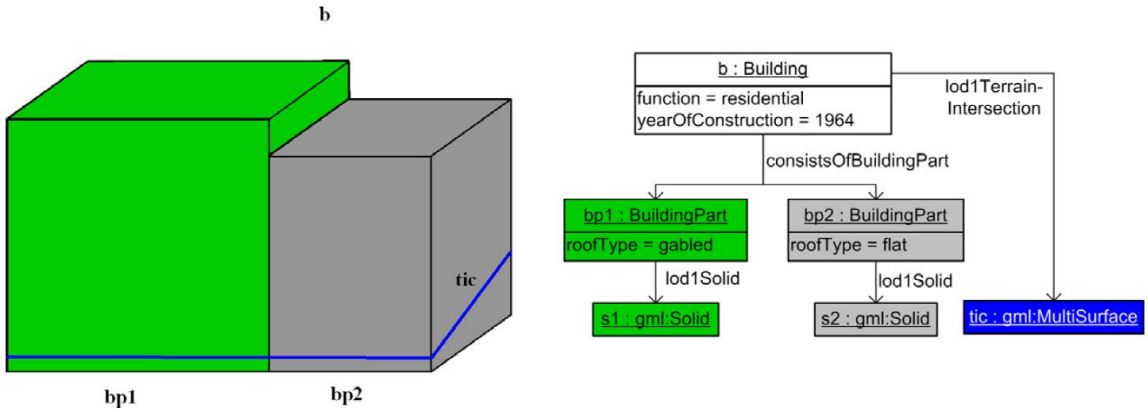

Figure 13. Example of objects in LOD1 (left) and CityGML structure as UML instance diagram (right) (Gröger & Plümer 2012).

LOD2 brings the addition of simple roof structure and it also has the information of larger structure parts of the building, such as balconies. In geometrical perspective, the biggest differences between LOD1 and LOD2 are that the outer walls and roof of a building can be represented with multiple faces, and the fact that curve geometries of the building can be represented in the model structure (Isikdag & Zlatanova 2009). Example of LOD2 can be seen in figure 14.

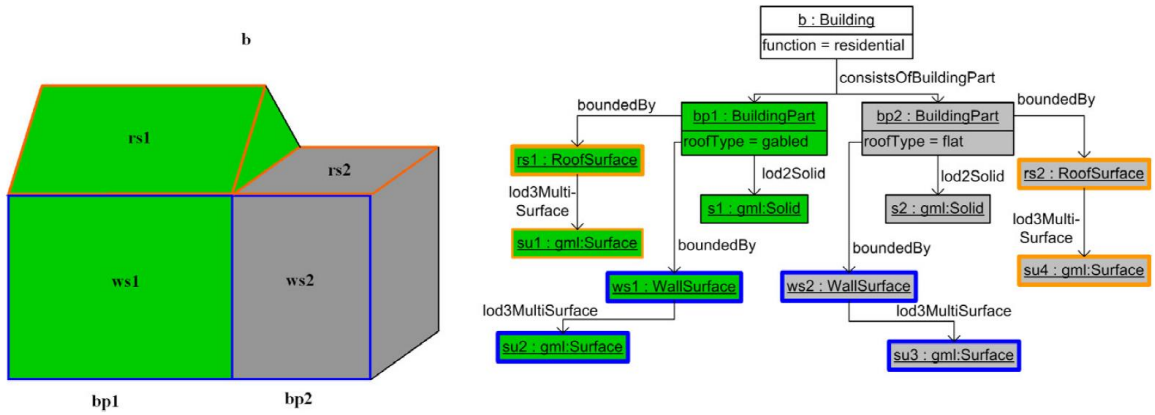

Figure 14. Example of objects in LOD2 (left) and CityGML structure as UML instance diagram (right) (Gröger & Plümer 2012).

LOD3 is an even more detailed level where the buildings have detailed wall and roof structures as well as doors, windows, and bays being represented. In figure 15 can be seen that LOD3 brings significant improvement to the accuracy of the building compared to the LOD2 in figure 14. The roof is more detailed and the walls now have windows and doors attached to the model. (OGC 2012; Kolbe 2009; Gröger & Plümer 2012).

The highest accuracy level, LOD4, adds information of interiors of the building. In LOD4 the rooms, stairs, and furniture are represented. Textures can be added and applied to all of the LODs for the structures. Because there are objects that are not enclosed completely, such as tunnels, it is needed to use special surfaces to close the elements. These surfaces have no counterpart in reality and are only taken into account, when needed to calculate volumes, and neglected, when they are not needed, like in visualization purposes. Examples of each LOD can be seen in figure 16 and figure 17. (OGC 2012; Kolbe 2009; Gröger & Plümer 2012).

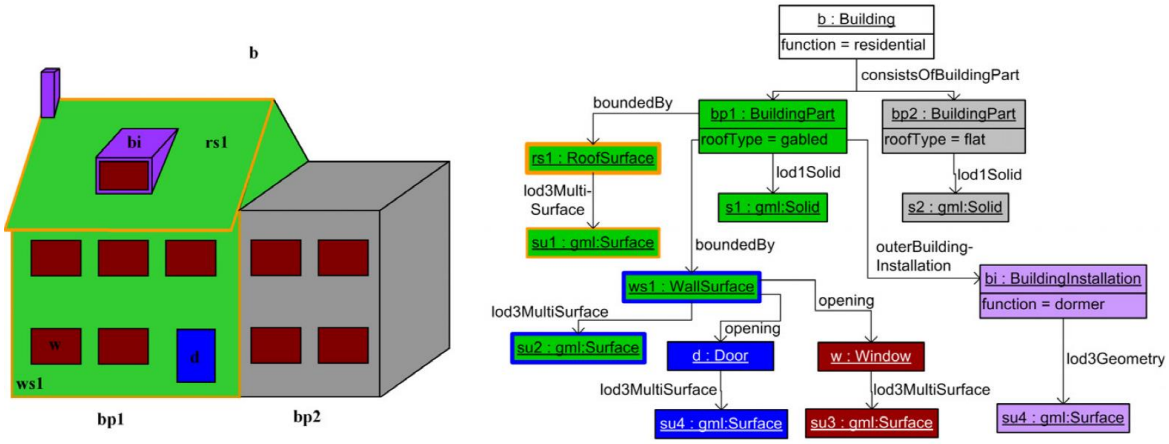

Figure 15. Example of objects in LOD3 (left) and CityGML structure as UML instance diagram (right) (Gröger & Plümer 2012).

Each of the levels described above have different characteristics to them that make them unique. As mentioned in OGC (2012), these characteristics are debatable and might change in the future. CityGML as of now does not support relative 3D point accuracy but it will be added in future. In table 1 different characteristics are shown for each LOD. For example, in LOD1 the positional and height accuracy of points should be 5m or less, and all objects with a footprint of at least 6m by 6m should be considered.

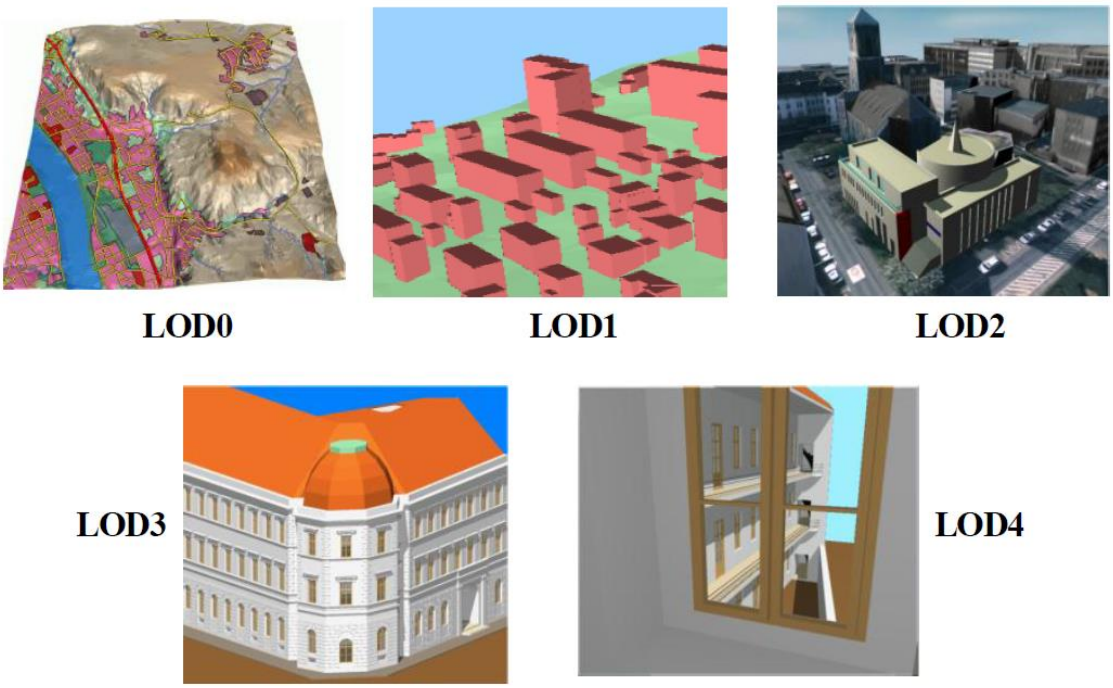

Figure 16. Different levels-of-detail in CityGML (OGC 2012).

Table 1. Accuracy requirements of different LODs (OGC 2012).

	LOD0	LOD1	LOD2	LOD3	LOD4
Model scale descriptor	regional, landscape	city, region	city, city districts, projects	city districts, architectural models (exterior), landmark	architectural models (interior), landmark
Class of accuracy	lowest	low	middle	high	very high
Absolute 3D point accuracy for position and height	lower than LOD1	5m	2m	0.5m	0.2m
Generalization	maximal generali-zation	object blocks as generalized features; >6*6m/3m	object as generalized features; >4*4m/2m	object as real features; >2*2m/1m	constructive elements and openings are represented
Building installations	no	no	yes	representative exterior features	real object form
Roof structure	yes	flat	differentiated roof structures	real object form	real object form
Roof overhanging parts	yes	no	yes, if known	yes	yes
City furniture	no	important objects	prototypes, generalized objects	real object form	real object form
Vegetation objects	no	important objects	prototypes, higher than 6m	prototypes, higher than 2m	prototypes, real object form
Vegetation areas	no	>50*50m	>5*5m	<LOD2	<LOD2

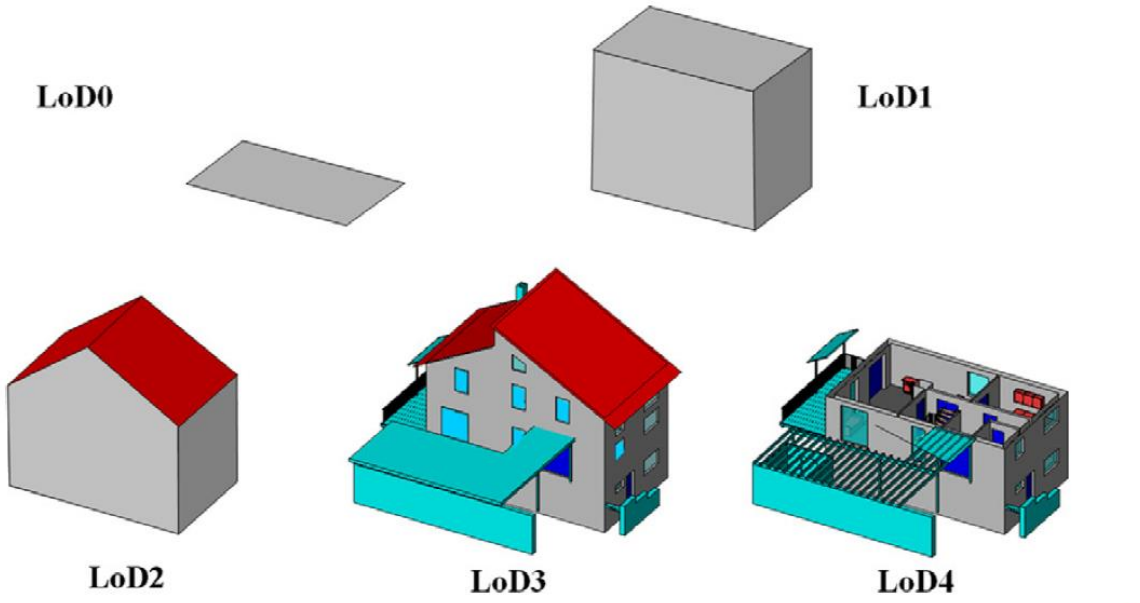

Figure 17. Representation of the five LODs (Gröger & Plümer 2012).

4.4 Terrain Intersection Curve

Terrain Intersection Curve is an important feature in CityGML as it identifies the exact line where the terrain intersects with the object. TIC helps solve the problems which arise from the use of different LODs or LODs from different providers. If the building and terrain do not meet perfectly then it is possible to integrate these two by "pulling up" or "pulling down" the terrain to fit the TIC. Use of TICs allow that the objects and the DTM can be maintained and provided separately. In figure 18 an example of the TIC line can be seen that denotes the exact position where the terrain touches the object. (OGC 2012; Gröger & Plümer 2012).

Figure 18. Example of the Terrain Intersection Curve represented as black line (OGC 2012).

In figure 19 there is an example of how the ground surface should be modelled according to the CityGML standard. On the left is presented a ground surface when cellar information is available, in the middle is the recommended way of representing the ground surface when cellar information is not available, and on the right is the not recommended way of generating the ground surface.

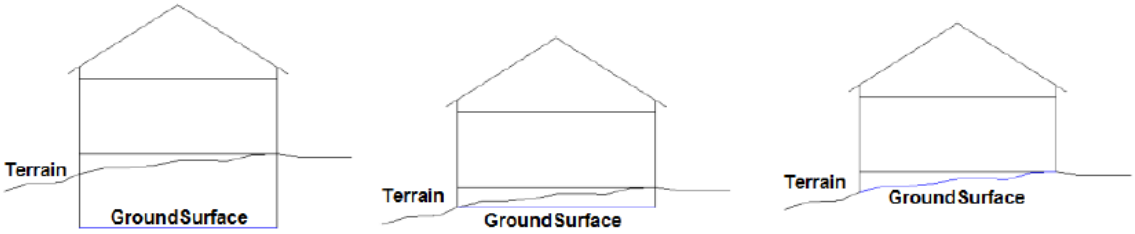

Figure 19. Examples of representations of ground surfaces (SIG 3D 2014).

4.5 Semantics in CityGML

Utilization of semantic information is what separates CityGML from typical 3D models. Semantic CityGML model uses the ISO 19100 standards family framework for the modelling of geographic features (Kolbe 2009). Geographic features are abstractions of real world objects (ISO 19109). They can be modelled by classes that have been formally specified by UML notations. (Kolbe 2009).

