DIPLOMARBEIT

3D CITY MODELING: A COMPARATIVE STUDY OF THREE APPROACHES

Ausgeführt am Department für
Geodäsie und Geoinformation
der Technischen Universität Wien

unter der Anleitung von
Univ.Prof.Dipl.-Ing. Dr. Josef Jansa

durch

Nilüfer Çipa

Blumauergasse 13/7
1020 Wien, Österreich

14.02.2017

ACKNOWLEDGMENTS

First of all, I am grateful to my supervisor Univ.Prof.Dipl.-Ing. Dr. Josef Jansa for his constant support throughout the time. He gave me freedom to develop my own ideas, reviewed and criticized my work, and encouraged me to finish this thesis.

I would like to extend my gratitude to DI Dr. Gerald Forkert and Mag. Samuel Cief at UVM Systems GmbH, who have given me great help with data, software, support, and expertise. Sincere thanks are given to Dipl.-Ing Felix Puls at Leica Geosystems for the providing the software packages.

And last but not least, I would like to acknowledge friends and family who supported me during my time. First and foremost I would like to thanks to my parent, Suat and Gülcan Cipa and my sister, Nazmiye Akoglu for their constant love and support. I am lucky to have met Matthias Hauck here, and I thank him very much for his friendship and unyielding support. I owe thanks to my boyfriend, Lukas Plautz and my best friend Gülin Uymaz, who are always there for me. This accomplishment would not have been possible without them.

LIST OF ABBREVIATIONS

ALS – Airborne Laser Scanning

BMP – Bitmap file format for raster graphics (introduced by Microsoft)

CAD – Computer-aided design COLLADA - Collaborative Design Activity (file-format)

DSM – Digital Surface Model

DTM – Digital Terrain Model

DWG – The native file format for AutoCAD

DXF – Drawing Exchange Format for CAD data file

GIS – Geographic information system

GML – Geography Markup Language

GPS – Global Positioning System

INS – Inertial navigation system

LAS – Point Cloud Format

LIDAR – Light Detection and Ranging, equivalent to laser scanning

LOD – Levels of Detail

OBJ – Wave front Object (file-format)

OGC – Open Geospatial Consortium

RAW – A camera raw image file

SHP – Esri Shapefile

TLD – Tessellated Image Format used by Zeiss Scanners

TIFF – Tagged Image File Format

TIN – Triangulated irregular network

VRML – Virtual Reality Modeling Language

VIT– Bitmap image (VITec scanner raster format)

XML – Extensible Markup Language

ABSTRACT

3D city models have turned out to be an important dataset over the past few years and have been widely used in various applications such as city planning, traffic control, disaster management and so forth. Efficient visualization of 3D city models in different levels of detail (LODs) is one of the pivotal technologies to support these applications. Today there exist a number of software tools with different levels of features that make creating 3D city models possible, which in turn can be used for numerous applications within the fields of visualization, communication, and analysis. Each software shows different strengths and weaknesses. This thesis attempts to test and compare the performance and capability of a selection of software packages in 3D City Modeling using real world spatial data.

The individual modeling procedures for each software, which are necessary for the generation of 3D city model, are shown and assessed in detail. The three software packages are assesed in respect to the resulting model generation, the amount of manual effort required and the time consumed as well as the easiness of handling the software. The following software packages have been evaluated: CityGRID® Modeler from UVM Systems, SketchUp from Trimble and 3D Editor from Hexagon. The outcomes of this thesis besides model generation provides a thorough and extensive comparison of existing tools for generation of 3D city models.

KURZFASSUNG

3D Stadtmodelle wurden im Laufe der letzten Jahre zu wichtigen Datensätzen und werden vielfach in verschiedenen Anwendungsbereichen, wie etwa Stadtplanung, Verkehrsüberwachung, Katastrophen-Management und vieles mehr, eingesetzt. Die effiziente Visualisierung der 3D Stadtmodelle in unterschiedlichen Detailierungsgraden (Levels of Detail = LODs) ist eine der zentralen Technologien, um diese Anwendungsbereiche zu unterstützen. Heutzutage gibt es eine Reihe von Computerprogrammen mit unterschiedlichen Merkmalen, welche für die Erstellung von 3D Stadtmodellen verwendet werden können, die dann wiederum für eine Vielzahl an Anwendungen in den Bereichen Visualisierung, Kommunikation und Analyse genutzt werden. Jedes Programm zeigt andere Stärken und Schwächen. Diese Arbeit versucht die Leistungsfähigkeiten und die Möglichkeiten einer Auswahl von Programmen für 3D Stadtmodellierung zu testen und zu vergleichen, indem reale räumliche Daten eingesetzt werden.

Die einzelnen Vorgehensweisen für die Modellierung, welche für die Erstellung von 3D Stadtmodellen notwendig sind, werden für jedes der Programme gezeigt und detailliert beurteilt. Die drei Computerprogramme werden in Bezug auf die erhaltenen Ergebnisse, auf den notwendigen Arbeits- und Zeitaufwand und schließlich auf die Handhabbarkeit evaluiert. Die folgenden drei Programme wurde beurteilt: CityGRID von UVM Systems, SketchUp von Trimble und 3D Editor von Hexagon. Neben der Modellentwicklung werden in dieser Arbeit verschiedene Programme zur Erstellung von 3D Stadtmodellen einem gründlichen Vergleich unterzogen.

Contents

ACKNOWLEDGMENTS III

LIST OF ABBREVIATIONS IV

ABSTRACT V

CHAPTER 1 1

1. INTRODUCTION 1

1.1. Background 1

1.2. Scope of the research 1

1.3. Relation to other projects 2

CHAPTER 2 3

2. FUNDAMENTALS OF 3D CITY MODEL 3

2.1. Definition 3

2.2. Level of Detail 5

2.3. Generation Methods 7

2.4. Data Formats 11

2.5. Application of 3D City Models 13

CHAPTER 3 17

3. SOFTWARE PACKAGES FOR 3D CITY MODELING 17

3.1 Tridicon® 3D Editor 17

3.2 CityGRID® Modeler 22

3.3 SketchUp by Trimble 27

CHAPTER 4 35

4. STUDY AREA AND DATA DESCRIPTION 35

4.1 Study Area 35

4.2 Data 35

CHAPTER 5 39

5. EXEMPLARY 3D CITY MODELING 39

5.1 3D City Modeling with 3D Editor 39

5.2 3D City Modeling with CityGRID® Modeler 58

5.3 3D City Modeling with SketchUp 70

CHAPTER 6 81

6. RESULTS AND DISCUSSION 81

6.1 Comparison of Basic Working Principle 83

6.2 Comparison of Import & Export Capacity 83

6.3 Comparison of Modeling Process 84

6.4 Comparison of Required Time 89
6.5 Comparison of Easiness of software 89
6.6 Comparison of Texturing Process 89
6.7 Comparison of Updating Process 90
CHAPTER 7 93
7. CONCLUSIONS 93
8. REFERENCES 95
APPENDIX A: Supported Object Type by Tridicon® 3D Editor 98

CHAPTER 1

1. INTRODUCTION

1.1. Background

Recent studies have shown that “54 percent of the world’s population lives in urban areas, a proportion that is expected to increase to 66 percent by 2050. Projections show that urbanization combined with the overall growth of the world’s population could add another 2.5 billion people to urban populations by 2050” (News article based on the World Urbanization Prospects – 2014 Revision by the UN Department of Economic and Social Affairs, United Nations 2014: http://www.un.org/en/development/desa/news/population/world-urbanization-prospects-2014.html). Urban areas are continuously changing due to construction and extension of features such as buildings (Morgan & Habib, 2002). Hence, it is critical to develop efficient techniques to assist the management of modern cities, which requires a platform that is capable of integrating city information from different resources.

Most of the current urban developments are based on using 2D mapping models. However, interest in 3D models is rapidly increasing for various application such as urban planning and design, pre-visualisation of new developments and utility service planning, microclimate investigation, telecommunication, noise simulation (Zhou et al., 2004), urban regeneration, tourism and visualisation of the urban environment (Baltsavias & Gruen, 2003; Vosselman, 2002). Stadler and Kolbe define (2007) a 3D City Model as a digital representation of Earth’s surface and its related spatial objects within urban areas such as building, tree, vegetation and some other manmade features. The realization of 3D city model brings the user considerable new advantages such as moving interactively within the model, creating animation or performing the analysis. Many companies utilize 3D city models to attract more users by enhancing expressiveness and simplifying the operations of their applications (Mao, 2011). Research on 3D city models is becoming a hotter topic in both academia and industry. Many algorithms and methods are developed to deal with challenges of 3D city models, and efficient visualization is one of the most critical issues.

The automatic reconstruction of urban 3D models became an important part of photogrammetric research almost two decades ago (Grün et al., 1995, 1997).The recent development of software packages makes the generation of 3D city models possible. However, there is still need for improvement of existing software in order to reduce the human interaction and increase the accuracy. This master thesis reviews a selection of software packages by carrying out an exemplary 3D city model in to give an overview of the current state of the art of 3D city model generation.

1.2. Scope of the research

In the context of the thesis, it is intented to get a comparative assessment (in a disinterested way) for the performance and capability of a selection of existing software packages for 3D city modeling using existing spatial data. In these days, there are plenty of methods and commercial software available in the market to create 3D city models. Each software has some advantages as well as limitations. The main purpose of this thesis is to find out the suitable software based on the need of 3D city model project. In order to achieve this, predefined basic tasks are to be carried out for each software, such as:

- modeling roof overhangs and related geometric properties above given building footprints,
- automatic and interactive texturing of roofs and facades.

It is also important to put together and compare the functionality of these software tools, such as the approaches to modeling; data requirements; import and export possibilities from or to different file formats; and eventually their strengths and limitations as far as certain tasks are concerned. The following software packages will be compared, and their performance must be analyzed:

- CityGRID® Modeler by UVM Systems,
- SketchUp® by Trimble and
- Tridicon® 3D Editor and TextureMapper by 3DCon GmbH , a subsidiary company of Hexagon Europe

The evaluation process will take into consideration both the resulting models as well as the process itself. The outcomes of this thesis besides model generation provides a thorough and extensive comparison of existing tools for generation of 3D city models.

1.3. Relation to other projects

The aim of the thesis is to give a comparative assessment for 3D city model software packages. Some related works are given below:

Bratfisch et al. (2014) explained a comparative study of different software packages based on the 3D model of city landmarks. In their study, four software tools have been assessed: CityEngine from Esri, CityGRID Modeler from UVM Systems, SketchUp from Trimple and tridicion Architecture from 3DCon GmbH.

A good comparative study of different image-based approaches for 3D city models can be found in Singh et al.`s study (2014). In the study, four image-based approaches have been defined and compared. Those approaches are Sketch-based modeling, Procedural grammar based modeling, Close range photogrammetry based modeling

Rothe and Janne (2009) explained the usage of the 3D Models by comparing tridicon® CityDiscoverer und Autodesk® LandXplorer software packages.

Qiming and Wenjiang (2004) presented a preliminary review on three-dimensional city model. This study was based on the comparison of three 3D data model of GIS: 3D Formal Data Structure (3D FDS), a hybrid data structure (V3D) and objected-oriented geo-database kernel system (GeoToolKit).

CHAPTER 2

2. FUNDAMENTALS OF 3D CITY MODEL

The purpose of this chapter is to establish the basis of knowledge of 3D model representation, generalization, and visualization.

2.1. Definition

Computer Graphics utilize geometrical information such as position, shape, and size of an object that is stored in the computer. The product of the process is called 3D model. There are three common types of 3D models, (i) wireframe model that stores edge information, (ii) surface model that stores surface information, and (iii) solid model that identifies inside and outside of volumes (Huang, 2008).

3D city models represent spatial and geo-referenced urban data by means of 3D geovirtual environments that basically include terrain models, building models, vegetation models as well as models of roads and transportation systems (Döllner et al., 2006). There are different expressions used for 3D city models such as “Virtual City,” “Cybertown,” “Cybercity,” and “Digital City” (Sadek et al., 2002).

Theoretically, all objects located within a city can be integrated into a 3D city model if they are capturable with any method. However, the most important objects of 3D city model are constructions (i.e. buildings, landmarks), roads, vegetation, and terrain. Depending on the application type, a 3D city model can consist of geometry (i.e. location, shape), texture (i.e.colors, images), and semantic information.