According to Fan et al. (2009) semantic information helps in the process of generalization of the models. When moving between different LODs, having semantic on the features is

beneficial since it can help avoiding deleting important features for visual impression, and also help when merging polygons which belong to different entities.

CityGML can be seen consisting of two different hierarchies: semantic and geometrical. In both of these the objects of the model are linked together by relationship attributes. In the semantic level the real world objects are represented by using features. For example, buildings, walls, and windows can be such features. The description of these features can be also made to include attributes, relations, and aggregation hierarchies between other features. This means that it is possible to reconstruct the part-of-relationship model without the need of geometrical information. However, with the addition of the geometrical information, geometry objects can be assigned to the features in order to represent their spatial location and extent. (OGC 2012).

The advantage of having two levels of hierarchy is that performing analyses, or answering queries can be performed in both hierarchies arbitrarily. If an object has information on both levels, it is required that the levels are coherent so that the levels match and fit together. For example, wall with two windows has to have information of these objects and relationships on the semantic as well as on the geometrical level (OGC 2012). Example of a fully coherent object can be seen in figure 20.

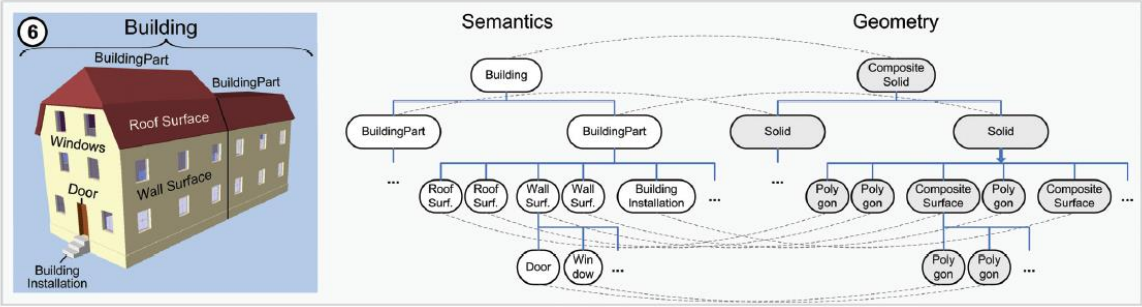

Figure 20. Modelled building that is coherent on semantical and geometrical levels. (Stadler & Kolbe 2007).

In CityGML most important geographic features in 3D city models like buildings, DTM, water bodies, vegetation, and city furniture have class definitions, normative regulations, and explanations of the semantics. Every object in the model can be linked to external databases. These references can link object to its equivalent in other database, such as in cadaster database. (Kolbe 2009).

Prieto et al. (2012) describe in their article that what is holding CityGML back the most is the difficulty of adding semantics to the 3D models. Another problem is how these models can effectively be visualized because CityGML is not the most efficient format for this task.

Semantic information can be added to the 3D model in a semi-automated way in which building parts, such as roofs and walls, are detected. Smaller elements like windows and doors can still be quite challenging task to detect completely automatically. (Prieto et al. 2012).

The way CityGML was designed, was that it would be a universal topographic information model. This model would define object types and attributes that are useful for a wide variety

of applications (OGC 2012). In table 2 are some attribute examples from Special Interest Group 3D (SIG 3D).

Table 2. Attributes of CityGML (SIG 3D 2014).

Name	Importance	Explanation
gml:id	mandatory	Identificator
gml:name	recommended if available	name of the building
bldg:class	no suggestion	unspecified classification of the building
bldg:function	recommended	functional significance of the building
bldg:usage	conditionally recommended	use of the building and respective percentage usage share of the total use
bldg:yearOfConstruction	recommended	year of completion of construction or alteration of the building
bldg:yearOfDemolition	no suggestion	year of deconstruction of the building
bldg:roofType	recommended	describes the characteristic shape of the roof
bldg:measuredHeight	recommended	difference between highest point of roof and the defined footprint of the building in meters
bldg:storeysAboveGround	recommended if available	number of above-ground floors
bldg:storeysBelowGround	recommended if available	number of underground floors
bldg:storeysHeightsAboveGround	no suggestion	height between two consecutive floors above ground
bldg:storeysHeightsBelowGround	no suggestion	height between two consecutive floors underground
bldg:lodXSolid	no suggestion	LODX geometry (volume) of the building
bldg:lodXMultiSurface	not recommended	LODX geometry (surface) of the building
bldg:lodXTerrainIntersection	no suggestion	LODX geometry (curve) of the TIC of the building
bldg:outerBuildingInstallation	no suggestion	relation to LOD2/3 building installation
bldg:boundedBy	no suggestion	relation to boundary surfaces
bldg:consistsOfBuildingPart	no suggestion	relation to LOD1-3 building parts
bldg:address	no suggestion	relation to one or more building addresses

4.6 Visualization of CityGML

There are few suggested ways for the visualization of the CityGML data. One way is to visualize CityGML files with existing CityGML viewers. The problem with this approach is that it is hard to visualize the 3D content of GML with these programs. Another way is to export the geometry of CityGML to a more efficient format. Some open formats that can interoperate with CityGML are KML and X3D. (Prieto et al. 2012).

X3D is an open standard and run-time architecture that is used to represent 3D objects based on XML syntax. It offers a system for storage, retrieval, and playback of real time graphics content embedded in applications. It is designed for the visualization of 3D scenes and objects over web. (X3D; Mao 2011).

X3D can be considered the best option for the conversion, since it has a geospatial component that can be utilized with the GIS component provided by CityGML. X3D also has the possibility to visualize the different LODs of CityGML. The downside is that some of the semantics of CityGML will be lost in the conversion. (Prieto et al. 2012).

In the recent years, visualization of 3D content over internet has been predominated by different plugins. The disadvantages of these are security and incompatibility issues. One way to visualize 3D content without the need of plugins, is to use WebGL (Prieto et al. 2012). WebGL (Web-based graphics library) is a plugin-free, cross-platform API for rendering 3D models within web browsers. (WebGL).

Callieri et al. (2015) mention that visualization of 3D models is a difficult task due to issues with handling complex data, and how to interact with them. They say that navigating 3D environments can be a challenging task because of the nature of the data as well as the inexperience of the users with these models.

The CityGML model can be stored into a 3D CityDB spatial database. This makes it possible to easily manage even larger 3D city models. Because CityGML format is not the most optimal for presenting and visualizing 3D geometry, the 3D geometric part of the data can be stored as different, more common 3D format. Semantic information of these models can then be queried with the specific CityGML ID of a feature from the database. (Prandi et al. 2015).

5 Case study: From point clouds to CityGML-based 3D city models

5.1 Material and software

Airborne laser scanning data used for this thesis was obtained in 2013 in the Espoo region by FM-International Oy FINNMAP. Two areas with different types of buildings were chosen as test areas to be used. The buildings in these areas can roughly be classified as detached houses, and apartment buildings. Sample of the test data can be seen in figure 21. Aerial photos of the test areas from Google Earth can be seen in figures 22 and 23.

The laser scanning was done with IGI LiteMapper 6800 system and Riegl ALS LMS-Q680i laser scanner. The laser scanning was done with the following parameters:

- Field of view: 60°
- Altitude: 500 m
- Ground speed: 80 knots
- Pulse repetition frequency: 400 kHz
- Side overlap: 30%
- Point density: 10 points/m^2

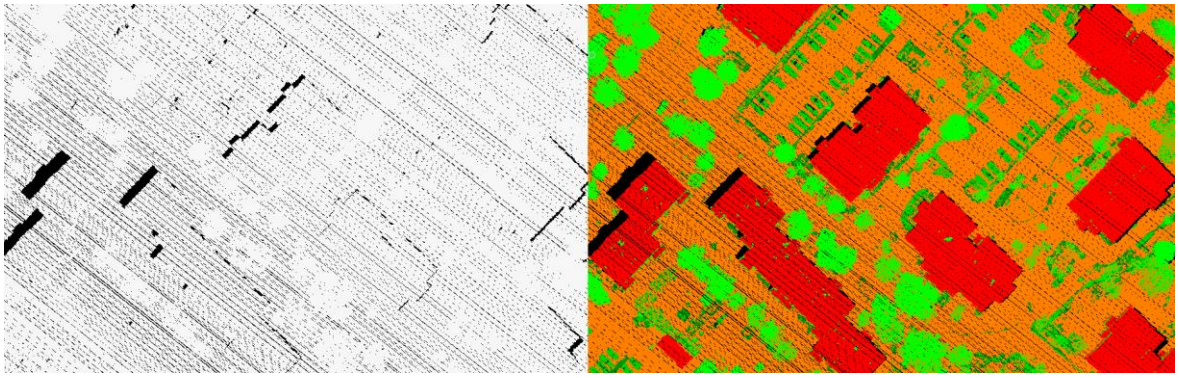

Figure 21. Sample of the laser scanning data used for the study. On the left is the raw point data without any classifications and on the right the same point data with classifications. Ground points are marked as orange, vegetation and other above ground points as green and dark green, and building roof points as red.

The two test areas were selected to represent different types of building and roof types to test the performance of several software in different situations. In the area of detached houses, some buildings were partly covered by vegetation as can be seen from the aerial photo in figure 22. This brought up another aspect for the modelling as to how they can perform in sub-optimal conditions.

The area with detached houses had 33 buildings and smaller sheds, and the area with apartment buildings had 36 buildings and smaller sheds. Most of these buildings also had building installations attached to them, such as chimneys or balconies.

Figure 22. Test area 1 with detached houses marked as red line (image taken from Google Earth).

Figure 23. Test area 2 with apartment buildings marked as red line (image taken from Google Earth).

The point cloud was processed with Terrasolid products (TerraScan, TerraMatch, TerraModeler, and TerraPhoto). They were also used in order to test the functionality of automatic vectoring of building seen in figure 24.

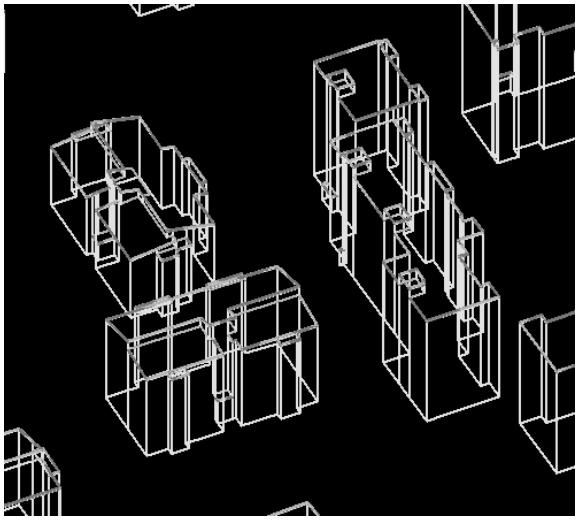

Figure 24. Examples of buildings modelled with Terrasolid products.

Another software used in this thesis was BuildingReconstruction 2015 from VirtualcitySYSTEMS. It can be used to automatically reconstruct buildings from point clouds. The buildings are constructed using DSM, DTM, and footprints of the buildings. The software uses model-based approach for the reconstruction process. The constructed models are watertight and can be edited manually with the tools provided. (VirtualcitySYSTEMS 2015).

BuildingReconstruction software also offers the possibility to export the created building models in CityGML format and to easily add semantic information, such as the year of construction to the CityGML model. BuildingReconstruction supports the export of buildings in LOD1 and LOD2 according to the OGC CityGML standards. Buildings are modelled to consist of wall, ground, and roof surfaces and each of these can be enriched with thematic attributes. (VirtualcitySYSTEMS 2015).

Lastly software from 3DCon GmbH was tested in order to see their capability of building reconstruction and export to CityGML. Tested software products were called Tridicon BuildingFinder, Tridicon CityModeller, and Tridicon Export.

Tridicon CityModeller can be used to reconstruct buildings from point clouds or nadir stereo aerial images with the help of building footprints (Tridicon CityModeller). Tridicon BuildingFinder can be used to create point clouds from aerial images and to detect buildings from this or from already existing point cloud. The main difference to CityModeller is that BuildingFinder does not require footprints of the buildings, and thus, can be used in an area where these are not available or easily obtainable. BuildingFinder reconstructs buildings by searching planar surfaces from the point clouds which it then allocates to building objects. (Tridicon BuildingFinder). Tridicon Export can be used to convert the files created with other Tridicon software into various other data formats. These formats are for example CityGML, COLLADA, KML, and shape.

5.2 Workflow

In order to start the modeling process, the obtained airborne laser data had to be processed. In the processing phase the point cloud was generated from the raw laser data and GNSS observations. This newly generated point cloud was then matched between individual flight lines to correct small mismatches. The whole workflow of the study can be seen from figure 25, where the steps of each software used are shown and classified in the main phases of the process of this work.

After generating the point cloud, different classes had to be classified from the point cloud. One of the most important classes regarding this thesis was the ground, as this would be used in the process of building reconstruction with all of the software.

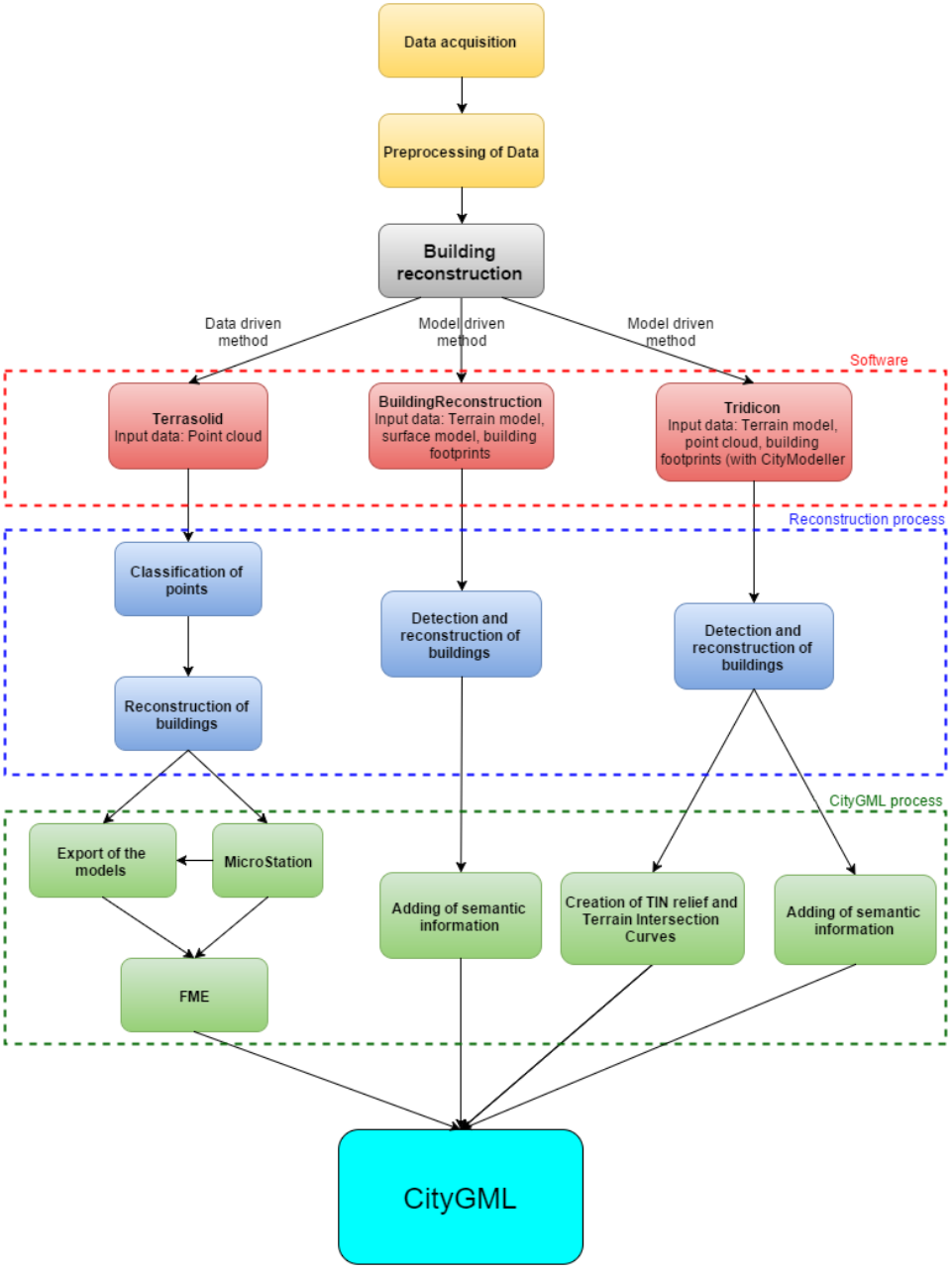

Figure 25. Workflow diagram of the study.