Building Reconstruction

The buildings are the most important component in a 3D city model. There are two main paradigms for reconstruction of 3D building models: Model-driven and Data-driven. Model-driven modelling approach consists of searching the most appropriate model among basic building models contained in a models library (Tarsha-Kurdi et al., 2007). A model library consists of building primitives. Building primitives are a number of building models with different roof structure that can be represented by a set of parameter or topographical expression (Figure 2.1)

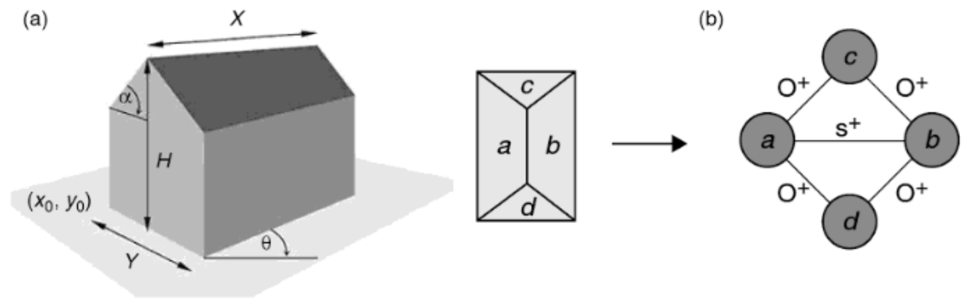


Figure 2.1: Different expression of building primitives (a) Parameter definitions (Image source: Maas, 1999) (b) Topographical &Topological definition (Image source: Verma et al. 2006), (Heritage & Large, 2009)

The data-driven approach usually assumes that a building is a polyhedral model. It analyzes the building point cloud as a unity, without relating it to a set of parameters (Tarsha-Kurdi et al., 2007). The most common method is to detect various segments of planes and group them all the segments that have similar features (e.g. slope, orientation) (Heritage & Large, 2009). For instance, Schwalbe et al. (2005) use the data-driven approach in ALS data to generate 3D buildings. Firstly, it detects the line in 2D in order to find the lines that represent roof faces (planes) and then combine the individual planes to model roof structure (Figure 2.2).

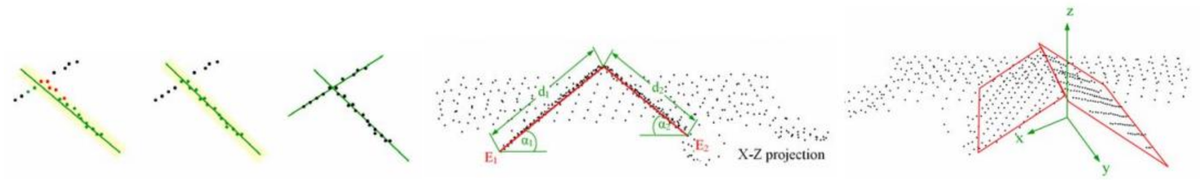


*Figure 2.2: 2D Line extraction (left), Lines representing roof faces (middle), Extracted roof planes (right), (*Image *source: Schwalbe et al., 2005)*

Building Texturing

Aside from a correct geometric generation, the texture reconstruction of the buildings has become an essential component of 3D city modeling for better visualization. Besides resulting in a photo-realistic look, texture also creates the false impression of a higher level of geometric detail, a fact that is exploited in image-based rendering (Frueh et al., 2004).

3D buildings can be textured by using synthetic or digital images (i.e. aerial, terrestrial). Synthetic textures are artificial generated textures that are often stored in texture library of the software packages. They may be just solid colors or certain pattern like stone, brick or fences (Figure 2.3).

The actual texturing of the buildings can be achieved with the use of acquired imagery (Figure 2.3). Roof surfaces are often textured by using aerial images. For the texturing of the building facades, aerial images may not always be optimal as all side of the buildings are sometimes not visible in the images. If the aerial images are used for the texturing of the facade, it´s necessary to have multiple images from different angles so that the buildings facades can be visible in several images. Besides, that oblique aerial images or terrestrial images can be used to texture the building facades.

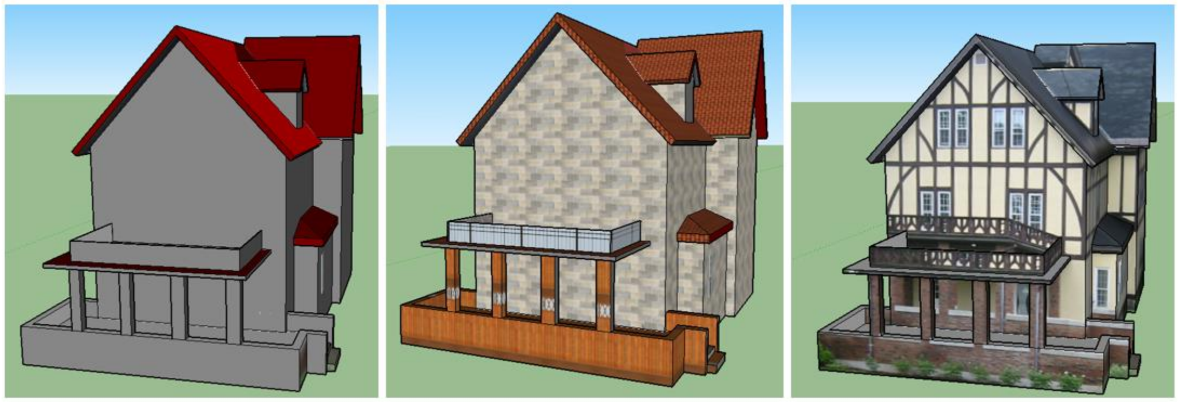
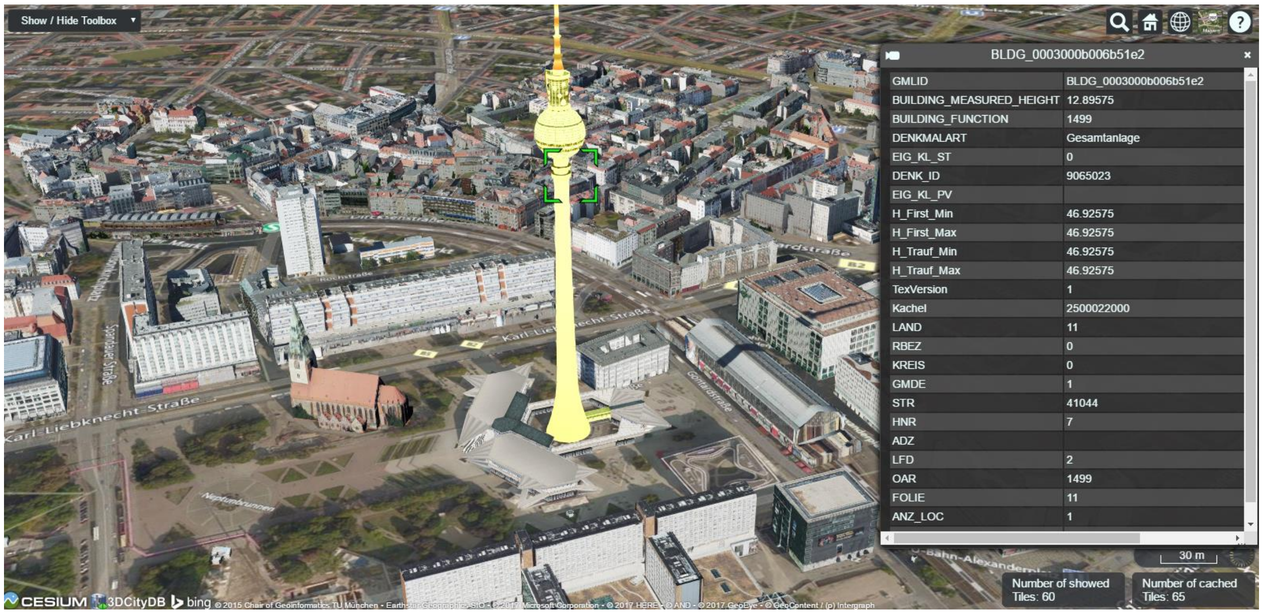


Figure 2.3: Different texture types: Synthetic texture, solid color (left), Synthetic texture with different material patterns (middle), Texture with digital image (right)[1]

Semantic Modelling

Semantic information can be described as any information that is not visualized as geometry or texture, for example, attributes of objects such as the use of a specific building (Uggla, 2015). In figure 2.4, a 3D city model of Berlin can be seen, which provides semantic information besides the geometry.



BLDG_0003000b006b51e2

GMLID	BLDG_0003000b006b51e2
BUILDING_MEASURED_HEIGHT	12.89575
BUILDING_FUNCTION	1499
DENKMALART	Gesamtanlage
EIG_KL_ST	0
DENK_ID	9065023
EIG_KL_PV	
H_First_Min	46.92575
H_First_Max	46.92575
H_Trauf_Min	46.92575
H_Trauf_Max	46.92575
TexVersion	1
Kachel	2500022000
LAND	11
RBEZ	0
KREIS	0
GMDE	1
STR	41044
HNR	7
ADZ	
LFD	2
OAR	1499
FOLIE	11
ANZ_LOC	1

Figure 2.4: A semantic 3D city model of the German capital Berlin (http://www.3dcitydb.org/3dcitydb/VisualizationBerlin/,last access December 2016)

2.2. Level of Detail

The level of detail (LOD) of a 3D city model is one of its most important characteristics. It denotes the adherence of the model to its real-world counterpart, and it has implications on its usability (Biljecki et al., 2014). The CityGML 2.0 standard from the Open Geospatial Consortium

[1] The building model in the figure is freely downloadable via https://3dwarehouse.sketchup.com/model.html?id=82e4379f93e135cf3d02d646860fb5fe; last access February 2017

(2012) classifies digital city model by level of detail (LOD) regard with both geometry and semantics. There are five different categories of LODs from LOD0 to LOD4 (Figure 2.5).

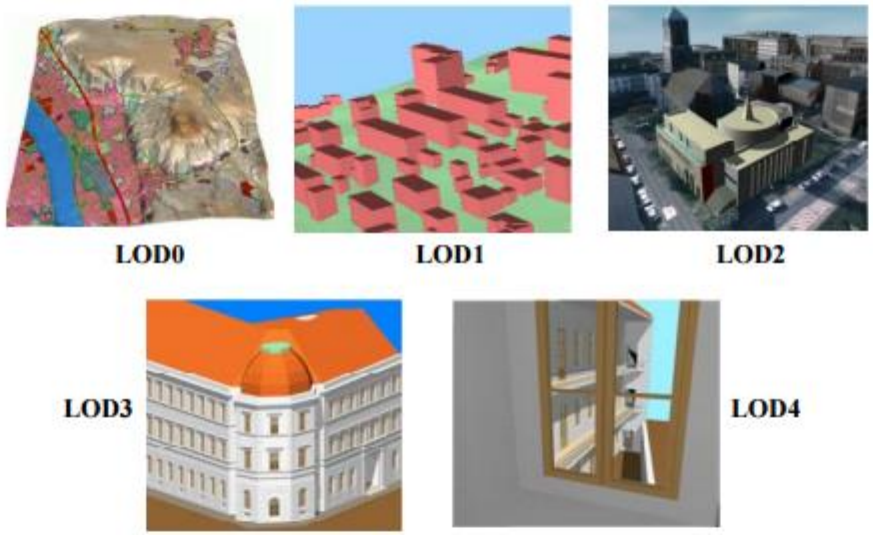


Figure 2.5: Level of Detail from LoD0 to LoD4 (Kolbe, 2005)

LOD0 is essentially a 2.5D Digital Terrain Model (DTM) over which an aerial image or a map may be draped and is separated from LOD1-4 (Figure 2.6) as it does not contain any separately modeled buildings. The building is represented by either the building footprint or the roof outline. LOD0 is used for regional and landscape applications. LOD1 is a volume that can be distinguished as a building with flat roof structures (without any semantically structuring). This level is used for both city and region coverage. In contrast, LOD2 is a model with a simplified roof shape, and where the object's parts can be modeled in multiple semantic classes (e.g. roof, wall). This level is used for city and region coverage. LOD3 are architectural models with detailed wall and roof structures potentially including doors and windows. LOD4 completes an LOD3 model by adding interior structures for buildings.

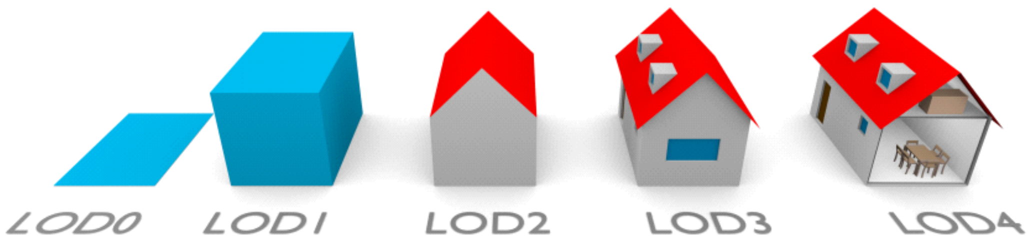


Figure 2.6: The five LODs of CityGML 2.0. The geometric detail and the semantic complexity increase, ending with the LOD4 containing indoor features.