Ground classification was done with Terrasolid software called TerraScan. It uses an iterative method for creating a triangulated surface model. This method basically works as follows: The algorithm chooses some seed points from the data that can be can assumed to be points of the ground. From these seed points, a TIN model is created where most triangles can be seen to be under the actual ground level and only the vertices are touching the ground. From this point the algorithm iteratively adds more and more points to the model, and thus, improving the accuracy of the TIN model. (Terrasolid Oy 2015).

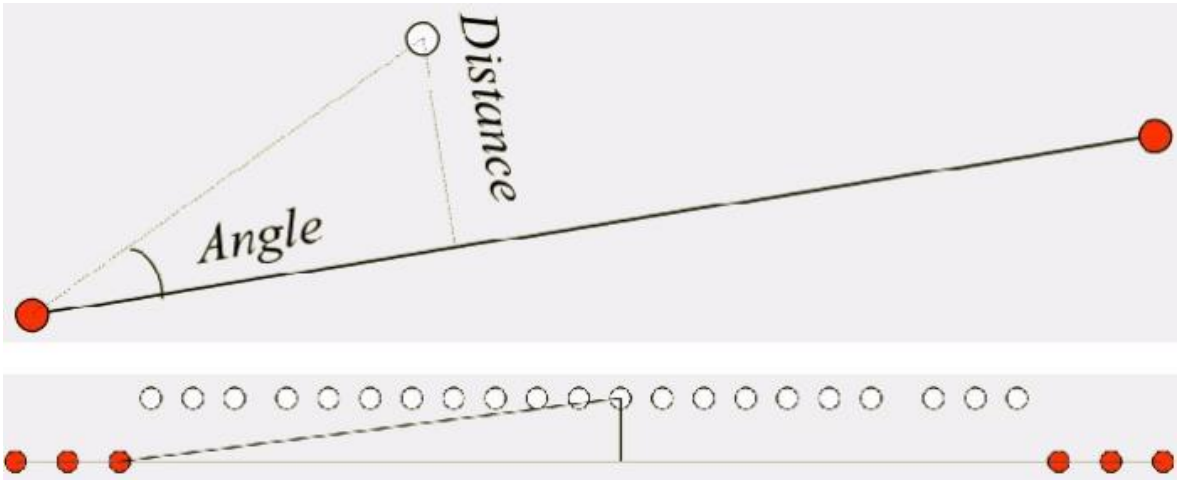

Figure 26. Explanation of the parameters that TerraScan uses for ground classification (Terrasolid Oy 2015).

The iteration process and the points that are accepted to the ground model can be controlled with a set of parameters in TerraScan. Iteration angle and iteration distance are the parameters that affect the model the most out of these parameters. Iteration angle affects the maximum angle between a point, its projection on the TIN model and the closest triangle vertex. This directly affects the amount of points that are accepted in to the ground class. Iteration distance is used in order to try to avoid that points from low vegetation or low building are classified as ground points. It checks that there are no big jumps between points in the classification phase, and discards points that are above a certain threshold. Illustration of these parameters can be seen in figure 26. (Terrasolid Oy 2015).

In addition to the ground, above ground points had to be classified to a separate class. This could be done with "height from ground" tool that uses the ground point class and compares the distance between a possible above ground point to the ground class and if the distance is above or between some thresholds, the point is classified as above ground point. (Terrasolid Oy 2015).

After these classes have been found, the points that belong to buildings can be classified. TerraScan offers "classify buildings" classification routine to automatically classify these points. This tool finds building roofs based on the planarity of these surfaces. The algorithm focuses on holes in the ground surface and tries to find planar surfaces above these holes. (Terrasolid Oy 2015).

Automatic building reconstruction could be done with "vectorize buildings" tool in TerraScan. For this it was necessary to have building, ground, and above ground classes

classified. It is also possible to use building footprint polygons in the process of reconstructing the buildings but they were not used in this test case. This tool creates MicroStation cell elements that contain shapes for each of the roof and wall planes. TerraScan uses data driven methods in the building reconstruction phase. (Terrasolid Oy 2015). MicroStation is a 3D CAD software developed by Bentley Systems.

These 3D building models were then exported from TerraPhoto as COLLADA as well as from MicroStation as COLLADA and KML. These models were then taken to FME (Feature Manipulation Engine) and converted to CityGML format. FME is a software for data conversion and integration between different formats developed by Safe Software Incorporation.

In addition to the buildings vectorization with TerraScan, BuildingReconstruction 2015 was used. For input this software needs digital surface model and digital terrain model as ASCII grid or XYZ point cloud. TerraScan was used to convert LAS 1.2 point clouds into XYZ format. Digital surface model was also created with TerraScan by using a tool called "classify by echo" which classifies the point cloud using echo information. DSM consists of a combination of points that were classified as only echoes or echoes that were first echoes of many from a pulse. Digital terrain model was the same ground class as the one used with the Terrasolid software. For the reconstruction process with BuildingReconstruction 2015, 2D building footprints were also needed. These were obtained from Helsinki Region Infoshare, which is a service for distribution of open data in the Helsinki region. This software uses model driven methods. (VirtualcitySYSTEMS 2015; Terrasolid Oy 2015).

BuildingReconstruction 2015 had the option to export the created building models directly in CityGML format with options to add semantic information to the building models. This export was used in order to export LOD1 and LOD2 models of the buildings.

Building reconstruction was also done with 3DCon software Tridicon CityModeller and Tridicon BuildingFinder. For Tridicon CityModeller point clouds of the two test areas were used that contained all the points as well as a cloud that only contained the ground points. Building footprints used were the same as in the BuildingReconstruction 2015 case. Tridicon BuildingFinder was tested with the same data excluding the building footprints.

Building models created with these two software products were then taken into Tridicon Export where they were converted into CityGML. This software allowed terrain data to be attached to the CityGML, so terrain intersection curves could be added into it as well as the TIN model of the area. LOD1 could be attached to the same model as LOD2 or written as a separate CityGML file. Tridicon Export had a lot of different options for the creation of the CityGML file and the dialog can be seen in figure 27.

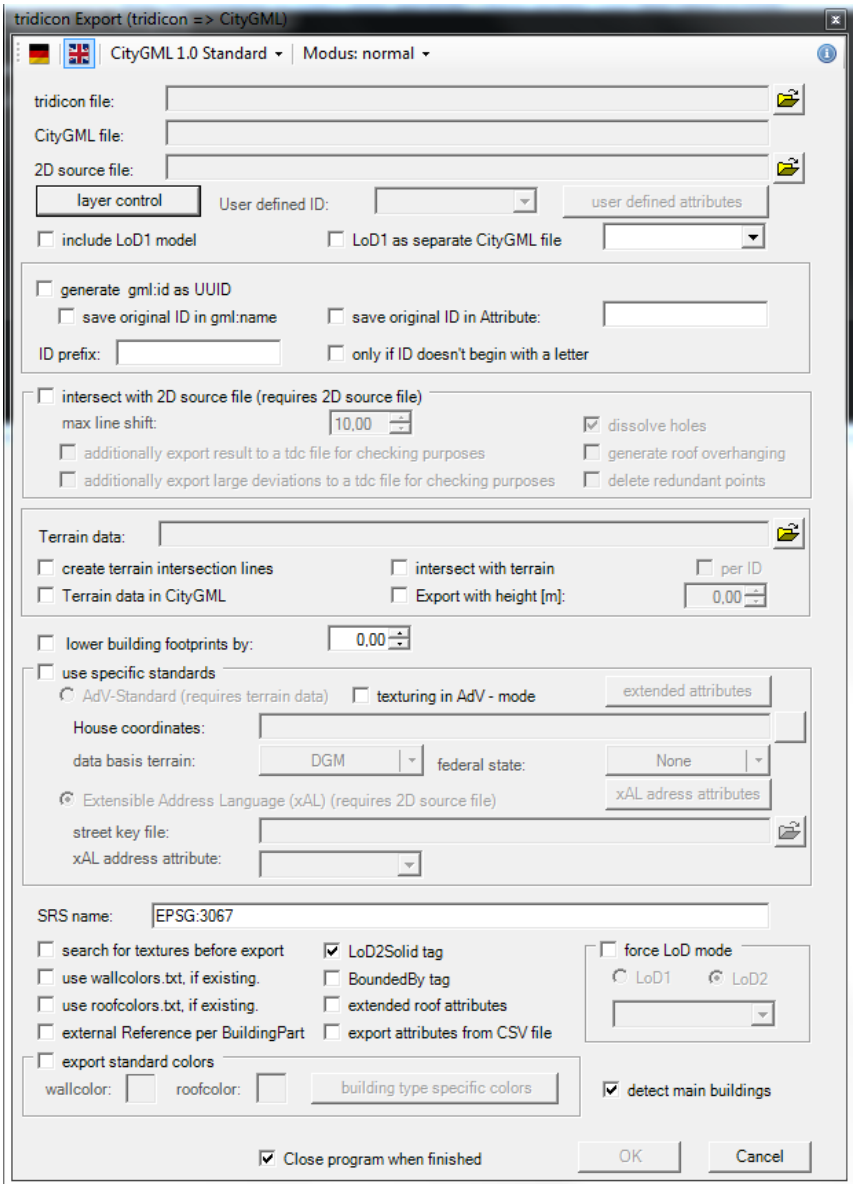

Figure 27. Tridicon Export dialog for CityGML export.

6 Results

6.1 Automatic building reconstruction

The automatic building reconstruction process worked overall well. Data driven and model driven methods produced different results. Software that use model driven methods had more problems with complex roof structures than data driven methods. Data driven methods also were better at modelling smaller building installation parts of the buildings than model driven methods. Both methods could detect all the main buildings in the test areas, only few smaller sheds covered by vegetation could not be modelled or were modelled incorrectly. Next the individual software processes will be explained with more details.

Reconstruction of buildings worked well with Terrasolid software. As it used data driven methods to reconstruct the buildings, it did not have such big problems with more complex roof types as can be seen from figure 28. Here it could successfully model the straight jump from one roof plane to another.

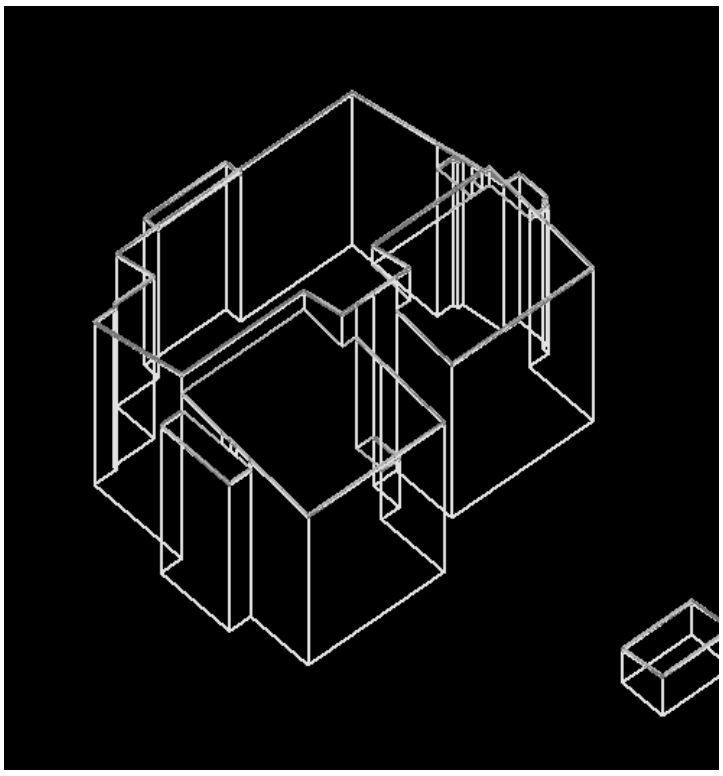

Figure 28. More complex roof type successfully modelled with Terrasolid software.

As Terrasolid software products use data driven reconstruction methods and do not have a fixed database of certain roof types, it was also possible to model smaller building parts as part of the actual building. From figure 29 can be seen that smaller attachments on the building's roof have also been modelled.

Problems were noticed with buildings that were covered by vegetation as it made the automatic classification of building points much harder and in some cases impossible. Relatively high point density in the area made modelling with Terrasolid software quite

successful in the case of most of the buildings. These building models were also constructed without a ground element, making these models not watertight.