Each level of detail necessitates a different type of data collections. Models with LOD0 to LOD2 can be produced from aerial images whereas interior structures, for example, depend on the availability of either derailed surveys or architectural document (Navratil et al.,2010).

2.3. Generation Methods

A number of generation methods are available for 3D city models related to the available dataset. Depending on the application, budget and accuracy requirements, different data types and technologies can offer the right solution (Marre, 2011). Singh et. al(2013) categorized the 3D city modeling into the following approaches:

Based on Automation:

- Automatic
- Semi-automatic
- Manual

Based on Data input techniques:

- Photogrammetry based methods
- Laser Scanning based methods

Photogrammetry based methods for 3D City Model generation:

- Aerial Photogrammetry based model
- Satellite Photogrammetry based model
- Close Range Photogrammetry based model

LASER scanning based model:

- Aerial Laser based model
- Terrestrial laser based model

Hybrid methods:

- The combination of these methods is also a method to create virtual 3D City model.

Today, the most commonly used data acquisition methods for the generation of 3D city models are aerial photography and airborne laser scanning. Those methods will be further explained in the next section.

2.3.1 Laser-Based Method

Airborne Laser Scanning or also known as LIDAR has been used for various 3D mapping application including 3D city modeling project for over two decades. LIDAR mapping is an accepted method of generating precise and directly georeferenced spatial information about the shape and surface characteristics of the real world (Carter et al., 2012).

The main units of LIDAR systems are a global positioning system (GPS), an inertial navigation system (INS), GPS base station and laser scanning system (Figure 2.7).

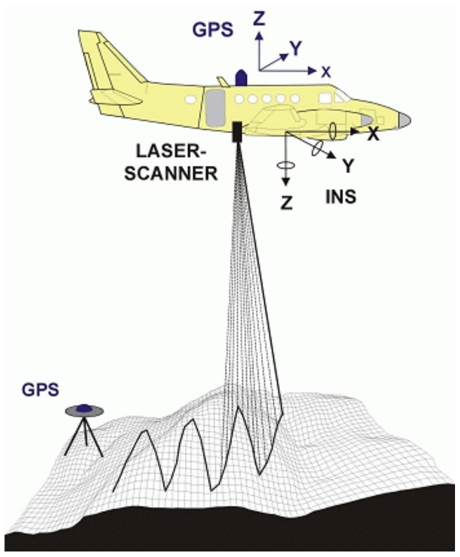


Figure 2.7: LIDAR system component (Stadtvermessung Wien)

The position and orientation (roll, pitch, and heading) of LIDAR sensor are ensured by INS and GPS. The laser scanning system is installed on a platform (e.g. plane, helicopter, satellite) and emits pulses to earth´s surface. The pulses are reflected off the earth´s surface or objects on it and return to the receiver. High-speed counter measures the time of flight from the start pulse to the return pulse (Moskal, 2008). By accurate time measurement, the range between sensor and target can be calculated using following formula:

$$R = \frac{v.t}{2}$$

R is the range, *v* is the speed of light and t is the time measurement. Combined with position and orientation of the laser sensor, the range between sensor and target can be transformed to the precise three-dimensional coordinates of the target.

Characteristics of LIDAR Data

An emitted laser pulse may return to LIDAR sensor as one or multiple returns (Figure 2.8).

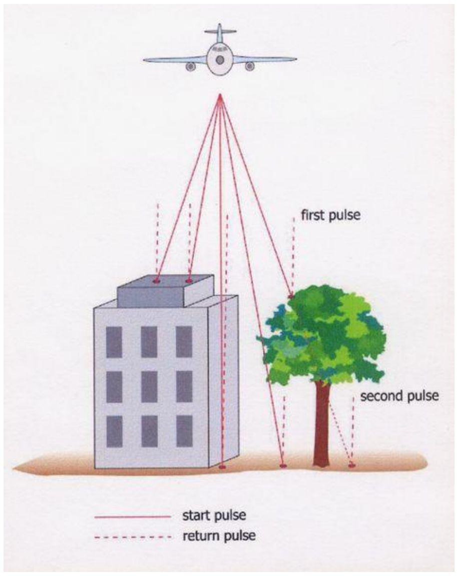


Figure 2.8: LIDAR Laser pulse return
(http://www.coastalwiki.org/wiki/Use_of_Lidar_for_coastal_habitat_mapping)

The first returned pulse (first echo) is the highest feature above the ground surface and it can be used to create digital surface model (DSM), whereas last pulse (last echo), in general, returns from impenetrable surface and it makes creation of digital terrain model (DTM) possible (Figure 2.9). An emitted pulse can be recorded as separate pulse or as one continuous wave, which is called full-waveform.

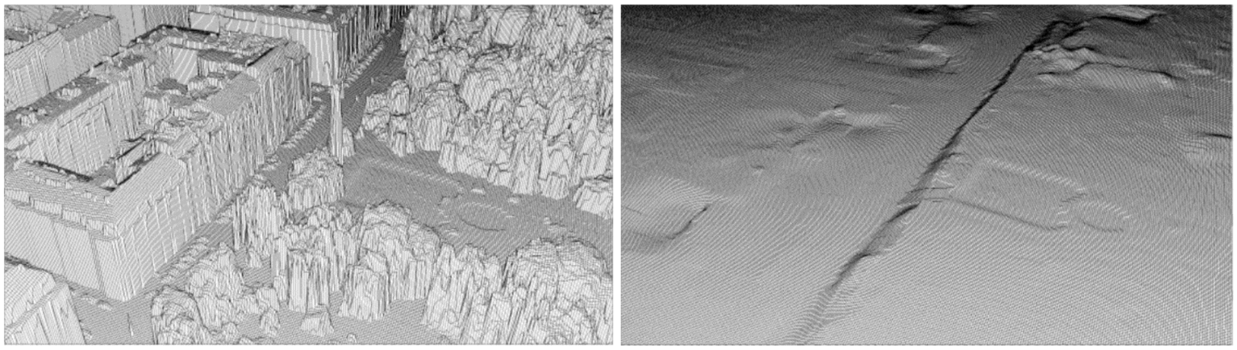


Figure 2.9: DSM computed from all LIDAR point (0.5m) (left), Final DTM (0.5 m) (right) (Rottensteiner, 2002)

Besides recording the time of a return pulse, LIDAR systems record the intensity, or the magnitude, of the return pulse; in other words, they not only measure the fact that there is a return pulse, but they also measure the strength of the return (Schuckman, 2014).