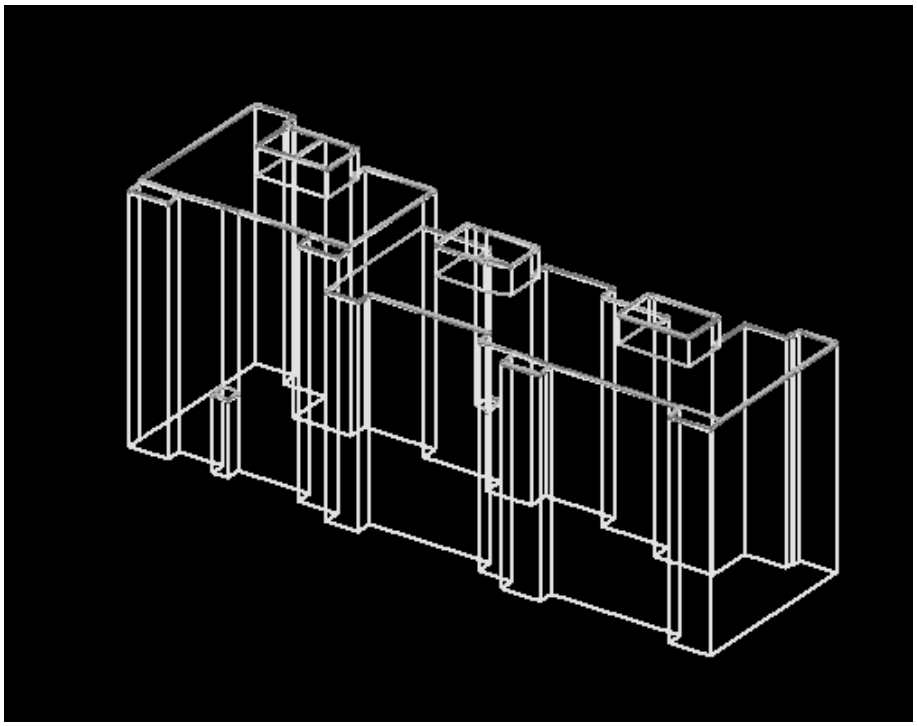

Figure 29. With Terrasolid software it was possible to model smaller building parts of a building.

BuildingReconstruction 2015 uses model driven approach to automatically reconstruct the buildings from a point cloud. It has a database of different roof types which it tries to match to the roofs that it finds from the point cloud. This approach works well when it comes to the simplest roof types but when it is a bit more complex, the algorithm runs into problems. In figure 30 can be seen a bit more complex presentation of a building data.

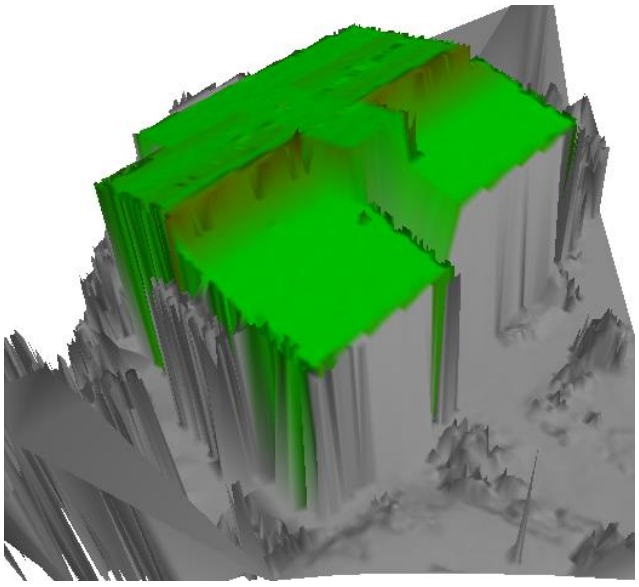

Figure 30. More complex building roof formation.

This type of roof was not in the database of the software and different parameters for the reconstruction did not help the software recognize the shape of this building. In figure 31 can be seen how the software ended up modelling this kind of roof types.

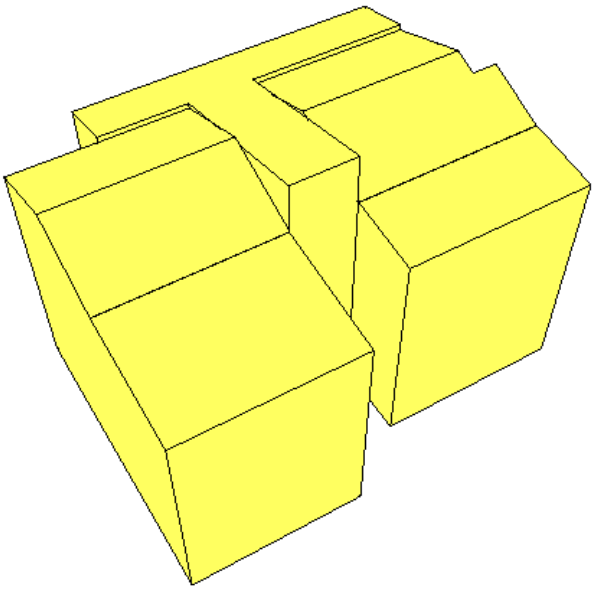

Figure 31. Incorrect modelling of the roof with BuildingReconstruction 2015.

As mentioned before, some buildings had smaller building parts on the roof. Example of this can be seen from figure 32 where smaller installation can be seen on the roof. These kinds of roofs are really hard to have in a database as such because there can be millions of small variations of them. The reconstruction algorithm would have to be more sophisticated and to model smaller parts separately and add them to the whole model later on.

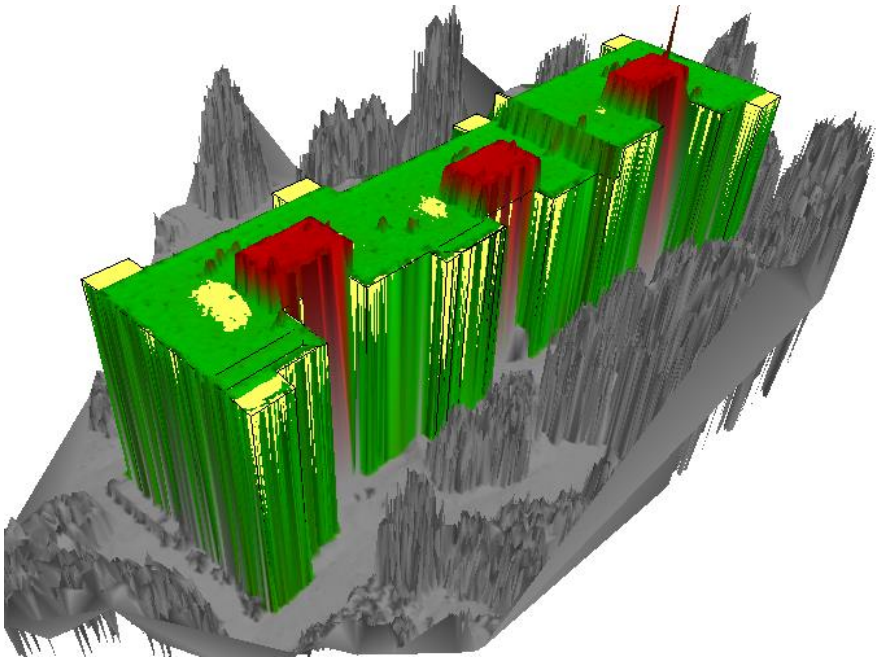

Figure 32. Example of a roof with building installations.

BuildingReconstruction 2015 does not have the ability to classify these smaller parts so the algorithm ignores them in the reconstruction phase. Figure 33 is an example of the building model produced from the situation of figure 32.

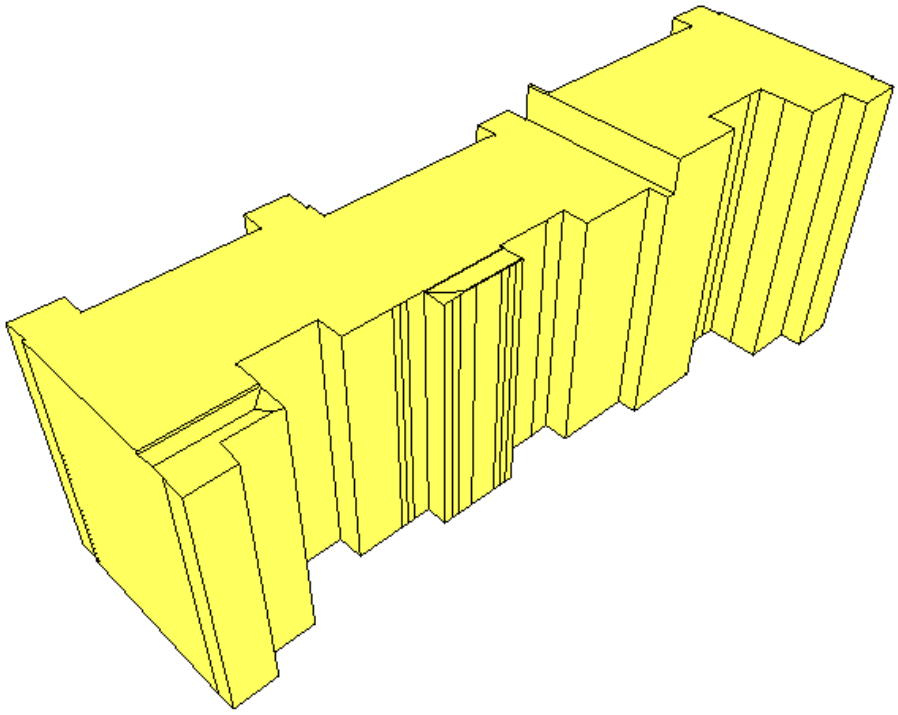

Figure 33. Model without building installations modelled with BuildingReconstruction 2015.

This software also used the footprints of the buildings in the modelling process. This allowed it to model more buildings than Terrasolid software because it knew where to look for them even if there were dense vegetation covering these buildings. This feature also caused some problems as can be seen from figure 34.

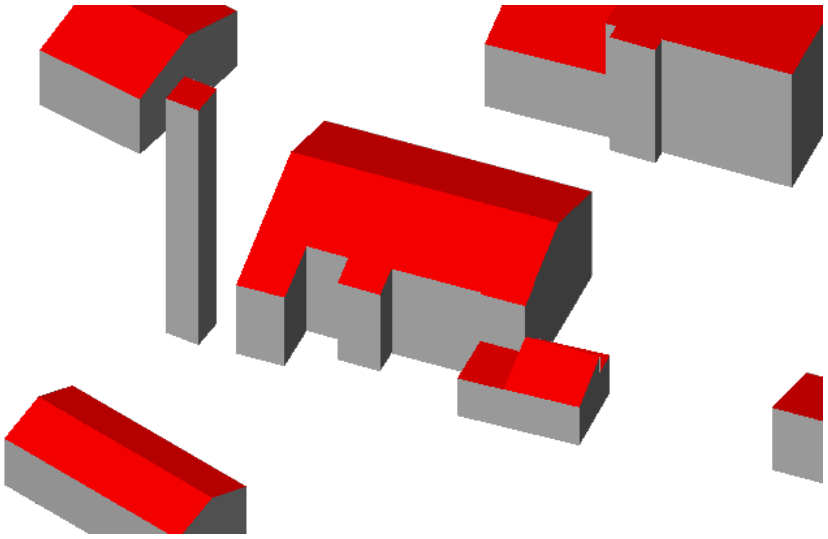

Figure 34. Incorrectly modelled building due to dense vegetation modelled with BuildingReconstruction 2015.

This problem is very understandable since there were only few hits from the actual building which practically made it impossible to model the building. Reason can be seen from figure 35. These types of buildings should be ignored since there is not much to do to fix them automatically if there is no prior height information available of the building.

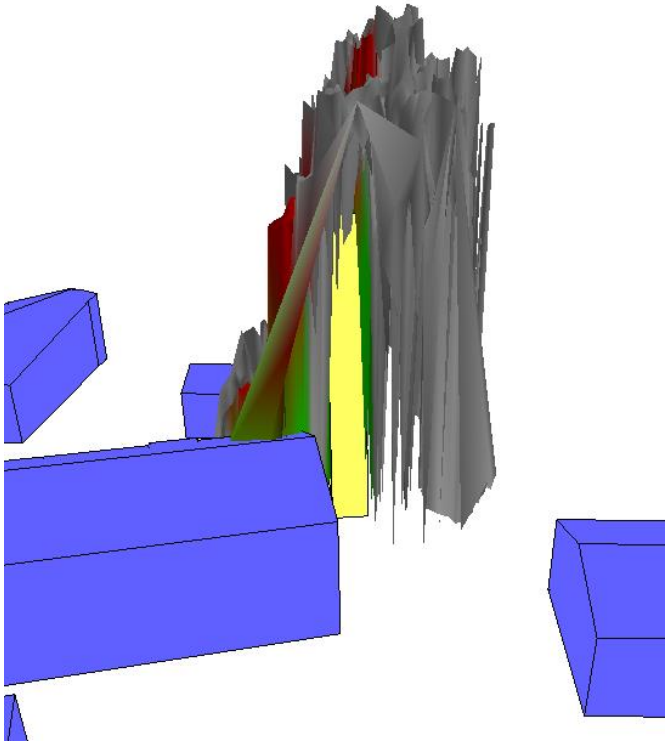

Figure 35. Dense vegetation covering the building causing erroneous result with BuildingReconstruction 2015.

Tridicon CityModeller also had problems with complex roof types. Same type of building as in figure 30 can be seen modelled with CityModeller in figure 36. The reason for incorrect modelling is the same as with BuildingReconstruction 2015; roof model database was insufficient. This also caused smaller objects as chimneys not to be modelled.

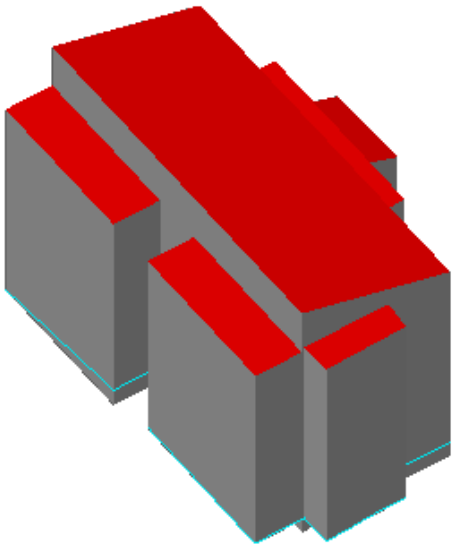

Figure 36. Complex building modelled with CityModeller

Tridicon CityModeller used building footprints in the process of detecting the buildings. This way the program could detect the main building and the outer installations that were attached to the building. In figure 37 can be seen an example of how overarching building parts should be modelled within CityGML according to Special Interest group 3D (SIG 3D).

Software had problems when it came to structures like these and they were modelled incorrectly as can be seen in figure 38.