2.3.2 Photogrammetry-Based Method

"Photogrammetry is the art, science, and technology of obtaining reliable information about physical objects and the environment through processes of recording, measuring and interpreting photographic images and patterns of recorded radiant electromagnetic energy and other phenomena" (Wolf and Dewitt, 2000). Photogrammetry requires overlapping images from a different position in order to determine the size, shape and the position of the object (Figure 2.10).

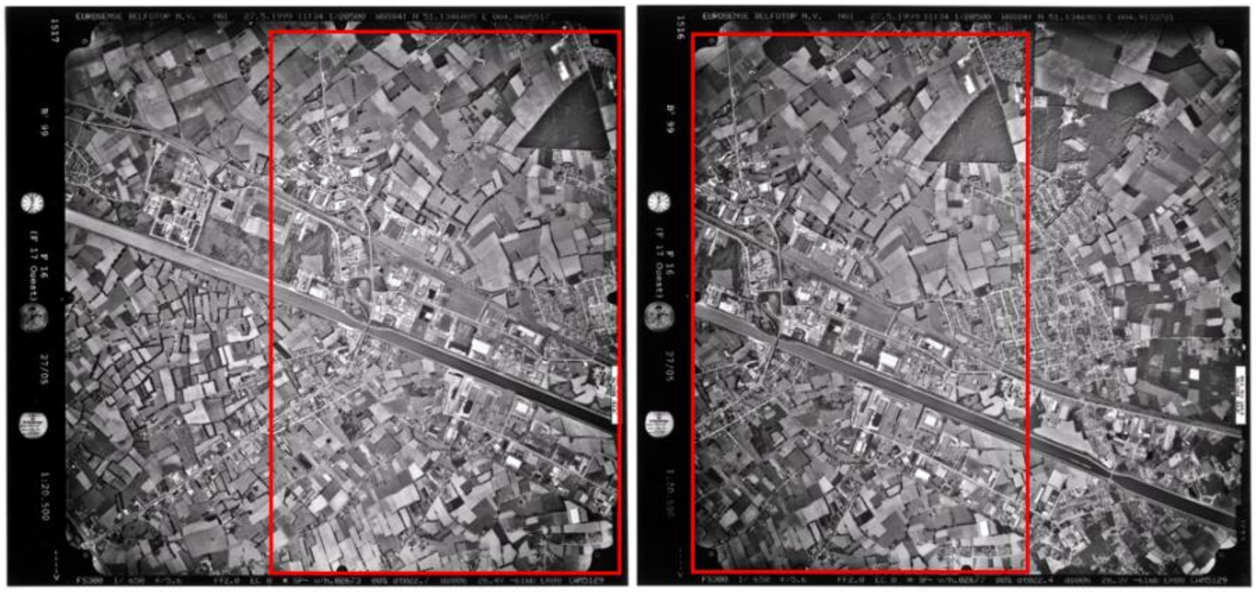


Figure 2.10: Stereo pair of Tongerlo, Belgium, 17-05-1999. The red rectangles indicate overlapping areas (Source: http://www.seos-project.eu/modules/3d-models/3d-models-c02-p04-s01.html, last access January 2017)

Figure 2.11, below, shows the image-taking process of aerial photogrammetry. Images must have an overlapping between consecutive images. Through the mathematical and geometrical relations between the overlapping images, the position of the object can be derived.

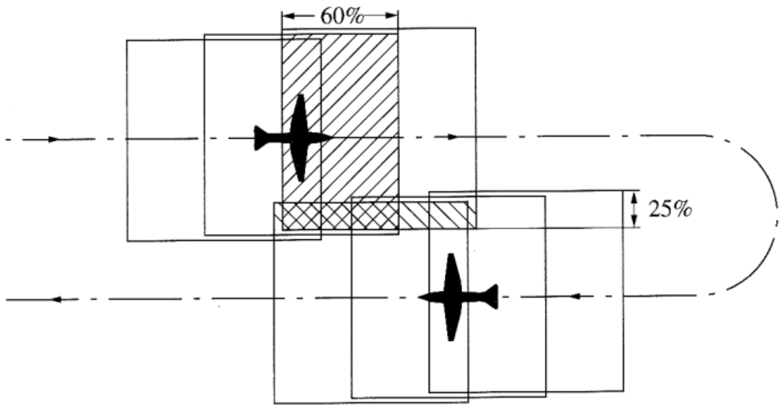


Figure 2.11: Aerial photography from above. Strips indicate the overlapping areas. (Source: http://www.seos-project.eu/modules/3d-models/3d-models-c02-p04-s01.html, last access January 2017

Figure 2.12 illustrates the principle of photogrammetry. 3D ground coordinates of point P can be calculated from 2D image coordinates of p1 and p2. To do so, point P must be visible at least in two images and exterior, and interior orientation of the camera must be known. The exterior orientation parameters indicate the location and rotation of the camera in space at the time of exposure, whereas interior orientation parameters define the internal geometry of the camera such as principle point, focal length, projection center. With knowledge of interior and exterior parameter, 3D coordinates of an object in space can be calculated through the collinearity equation, as a focal point, image point and imaged object lie on the same line. As indicated in Figure 2.12, the vector from focal point O_1 to imaged object P and image point p_1 to imaged object P are collinea, as well as the vector from O_2 to P and p_2 to P. In addition to this, coplanarity applies.

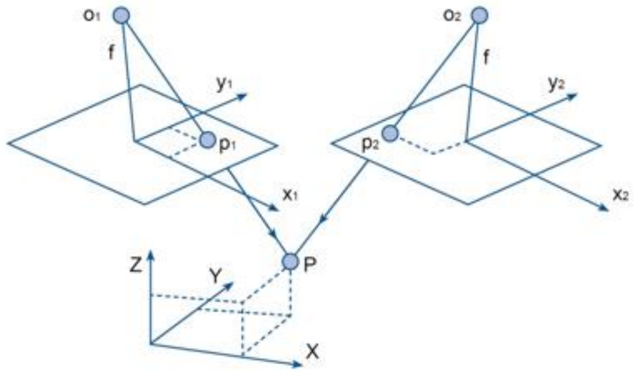


Figure 2.12: Principle of photogrammetry

The images for the photogrammetric measurement can be taken from a ground based stations so called (terrestrial photogrammetry) or airborne platforms (aerial photogrammetry). In the frame of 3D city modeling, the main advantages of the aerial-based method are to make possible to cover large areas using fewer images and in a shorter time. For instance, A 3D model of the Turkish city Konya has been generated by 42000 oblique aerial images which cover over 300000 buildings (Ozerbil, 2015). The terrestrial acquisition has the major advantage for LOD3 and LOD4 projects and texturing.