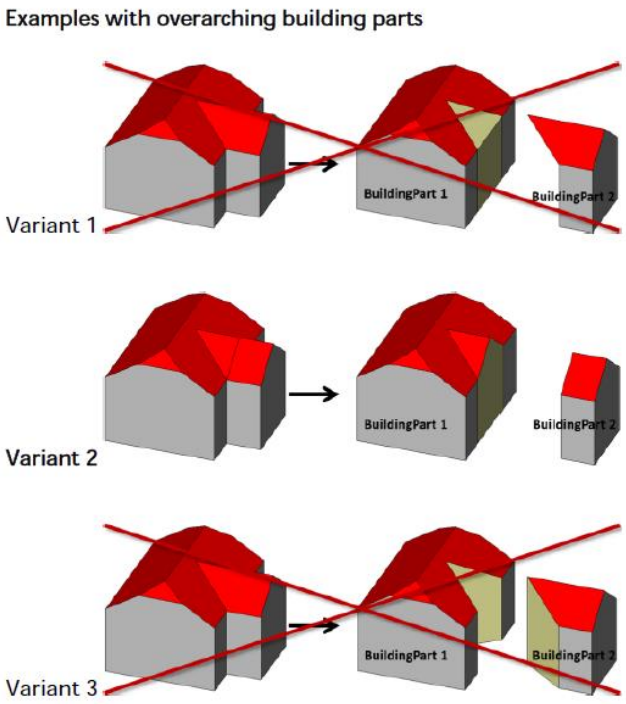

Figure 37. Correct and incorrect ways of modelling of the overarching parts of a building (SIG 3D 2014).

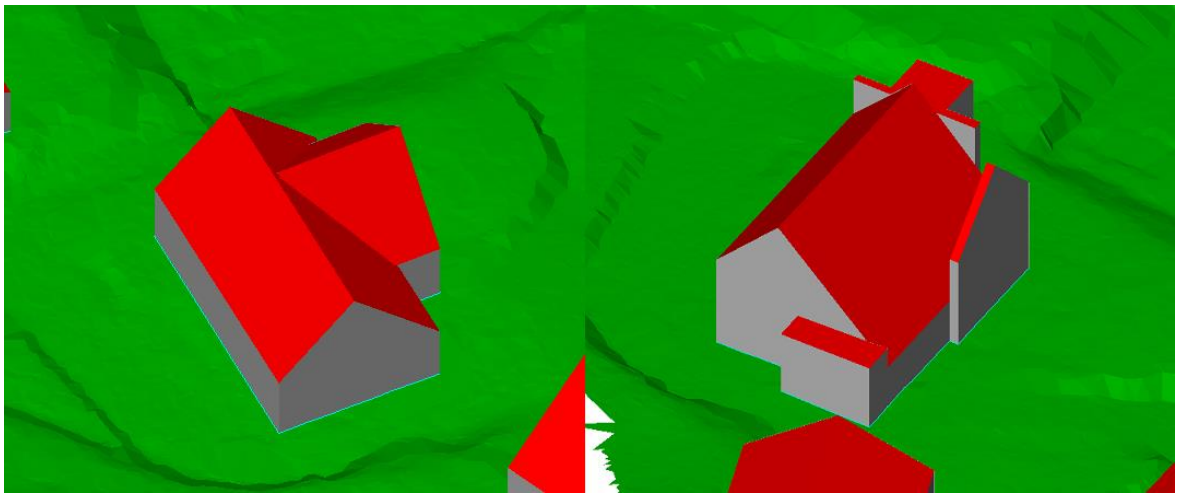

Figure 38. Incorrectly modelled building on the right modelled with Tridicon CityModeller and a better representation on the left modelled with Tridicon BuildingFinder.

Tridicon BuildingFinder could model these overarching parts with more success but as it did not use footprints it had problems of detecting buildings in parts where there were no buildings. In figure 39 there is an example where the software detects a hedgerow as a building and in figure 40 there is an example of detecting a planar surface and classifying it as a building.

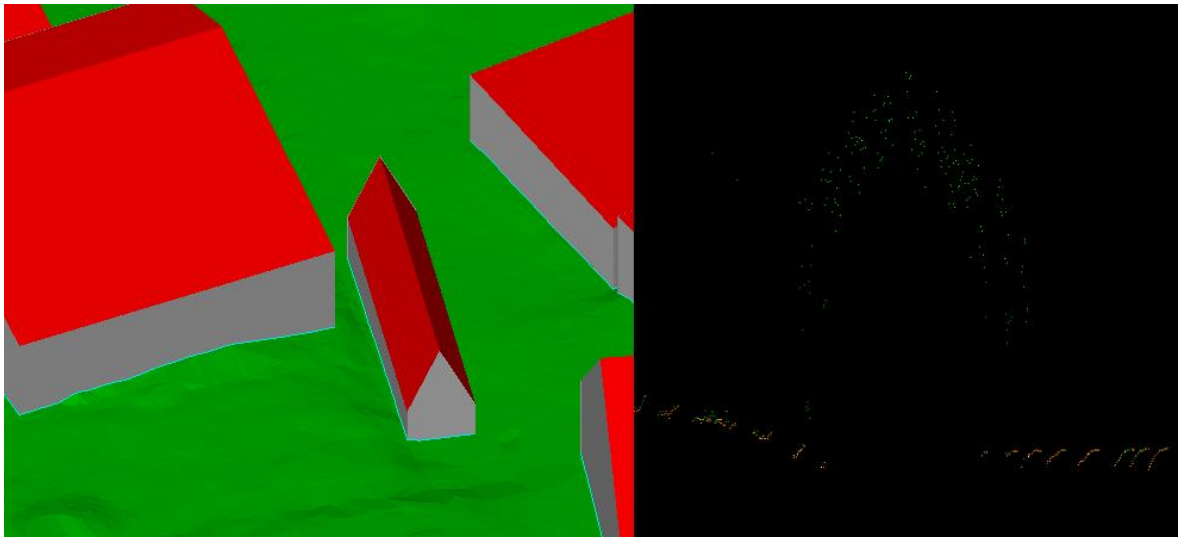

Figure 39. Incorrectly detected building in the middle of the left image modelled with Tridicon BuildingFinder and the cross-section of the hedgerow on the right.

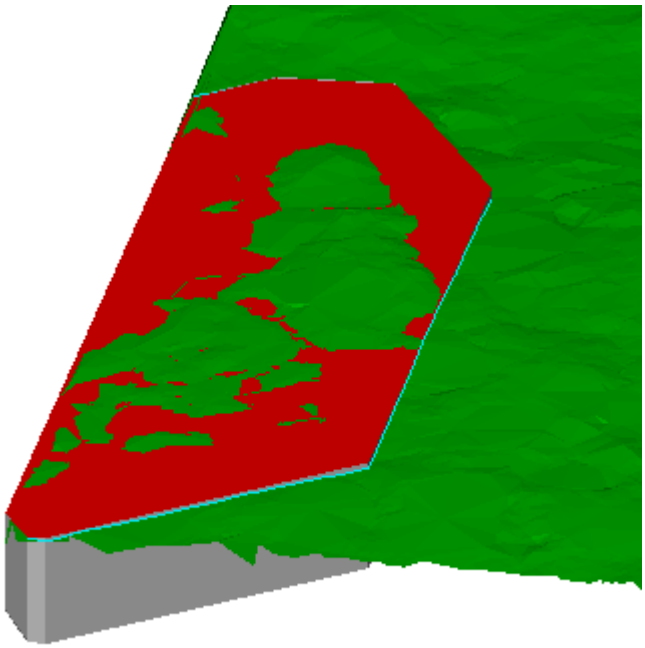

Figure 40. Incorrectly detected building probably due to planarity of the surface modelled with Tridicon BuildingFinder.

From table 3 can be seen the results of the building reconstruction phase. All of the software were able to find all or most of the buildings in the test areas. BuildingReconstruction 2015 and Tridicon CityModeller were both able to find all the buildings in the test areas. This was due to the used building footprints. Tridicon BuildingFinder found only 25 of the buildings in the test area 1 and also incorrectly modeled 3 objects as buildings. In test area 2 the software could not produce sensible result.

Table 3. Results of detection and reconstruction of buildings with each software. Area 1 is the first test area consisting mainly of detached houses and Area 2 is the second test area consisting mainly of apartment buildings.

	Terrasolid	**Building-Reconstruction 2015**	**Tridicon CityModeller**	**Tridicon BuildingFinder**
Buildings found	Area 1: 29/33 Area 2: 33/36	Area 1: 33/33 Area 2: 36/36	Area 1: 33/33 Area 2: 36/36	Area 1: 25/33 Area 2: -
Buildings missing	Area 1: 4/33 Area 2: 3/36	Area 1: 0/33 Area 2: 0/36	Area 1: 0/33 Area 2: 0/36	Area 1: 8/33 Area 2: -
Objects incorrectly modeled as buildings	Area 1: 0 Area 2: 0	Area 1: 0 Area 2: 0	Area 1: 0 Area 2: 0	Area 1: 3 Area 2: -

6.2 Building models to CityGML

When exporting the building models into COLLADA from MicroStation there was some problems with it projecting the models to wrong coordinates, and thus, making it useless. Exporting KML worked and the buildings were projected to right positions.

Problems arose when these models were tried to convert into CityGML format with FME. As a default FME marks all unrecognized objects as GenericCityObjects with LOD4 as it is the LOD with the most freedom. When visualized, the reconstructed building models were all GenericCityObjects without bounding surface information, such as which shapes are walls of the building. This fact alone causes the model to not reach the LOD2 specification in which it is said that bounding surfaces have to be determined.

In order to get the right syntax of CityGML easily out of FME with these data sets, it would require a lot of manual editing and formatting. Besides the CityGML that was exported from FME had problems with visualization. Some CityGML viewers, such as FZK Viewer developed by Karlsruhe Institute for Technology, Institute for Applied Science, could view the model, but for example AristotelesViewer developed in University of Bonn, Institute for Cartography and Geoinformation, had problems viewing the data.

With TerraPhoto it was possible to export the building models in COLLADA as separate buildings. But FME ran into same problems as before and could not export CityGML buildings out without manual work determining the surface members of each building.

Models exported from both MicroStation and TerraPhoto both suffered from lack of ground surface for the buildings. This means that the models are not according to OGC CityGML standard as they are not watertight models.

BuildingReconstruction 2015 allows direct export of the models to CityGML. These models are correctly formatted to be separate buildings that consist of roof, wall, and ground surfaces, as can be seen from figure 41.

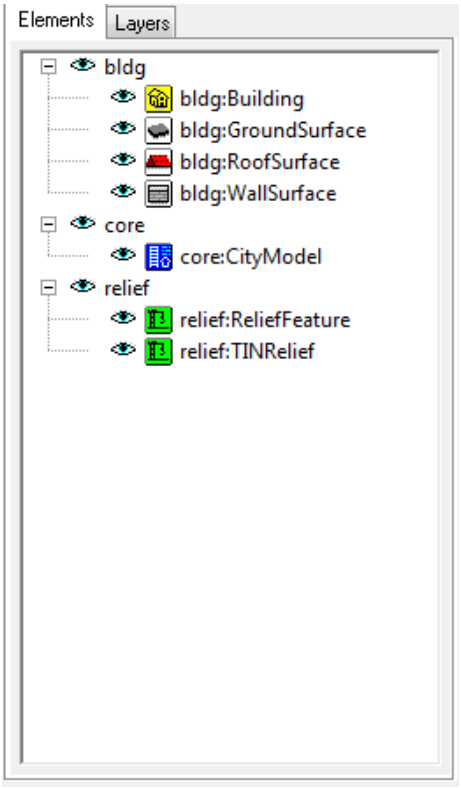

Figure 41. CityGML building model exported from BuildingReconstruction 2015.

These separate elements could then be visualized together or individually. As an example in figure 42 there are only walls visualized. All of these elements had information of which building they were part of. Different semantic information was also attached to the elements.

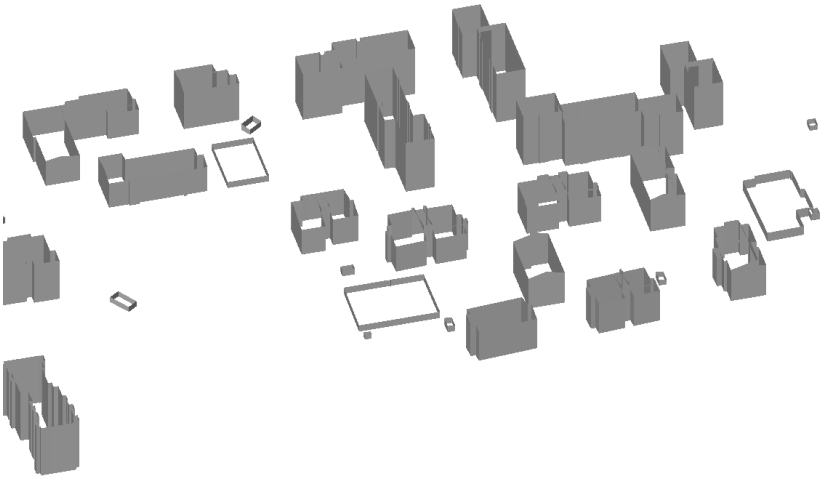

Figure 42. Example of an individual element layer of CityGML produced with BuildingReconstruction 2015.

Models exported as LOD1 were constructed correctly as they were without surface members and the roofs were modelled as flat roofs. The heights of the buildings were also modelled correctly according to OGC CityGML standard and SIG 3D (OGC 2012; SIG 3D 2014). Example of how models should be modelled can be seen in figure 43 and test case CityGML export between LOD1 and LOD2 in figure 44.