2.4. Data Formats

In this chapter, the of most commonly used 3D city model data formats have briefly been introduced.

CityGML

CityGML is a common information model and XML-based encoding for the representation, storage, and exchange of digital 3D city and landscape models (Kolbe, 2012). It was proposed in August 2008 by the Open Geospatial Consortium (OGC) which is the most important standardization in the field of geospatial information technologies.

CityGML provides a model and mechanism for describing 3D objects with respect to their geometry, semantics, and their relations. Therefore, it supports 3D content for visualization but goes far beyond that point to support manifold analytical capacity. Unlike the Keyhole Markup Language (KML) used in the context of Google Earth, Collada or X3D, for instance, it distinguishes real world features providing 98 classes with 372 well-defined attributes in total. These classes may have geometrical properties or not. (Löwner et al., 2013For instance, the different classes of a building object in CityGML can be seen in figure 2.13. The classes differentiate depending on the level of detail of the model.

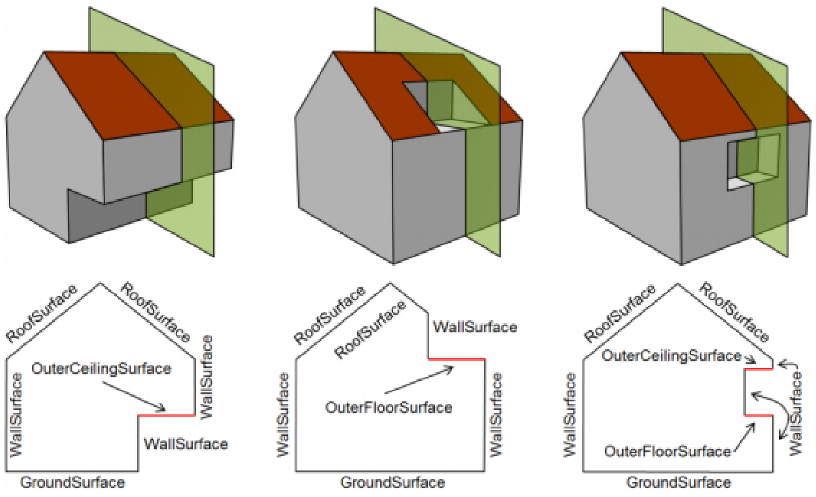


Figure 2.13: Examples of boundary surfaces for a building in LOD2 in CityGML (Löwner et al., 2013)

CityGML is highly scalable (extensible with respect to a theme) through CityGML Application Domain Extensions (ADE), and datasets can include different urban entities supporting the general trend toward modeling not only individual buildings but also wider sites, districts, cities, regions, and countries (Biljecki, 2013).

KML/KMZ

Keyhole Markup Language (KML) is a standard format recognized by OGC and developed by Google Inc. KML is a file format used to display geographic data in an Earth browser such as Google Earth. KML uses a tag-based structure with nested elements and attributes and is based on the XML standard (https://developers.google.com/kml/documentation/kml_tut, last access January 2017). KMZ is a compressed version of KML.

COLLADA

Collaborative Design Activity (COLLADA) is an exchange file format for 3D application, developed by Sony Computer Entertainment. The format is widely used by game developers.

VRML

Virtual Reality Modeling Language (VRML) a text file[2] format for representation of 3D objects. It´s the first ISO-standard format for 3D browsing. With a plugin, 3D objects in VRML format can be open in very internet web browser.

3DS

3DS is a binary file format, used by Autodesk 3ds Max (formerly 3D Studio MAX).

DXF

Drawing Interchange Format or Drawing Exchange Format (DXF) is a file format developed by Autodesk. DXF file format enables data interchange between AutoCAD and other software. DXF can be either ASCII or binary formats.

OBJ

Object files define the geometry and other properties for objects in Wavefront's Advanced Visualizer. Object files can also be used to transfer geometric data back and forth between the Advanced Visualizer and other applications (http://paulbourke.net/dataformats/obj/, last access November 2016). The file format is open and has been adopted by other 3D graphics application vendors.

2.5. Application of 3D City Models

3D models are employed in increasing number of application, and some of those key application with examples have been listed in this section.

Planning application

The most common use case of 3D city model is currently in the planning application. 3D city models are improving the practice of urban environmental planning and design and help planning authorities to illustrate explicit photo-textured information of what the city environment will look like after a proposed change (Sadek, 2005).

Disaster management and emergency response applications

LOD2 and LOD3 city model can be highly useful in a state of emergency, as the details of the building are known i.e. the location of the entries, balconies, etc. It is also helpful in the disaster management, as the realistic effect of the disaster can be simulated.

Energy demand analysis

3D city models aid to evaluating any kind of energy demand Previtali et al. (2014) note the use of 3D city models to assess the cost of retrofitting of a building. In combination with other data, 3D city models may be used for thermal assessment, and to determine thermal bridges and heat losses from the building envelope and estimation of potential for exploration of solar energy (Biljecki et al., 2015) (Figure 2.14).

2 Text file is a computer file that is structured as a sequence of lines of electronic text (https://en.wikipedia.org/wiki/Text_file, last access November 2016)

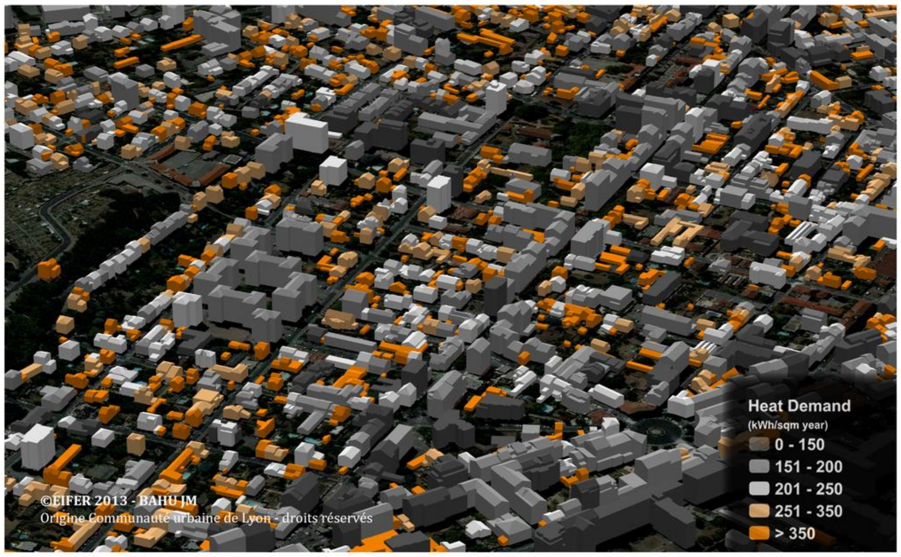


Figure 2.14: Results of the estimation of the heat demand of buildings (Bahu,2013)