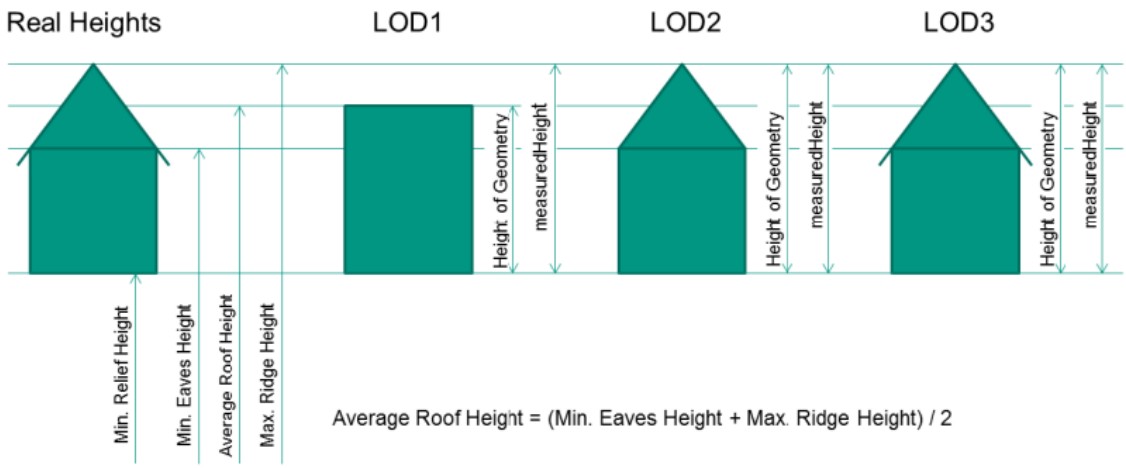

Figure 43. Example of building height specification (SIG 3D 2014)

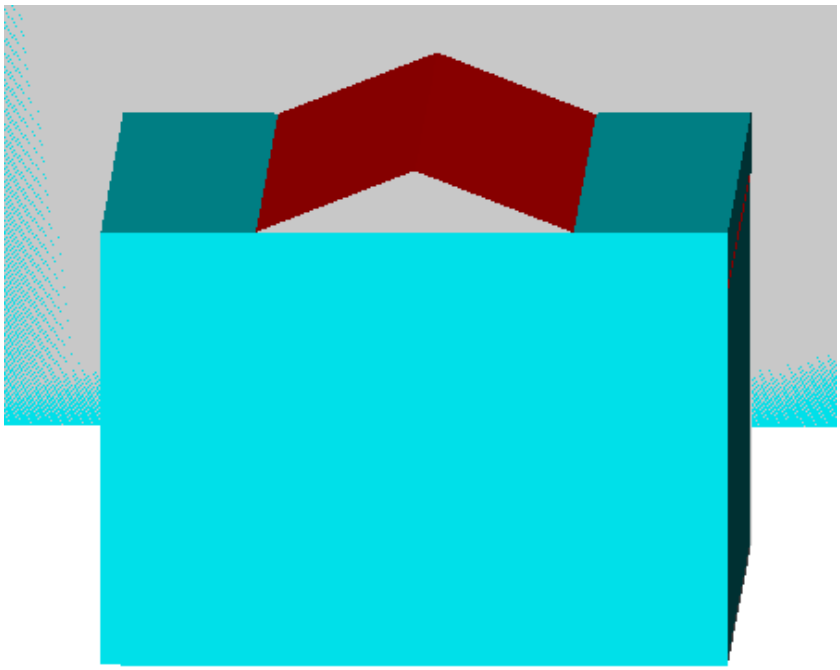

Figure 44. Example of the difference between LOD1 and LOD2 buildings exported with BuildingReconstruction 2015. LOD1 model can be as cyan and the roof of LOD2 model and be seen as red. Modelling follows the CityGML specification.

The problem with BuildingReconstruction was that it could only export the models in CityGML 1.0 so in order to get CityGML 2.0 it would be necessary to use some converter or do some manual editing to the XML code.

Also it was not possible to export the TIN relief out of this software. As a test I semi-manually constructed a TIN relief to see how it looks with the exported CityGML model. This was done by constructing a TIN relief out of model keypoints from TerraScan. Then by converting the TIN to Well Known Text in order to get easier access to the coordinates of the triangles. These coordinates were then read one by one with a small program that I

created for this purpose, and added to a separate XML file with the syntax of TINrelief of CityGML. Visualization of the relief with the buildings can be seen in figure 45.

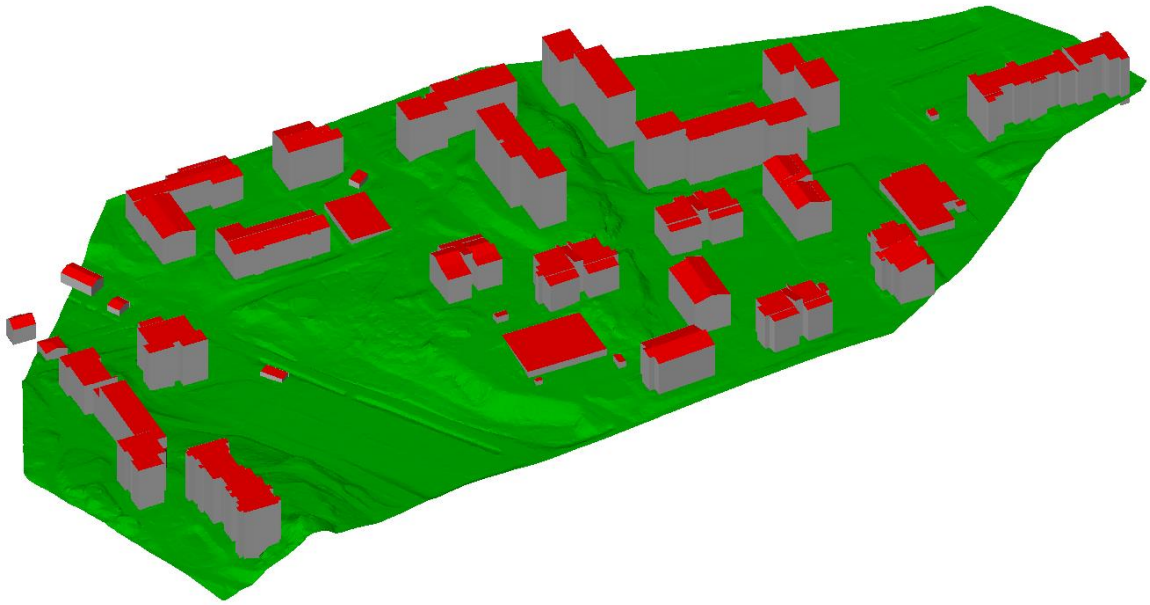

Figure 45. Visualization of the building models with the TIN relief

As mentioned before, with BuildingReconstruction it was possible to add semantics to the model. This was easiest to do by having some information as attributes in the building footprint shapes. These attributes can be basically anything and from the software the user has to define which source attribute links to which CityGML attribute. Example of this can be seen in figure 46.

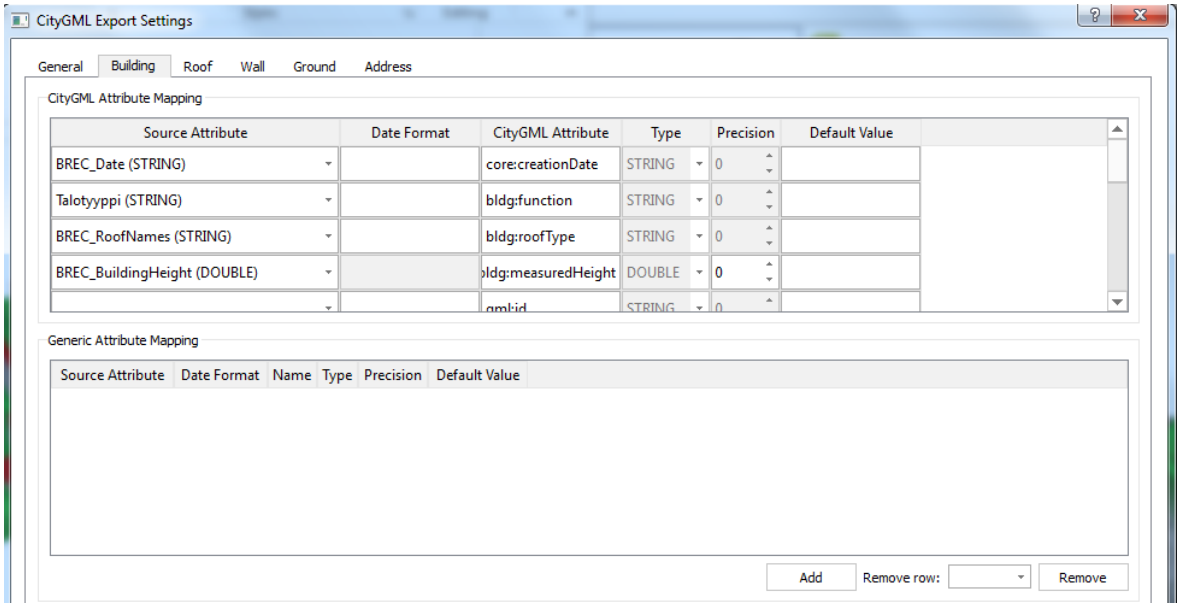

Figure 46. Example of linking different attributes with BuildingReconstruction 2015.

These footprint shapes can also have address information linked to them. By doing this, it is possible to give address information to the CityGML model in addition to the coordinate information. Example of adding address information to the CityGML export is seen in figure 47. In this case the address information was given as a default form in order to test the functionality but the information could as easily be taken from the individual shapes.

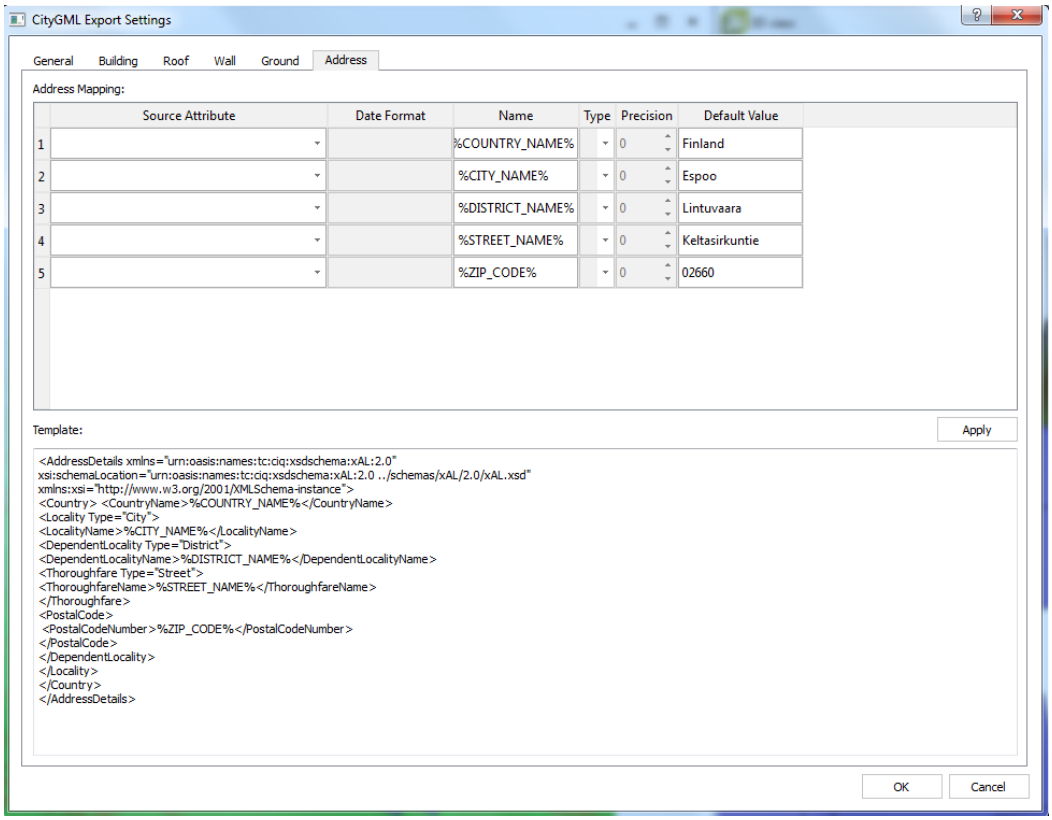

Figure 47. Adding address information to the CityGML with BuildingReconstruction 2015

In figure 48 is an example of how the added address information actually looks like in the CityGML code. This is of course individual for each of the buildings in the model and things like street number can easily be added to it.

```
<bldg:address>
 <core:Address>
  <core:xalAddress>
   <xAL:AddressDetails xsi:schemaLocation="urn:oasis:names:tc:ciq:xsdschema:xAL:2.0 ../schemas/xAL/2.0/xAL.xsd">
    <xAL:Country>
     <xAL:CountryName>Finland</xAL:CountryName>
     <xAL:Locality Type="City">
      <xAL:LocalityName>Espoo</xAL:LocalityName>
      <xAL:DependentLocality Type="District">
       <xAL:DependentLocalityName>Lintuvaara</xAL:DependentLocalityName>
       <xAL:Thoroughfare Type="Street">
        <xAL:ThoroughfareName>Keltasirkuntie</xAL:ThoroughfareName>
       </xAL:Thoroughfare>
       <xAL:PostalCode>
        <xAL:PostalCodeNumber>02660</xAL:PostalCodeNumber>
       </xAL:PostalCode>
      </xAL:DependentLocality>
     </xAL:Locality>
    </xAL:Country>
   </xAL:AddressDetails>
  </core:xalAddress>
 </core:Address>
</bldg:address>
```

Figure 48. Example of the address syntax in CityGML

Semantic information can be queried from the model to search for example a certain type of attribute. Examples of how these semantics can be visualized can be seen in figures 49 and 50. Figure 49 is taken from FZK Viewer and figure 50 from Aristoteles3D.

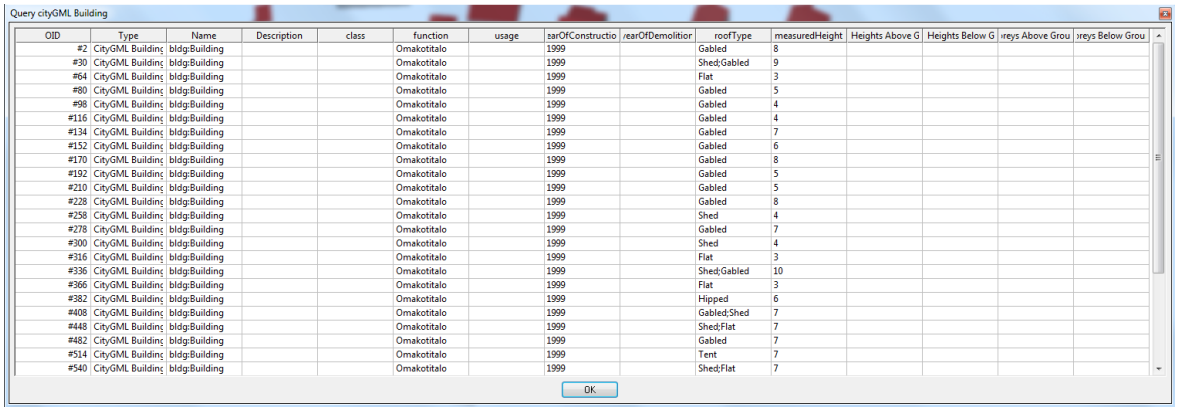

Figure 49. Example of attribute data that can be queried

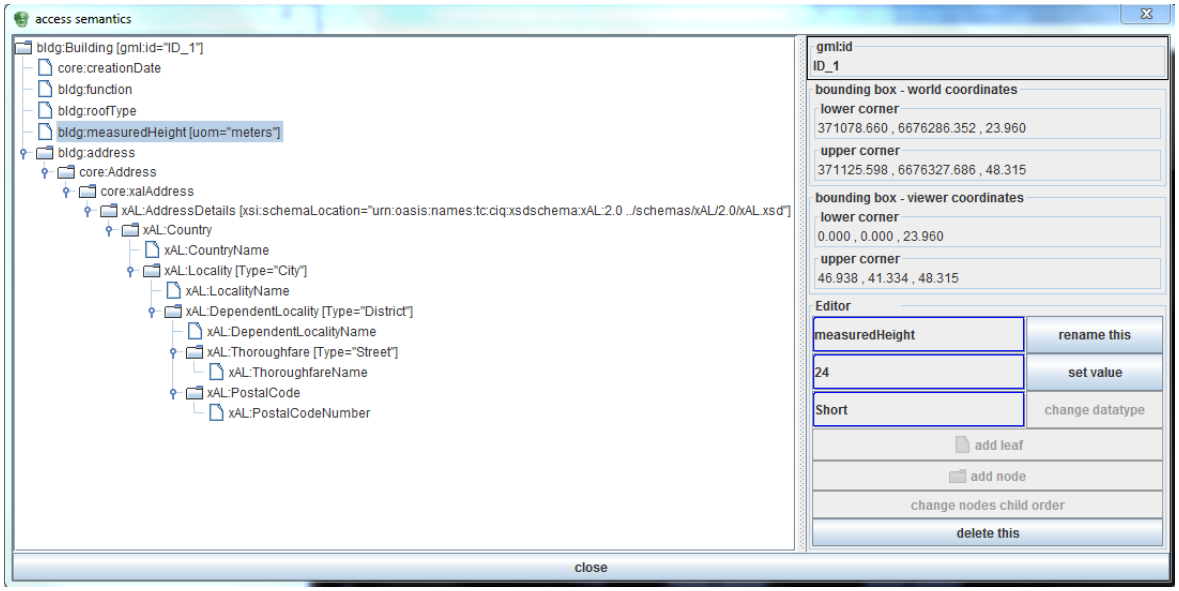

Figure 50. Example of how the attribute data looks like in a CityGML visualization.

The models created with Tridicon CityModeller and BuildingFinder could be exported into CityGML format with Tridicon Export. The export function is quite reliable and exports CityGML 1.0 standard. Ground surface can be added to the model in the export phase which allows the CityGML to have a TIN model for visualization and analysis purposes. This also enables that the terrain intersection curves have information about them in the CityGML file. In figure 51 there is an example of how the software was able to model the TIC for the buildings.

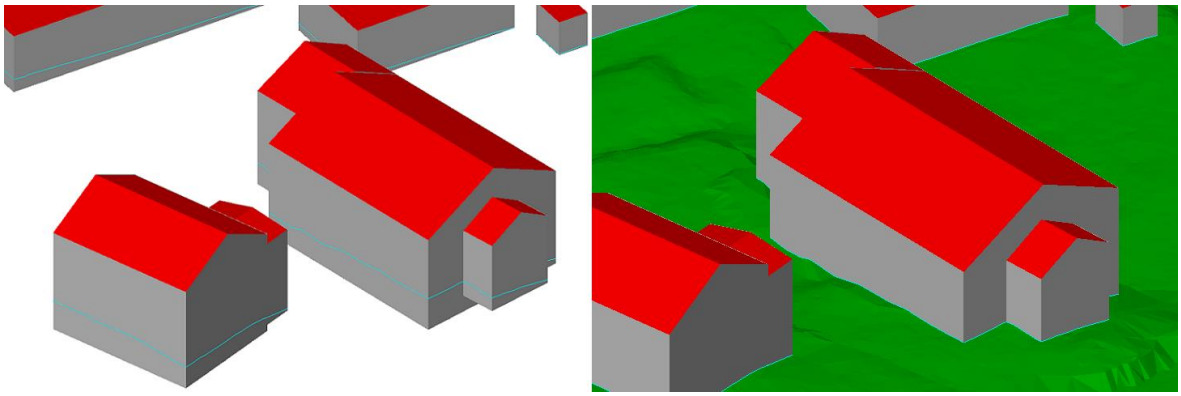

Figure 51. Result of the terrain intersection curve export with Tridicon Export. The TIC can be seen as cyan line on the face of the building.

As Tridicon CityModeller used building footprints, it could recognize which parts of the building could be considered as the "main" part. This allowed the separation in CityGML between buildings and building parts. In figure 52 can be seen an example of how in the CityGML visualization can be made difference between main building and for example a garage.

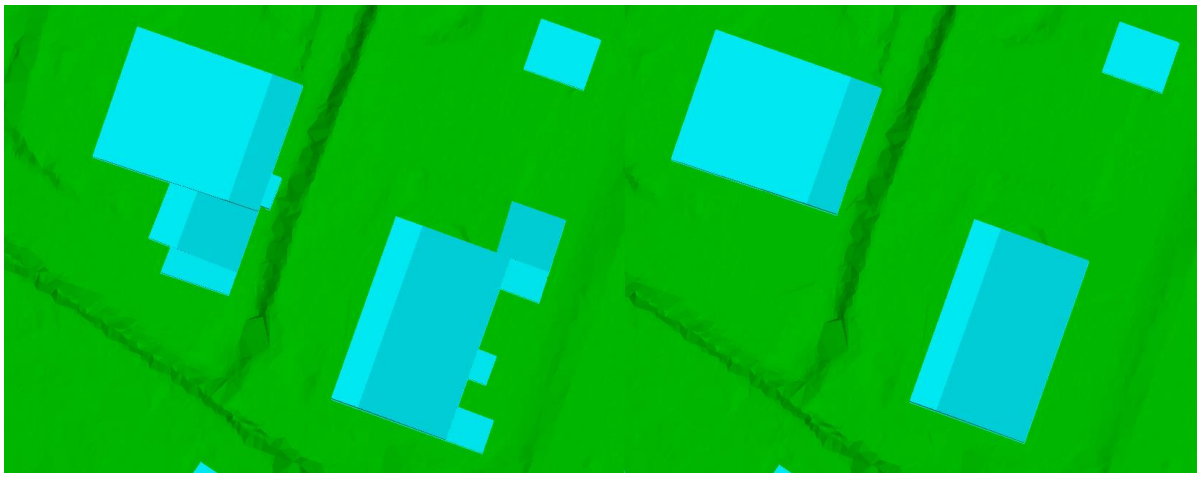

Figure 52. Example of buildings with building parts constructed with Tridicon CityModeller. On the left are buildings and their parts visualized and on the right only the software's guess for the main building.

Table 4. Comparison between different software

	Terrasolid	BuildingReconstruction 2015	Tridicon
Direct export to CityGML	No*	Yes	Yes**
Terrain Intersection Curve	No	No	Yes
TIN model export	Yes	No	Yes
CityGML ReliefFeature	No	No	Yes
Semantics	No	Yes	Yes
Watertight models	No	Yes	Yes
Chimneys etc. modelled	Yes	No	No
Building parts	No	No	Yes
Ability to add ADEs	With FME	No	No

*FME plug-in is available to export CityGML from MicroStation
**With Tridicon Export from models created with other Tridicon software

Table 4 shows how software used in this thesis differ from each other. In it can be seen what results were obtained with each tested software and which results were not possible to get from them.

With Terrasolid software directly getting LOD2 was impossible since it is still missing CityGML export function. Using FME with Terrasolid building models is possible, but as the way FME interprets these models is quite unique, a lot of manual work is needed in order to create a working translator in FME. This translator should be able to recognize which shapes belong to which building, give them correct ids and building part names. It should also be able to create the ground slab for the building.

In the manual of BuildingReconstruction it says that the software is able to export the building models in LOD2. Yet in the OGC CityGML specification it is mentioned that for building models to be considered LOD2, they should have information of building installations attached to them. This is because BuildingReconstruction only exports CityGML version 1.0 in which building installations were not yet classified to be part of LOD2 (OGC 2008). In this sense this software can be seen quite outdated as CityGML 2.0 should be used instead of CityGML 1.0. CityGML produced with BuildingReconstruction requires a lot of manual work in order to convert it to 2.0 and bringing it up to date.

Table 5. Geometrical result of models. Area 1 is the first test area consisting mainly of detached houses.

Test area 1 **Buildings: 33**

	BuildingReconstruction 2015	Tridicon CityModeller	Tridicon BuildingFinder
LOD1 buildings	32	20	18
LOD2 buildings	13	14	5
LOD1 successfulness	97.0%	60.6%	54.5%
LOD2 successfulness	39.4%	42.4%	15.2%

Table 6. Geometrical result of models. Area 2 is the second test area consisting mainly of apartment buildings.

Test area 2 **Buildings: 36**

	BuildingReconstruction 2015	Tridicon CityModeller	Tridicon BuildingFinder
LOD1 buildings	36	24	-
LOD2 buildings	11	13	-
LOD1 successfulness	100%	66.7%	-
LOD2 successfulness	30.6%	36.1%	-

Table 5 and 6 show how many of the buildings produced with each software could be seen to be in line with LOD1 and LOD2 specifications from their geometrical aspects. It can be seen that buildings overall could be geometrically modeled very well in line with LOD1 specification whereas LOD2 models were much harder to reach. Most of the LOD2 buildings were smaller sheds, which made it much easier to achieve the specification. Tridicon BuildingFinder did not produce any sensible result in test area 2 as stated before. LOD1 models created with BuildingReconstruction followed the roof shape requirements better than Tridicon software. Tridicon did not model all of the LOD1 buildings with flat roof.

Validation of the created CityGML files were done with two different software. The created CityGML files had outdated schema information in them and these had to be updated before the validation could be performed. Firstly, XML Tools plug-in for Notepad++ was tested with the files and all of the files were successfully validated. Second software tested was 3D CityDB for CityGML developed by VirtualcitySYSTEMS GmbH in collaboration with Technische Universität Berlin and University of Bonn, Institute for Cartography and Geoinformation. With this software the files were also validated successfully, and thus, can be seen to be in correct format.

7 Summary and conclusions

The objective of this thesis was to test the current situation of the process of reconstructing buildings from airborne laser scanning and exporting these building models in CityGML data format. These exported CityGML files were then inspected and compared against the official CityGML standard by Open Geospatial Consortium.

The thesis was divided into two main parts: theoretical, and empirical parts. In the theoretical part the basics of laser scanning, building detection and reconstruction, and CityGML standard have been explained. In the empirical part tests were ran on the data available and the obtained results were inspected.

Two test areas were chosen; one with mainly detached houses and smaller sheds of which some were partly covered by trees, and another one with mainly apartment buildings. Both of these areas had variety of different types of roof structures in order to test the reconstruction capabilities of different software.

From the perspective of this thesis, an important quality for the modelled buildings, regarding to the CityGML specification, is that smaller building installations should be modelled as a part of the building. Of the tested software during this thesis, only Terrasolid software was able to produce building models where also smaller building objects, such as chimneys, were modelled.

The tested software used different algorithms for the building modelling, which is the cause of the varying results for smaller building objects. Terrasolid was the only software package that used data driven methods for the building detection and reconstruction phase while other software used model driven methods. Model driven methods has to either have a huge library of different roof types, which would have huge space requirements as well as slower the reconstruction phase down, or the algorithm has to be sophisticated enough to recognize smaller objects and model these as, for example small flat roofed “building” which is connected to the main building.

In order to reach the CityGML 2.0 standard for buildings in LOD2 according to OGC, building installations have to be modelled within CityGML. The test results in this thesis show that automatically this process is still quite a challenge. Only Terrasolid software had the ability to model smaller parts of the building but getting these models to CityGML format was quite hard task and would basically require making its own program with FME to convert these models in the right way.

High point density is also needed when taking into account the requirement for smaller building installations as these are much harder for algorithms to detect from lower point density. By increasing the point density, the smaller objects like chimneys will be hit with more pulses resulting in a more coherent shape that then can be classified more easily. With lower point density it would be hard to tell apart for example a small dormer and a chimney.

The models that were reconstructed with BuildingReconstruction 2015 had problems with more complex roof types as was shown in figure 31. Solution to fix these roofs would be to manually edit them and try to make them look as close to the original as possible. This software offers a variety of editing tools for the buildings but it is highly manual work and can be seen as a bad solution when modelling bigger areas that require a lot of work. Automatically it seemed to be impossible to get the right result with this software.

The lack of ground surface from Terrasolid software can be solved by a new program. A program would be possible to be created that reads the wall surfaces and creates a new polygon with the lower corner coordinates of each wall. Other possibility, albeit slightly worse one, would be to create a TIN model from the ground points. Coordinates of the ground surface for the building could then be extracted from the TIN model inside the shapes that the walls form. This approach is not optimal since it would use the TIN model as ground surface and due to how triangulation works, the ground would not be a flat surface which, according to OGC CityGML standard, is sub-optimal case.

The table 4 in section 6.2 represents the possibilities of each tested software. When this table is compared with the table 1 that has the proposed requirements for different LODs according to OGC CityGML standard, can be seen how well each software delivered their models within standard.

Both BuildingReconstruction 2015 and Tridicon software offer a way to export to CityGML within their own systems. This is beneficial since it is easier to export data format from a known source rather than to create a universal translator between a city model and a certain data format. It can be seen from table 1 that LOD1 is possible to produce with BuildingReconstruction 2015 and Tridicon software as building installations are not required for this and both of these software offer a way to export LOD1 either in the same file as LOD2 or as a separate file.

Models created with both were watertight, and thus, align with the standard. Both also had the ability to easily add semantics to the models. With BuildingReconstruction the interface to do this was easier to use as it was done by simply filling certain fields whereas with Tridicon, a comma separated value file had to be created to add information to the model.

Tridicon software had few big advantages over BuildingReconstruction 2015. The biggest of these were the ability to model the TIN relief of the area and this way also calculate the TIC for the buildings. This was the only software that I found which had the ability to do this automatically. Another advantage was the ability to divide the models into “main” building and building parts. This is important for the work of building installation where for example balconies have to be modelled separately of the building and in this work the first step would be to recognize where the main building and balcony intersect. But as of now, it was not possible to create building installations within these software and because of this, the models cannot be considered LOD2 if we follow the recommended requirements for LOD2 found in OGC CityGML 2.0 standard.

Roof overhanging parts are not needed for LOD1 nor LOD2 according to OGC CityGML standard. In BuildingReconstruction 2015 and Tridicon software these are impossible to get automatically since they mostly use building footprints in the modelling process, but with TerraScan they are theoretically possible to obtain, albeit this can be quite a challenging job in some cases if only airborne laser data is used.

These software follow the guidelines of CityGML 1.0 standard in which building installations are not yet specified to be part of LOD2 models. With BuildingReconstruction this is also the reason for the lack of ability to export TIN relief and TIC for the models.

The reason for the export of only CityGML 1.0 could be the added costs and problems that the addition of building installations would bring. These reasons can potentially overshadow

the benefits that producing CityGML 2.0 would bring. The market for actual CityGML 2.0 might be so small that it is not economically beneficial to “waste" resources in order to automatically convert from 1.0 to 2.0.

LOD1 does not require building parts which would make TerraScan suitable for producing models of this level. TerraScan automatically produced building models with detailed roof structures which is not in line with LOD1 specification, so in order to use these building models for CityGML LOD1 export, they have to be simplified first. This simplification can be done with FME but finesse is needed when the height of the model is formed because rules, such as in figure 19, apply to the height modelling.

None of the tested methods could be deemed as fully automatic as each could be clearly divided into steps. All of the methods required a step where the raw point cloud was processed into suitable classes and products. Another step in the process was to reconstruct the buildings from the processed point cloud. Lastly the reconstructed models had to have semantics added to them and exported to CityGML format. Fully automatic process would require the use of a macro or a script of some sort that would be able to do all of the above steps with a given set of parameters.

If point density is high enough, then the results from this study suggest that LOD2 building models can be produced from ALS alone. Keeping in mind that without high scanning angle, the balconies will be modelled as one block since there will not be any returns between different balconies.

The required level-of-detail should be considered when producing models. For most applications LOD1 is enough to cover the purpose. Lower LODs also have the benefit that it is easier to update them and they do not set such high demands for hardware or connections as models with higher amount of details.

Adding texture is also possible for these models and in this process one has to take into account that higher LODs will have more wall surfaces and overhanging parts. These facts will make it harder to add the right texture to the corresponding surface. And as the wall surfaces multiply, also the time and resources needed for the computation of the textures will grow significantly.

Producing LOD3 models from only ALS seems impossible with the current methods and software. Also automatically producing them can be quite a challenging task because most commercial software products only support exporting of LODs up to LOD2. For the creation of LOD3 in addition to ALS data, data from other sources is also needed. Combining ALS and land-based mobile laser scanning could be enough to create LOD3. Also images could be added to this equation to provide more information about the buildings.

References

Arefi, H., Engels, J., Hahn, M. & Mayer H. 2008. Level of Detail in 3D Building Reconstruction from LIDAR Data. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 37, part B3b. pp. 485-490.

Awrangjeb, M. & Fraser, C. S. 2014. Automatic segmentation of raw LiDAR data for extraction of building roofs. Remote Sensing. Vol. 6:5. pp. 3716-3751. DOI: 10.3390/rs6053716. ISSN: 2072-4292.

Awrangjeb, M., Lu, G. & Fraser, C. 2014. Automatic building extraction from LIDAR data covering complex urban scenes. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 40, part 3. pp. 25-32.

Axelsson, P. 1999. Processing of laser scanner data—algorithms and applications. ISPRS Journal of Photogrammetry and Remote Sensing. Vol. 54:2-3. pp. 138-147. DOI: 10.1016/S0924-2716(99)00008-8. ISSN: 0924-2716.

Baltsavias, E.P. 1999a. Airborne laser scanning: basic relations and formulas. ISPRS Journal of Photogrammetry and Remote Sensing. Vol. 54:2-3. pp. 199-214. DOI: 10.1016/S0924-2716(99)00015-5. ISSN: 0924-2716.

Baltsavias, E.P. 1999b. Airborne laser scanning: existing systems and firms and other resources. ISPRS Journal of Photogrammetry and Remote Sensing. Vol. 54:2-3. pp. 164-198. DOI: 10.1016/S0924-2716(99)00016-7. ISSN: 0924-2716.

Baltsavias, E.P. 1999c. A comparison between photogrammetry and laser scanning. ISPRS Journal of Photogrammetry and Remote Sensing. Vol. 54:2-3. pp. 83-94. DOI: 10.1016/S0924-2716(99)00014-3. ISSN: 0924-2716.

Boehler, W., Vicent, M. B. & Marbs, A. 2003. Investigating laser scanner accuracy. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 34, part 5, pp. 696-701.

Brenner, C. & Haala, N. 1998. Fast production of virtual reality city models. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 32:4.

Brenner, C. 2000. Towards fully automatic generation of city models. International Archives of Photogrammetry and Remote Sensing. Vol. 33 (B3/1; part 3). pp. 84-92.

Brenner, C., Haala, N., & Fritsch, D. 2001. Towards fully automated 3D city model generation. Automatic Extraction of Man-Made Objects from Aerial and Space Images (III). pp. 47-57.

Buyukaslih, I., Isikdag, U. & Zlatanova, S. 2013. Exploring the processes of generating LOD (0-2) CityGML models in greater municipality of Istanbul. In 8th 3DGeoInfo Conference & WG II/2 Workshop, Istanbul, Turkey. ISPRS Archives Vol. II-2/W1.

Callieri, M., Dellepiane, M. & Scopigno, R. 2015. Remote Visualization and Navigation of 3D Models of Archeological Sites. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 40, part 5/W41. pp. 147-154.

Carter, J., Schmid, K., Waters, K., Betzhold, L., Hadley, B., Mataosky, R. & Halleran, J. 2012. Lidar 101: An Introduction to Lidar Technology, Data and Applications. National Oceanic and Atmospheric Administration (NOAA) Coastal Services Center. Charleston, SC. p. 76.

Cheng, L., Gong, J., Li, M. & Liu, Y. 2011. 3D building model reconstruction from multi-view aerial imagery and lidar data. Photogrammetric Engineering & Remote Sensing. Vol. 77:2. pp. 125-139. ISSN: 0099-1112.

CityGML, http://www.citygml.org/ [Accessed: 29.7.2015].

Dorninger, P. & Pfeifer, N. 2008. A Comprehensive Automated 3D Approach for Building Extraction, Reconstruction, and Regularization from Airborne Laser Scanning Point Clouds. Sensors. Vol. 8:11. pp. 7323-7343. DOI: 10.3390/s8117323. ISSN 1424-8220.

Döllner, J. & Buchholz, H. 2005. Continuous level-of-detail modeling of buildings in 3D city models. In Proceedings of the 13th annual ACM international workshop on Geographic information systems. pp. 173-181.

Döllner, J., Kolbe, T. H., Liecke, F., Sgouros, T. & Teichmann, K. 2006. The Virtual 3D City Model of Berlin - Managing, Integrating, and Communicating Complex Urban Information. In Proceedings of the 25th Urban Data Management Symposium UDMS. Vol. 2006.

Elberink, S. O. 2008. Problems in automated building reconstruction based on dense airborne laser scanning data. Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 37, part B3a. pp. 93-98.

Fan, H., Meng, L. & Jahnke, M. 2009. Generalization of 3D buildings modelled by CityGML. In Advances in GIScience. pp. 387-405. DOI: 10.1007/978-3-642-00318-9_20.

Gröger, G. & Plümer, L. 2012. CityGML – Interoperable semantic 3D city models. ISPRS Journal of Photogrammetry and Remote Sensing. Vol. 71. pp 12-33. DOI: 10.1016/j.isprsjprs.2012.04.004. ISSN: 0924-2716.

Haala, N., & Brenner, C. 1997. Generation of 3D city models from airborne laser scanning data. In Proceedings EARSEL Workshop on LIDAR remote sensing on land and sea.

Haala, N. & Kada, M. 2010. An update on automatic 3D building reconstruction. ISPRS Journal of Photogrammetry and Remote Sensing. Vol. 65:6, pp. 570-580. DOI: 10.1016/j.isprsjprs.2010.09.006. ISSN: 0924-2716.

Isikdag, U. & Zlatanova, S. 2009. Towards defining a framework for automatic generation of buildings in CityGML using building Information Models. In 3D Geo-Information Sciences. pp. 79-96.

ISO 19109. 2005. Geographic information - Rules for application schema.

Karsli, F. & Kahya, O. 2008. Building extraction from laser scanning data. Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 37, part B3b. pp. 289-294.

Katzenbeisser, R. 2003. About the calibration of lidar sensors. In ISPRS Workshop. pp. 1-6.

Kolbe, T. H. 2009. Representing and Exchanging 3D City Models with CityGML. In 3D geo-information sciences. pp. 15-31. Springer Berlin Heidelberg.

Kolbe, T. H. 2007. CityGML-3D geospatial and semantic modelling of urban structures. Presentation on the GITA/OGC emerging technologies summit in Washington.

Landau, H., Vollath, U. & Chen, X. 2002. Virtual reference station systems. Journal of Global Positioning Systems. Vol. 1:2. pp. 137-143. ISSN 1446-3156.

Liukkonen, O. 2015. Kuntien paikkatiedon polku kantakartasta 3D-kaupunkimalliin. Master thesis, Aalto University. p. 92.

Mallet, C. & Bretar, F. 2009. Full-waveform topographic lidar: State-of-the-art. ISPRS Journal of Photogrammetry and Remote Sensing. Vol. 64:1. pp. 1-16. DOI: 10.1016/j.isprsjprs.2008.09.007. ISSN: 0924-2716.

Mao, B. 2011. Visualization and Generalization of 3D City Models. Doctoral Thesis, Royal Institute of Technology (KTH), Department of Urban Planning and Environment.

Matikainen, L., Hyyppä, J., Ahokas, E., Markelin, L. & Kaartinen, H. 2010. Automatic Detection of Buildings and Changes in Buildings for Updating of Maps. Remote Sensing. Vol. 2:5. pp. 1217-1248. DOI: 10.3390/rs2051217. ISSN: 2072-4292.

Morgan, M. & Tempfli, K. 2000. Automatic building extraction from airborne laser scanning data. International Archives of Photogrammetry and Remote Sensing. Vol. 33:(B3/2; part 3). pp. 616-623.

Open Geospatial Consortium (OGC). 2012. OGC City Geography Markup Language (CityGML) Encoding Standard Version 2.0.0. Available online: http://www.opengis.net/spec/citygml/2.0 [Accessed: 25.06.2015].

Open Geospatial Consortium (OGC). 2008. OpenGIS® City Geography Markup Language (CityGML) Encoding Standard. Version: 1.0.0. Available online: http://www.opengeospatial.org/standards/citygml [Accessed: 15.10.2015]

Pfeifer, N. & Briese, C. 2007. Geometrical aspects of airborne laser scanning and terrestrial laser scanning. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 36, part 3 / W52, pp. 311-319.

Prandi, F., Devigili, F., Soave, M., Di Staso, U. & De Amicis, R. 2015. 3D web visualization of huge CityGML models. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 40, part 3/W3, pp. 601-605.

Prieto, I., Izkara, J. L. & Delgado, F. J. 2012. From point cloud to web 3D through CityGML. 18th international IEEE conference on virtual systems and multimedia (VSMM). pp. 405-412.

Rabbou, M. A. & El-Rabbany, A. 2014. Non-Linear Filtering for Precise Point Positioning GPS/INS integration. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences. Vol. 40, part 2. pp. 127-132.

Rottensteiner, F. & Briese, C. 2002. A new method for building extraction in urban areas from high-resolution LIDAR data. International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences. Vol. 34, part 3/A. pp. 295-301.

Sampath, A. & Shan, J. 2010. Segmentation and Reconstruction of Polyhedral Building Roofs from Aerial Lidar Point Clouds. IEEE Transactions on Geoscience and Remote Sensing. Vol. 48:3. pp. 1554-1567.

Schindler, K., & Bauer, J. 2003. A model-based method for building reconstruction. Higher-Level Knowledge in 3D Modeling and Motion Analysis. pp. 74-82.

SIG 3D. 2014. Modeling Guide for 3D Objects, Part 2: Modeling of Buildings (LoD1, LoD2 and LoD3). SIG 3D – Quality Working Group / EC, Germany.

Stadler, A. & Kolbe, T. H. 2007. Spatio-semantic coherence in the integration of 3D city models. In Proceedings of the 5th International Symposium on Spatial Data Quality.

Tarsha-Kurdi, F., Landes, T., Grussenmeyer, P. & Koehl, M. 2007a. Model-driven and data-driven approaches using lidar data: analysis and comparison. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 36, part 3. pp. 87-92.

Tarsha-Kurdi, F., Landes, T. & Grussenmeyer, P. 2007b. Joint combination of point cloud and DSM for 3D building reconstruction using airborne laser scanner data. In Urban Remote Sensing Joint Event. pp. 1-7.

Tarsha-Kurdi, F., Landes, T. & Grussenmeyer, P. 2007c. Hough-transform and extended ransac algorithms for automatic detection of 3D building roof planes from lidar data. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 36, part 3/W52. pp. 407-412.

Terrasolid Oy. 2015. TerraScan User's Guide. Available online: https://www.terrasolid.com/download/tscan.pdf [Accessed: 24.10.2015].

Tomljenovic, I., Höfle, B., Tiede, D., & Blaschke, T. 2015. Building Extraction from Airborne Laser Scanning Data: An Analysis of the State of the Art. Remote Sensing, Vol. 7:4. pp. 3826-3862. DOI: 10.3390/rs70403826. ISSN: 2072-4292.

Tridicon BuildingFinder, Tridicon BuildingFinder – Generation of Point Clouds from Aerial Images and Detection of Building Objects in Point Clouds – Manual, version 14.11a.

Tridicon CityModeller, Tridicon CityModeller – Automatic Generation of 3D City Models (LoD2) – Manual, version 15.02.

Tridicon Export, Tridicon Export – Export if the data format op3d/tdc – Manual, version 15.02.

Tötterström, S. 2010. Katsaus VRS-teknologian nykytilaan ja tulevaisuuteen. Maankäyttö 3/2010. pp. 9-13.

VirtualcitySYSTEMS. 2015. BuildingReconstruction 2015 Manual. VirtualcitySYSTEMS GmbH.

Vosselman, G. & Dijkman, S. 2001. 3D building model reconstruction from point clouds and ground plans. International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences. Vol. 34, part 3/W4. pp. 37-44.

Vosselman, G. & Maas, H-G. 2010. Airborne and terrestrial laser scanning. Whittles Publishing, Scotland. p. 318. ISBN 978-1904445-87-6.

Wagner, D., Wewetzer, M., Bogdahn, J., Alam, N., Pries, M. & Coors, V. 2013. Geometric-semantical consistency validation of CityGML models. In Progress and new trends in 3D geoinformation sciences. pp. 171-192.

Wang, C. & Hsu, P. 2007. Building detection and structure line extraction from airborne lidar data. Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 37, part B3b. pp. 365-379.

Wang, C. Y. & Zhaoa, Z. M. 2008. Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 37, part B3b. pp. 203-208.

Wang, J. & Shan, J. 2009. Segmentation of LiDAR point clouds for building extraction. In American Society for Photogrammetry and Remote Sensing. Annual Conference, Baltimore, MD. p. 13.

WebGL, https://www.khronos.org/webgl/ [Accessed: 10.7.2015].

Wehr, A. & Lohr, U. 1999. Airborne laser scanning – an introduction and overview. ISPRS Journal of Photogrammetry & Remote Sensing. Vol. 54:2-3. pp. 68-82. DOI: 10.1016/S0924-2716(99)00011-8. ISSN: 0924-2716.

Wei, S. 2008. Building boundary extraction based on lidar point clouds data. Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 37, part B3b. pp. 157-161.

X3D, http://www.web3d.org/x3d/what-x3d [Accessed: 11.7.2015].

Zeng, Q. 2008. Data filtering and feature extraction of urban typical objects from airborne lidar point cloud. Proceedings of the International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. Vol. 37, part B3b. pp. 153-156.