Decision support system applications

3D city model can help planners and managers to answer informative decision as the result of the decision can be simulated using interactive models. As the example in Figure 2.15, a 3D noise simulation can be used to find out the most needed place to place noise barriers.

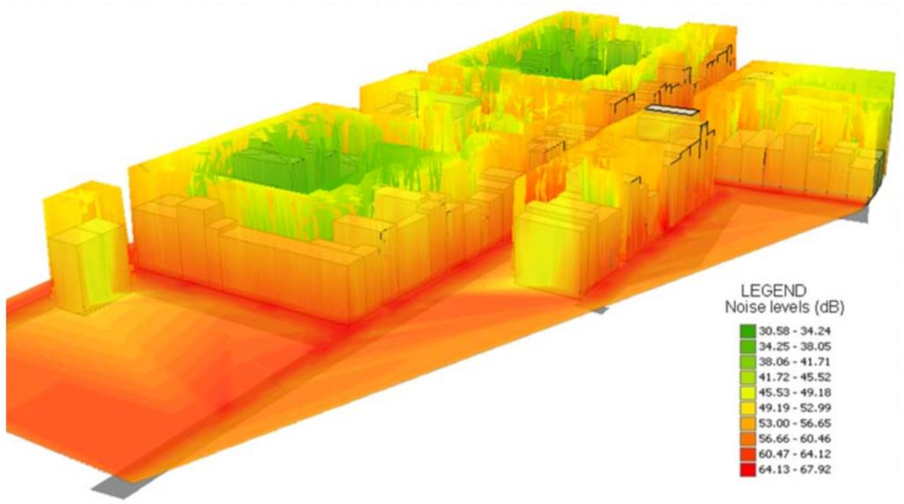


Figure 2.15:3D noise modeling using 3D City Model (Kurakula, 2007)

Navigational applications

Navigation system, which includes 3D city model, may enhance the visual depiction and help the driver to identify real environment easily (Figure 2.16).

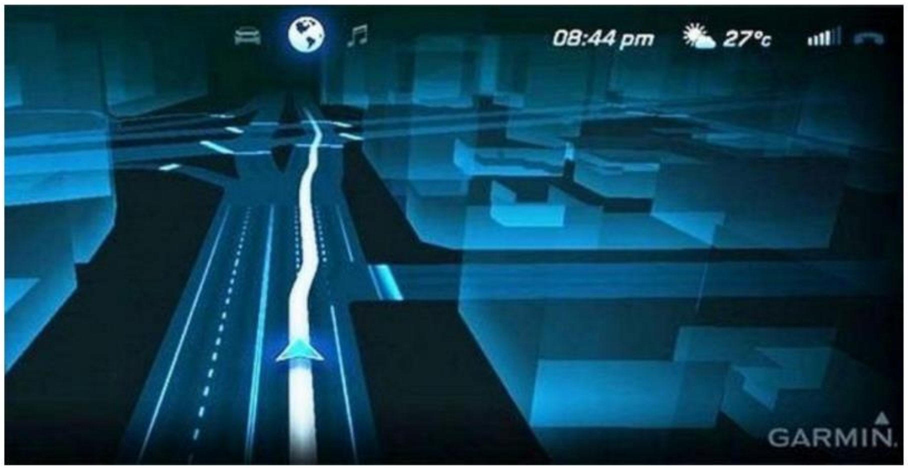


Figure 2.16: Demonstration of a 3D navigation system by Garmin Ltd

CHAPTER 3

3 SOFTWARE PACKAGES FOR 3D CITY MODELING

The aim of the thesis is to compare and assess different tools for 3D city models generation. For this purpose, three tools, which have diverse methods for model generation, have been chosen. In this chapter, the characteristics and functionalities of those tools have been described.

3.3 Tridicon® 3D Editor

Tridicon® 3D Editor is a software product developed by 3DCon GmbH (formerly by GTA Geoinformatik), a subsidiary company of Hexagon Europe[3]. The product offerings of tridicon® are quite comprehensive and provide solutions for building modeling, integration, and visualization of three-dimensional digital point clouds and further processing of point clouds for 3D City models and landmarks.

The program 3D Editor, which is aimed to be used in this thesis, is a digital photogrammetric workstation for aerial and satellite photogrammetry. The basic task of photogrammetry is the accurate mapping of the real world by deriving its 3D coordinates (X,Y,Z) from 2D coordinates (x,y) measured in overlapping aerial photographs. Basically, the 3D Editor is specifically designed for generating building models from aerial photographs. It is assumed that the orientations of the aerial images are known, for instance from an aerial triangulation carried out in advance, in order to provide the geometric relationship between the images and the real world. For the evaluations in this thesis, the software version 15.06 has been used.
In the real world, the shape of the buildings reveals a huge diversity. Thus the automated generalization of 3D city model from aerial images is still a challenging task. There are different approaches in order to simplify the automatical acquisition of 3D building models. One may be a restriction to a lower level of detail. In visualizations, the resulting lack of 3D details may, to a certain extent, be compensated by applying texture mapping techniques. Another approach may be the introduction of a catalog of permitted building shapes also known as building primitives. Both approaches reduce the number of shapes and details to be modeled. 3D Editor makes use of a predefined building primitives. Since each building is assumed to be limited by vertical walls along their boundaries, building primitives can be characterized by roof type, such as flat roof, lean-to roof, gable roof, tower, wall, etc. Each predefined object type has a constructional order which means the objects have a fixed order of nodes and straight connecting lines from the very beginning. Tough a new building has to be created by using the primitives, there is the possibility of augmenting already existing models by adding details such as dormers, chimneys, and other roof structures so that even more complex and refined building model can be created.

[3] Tridicon has become part of the new product line HxMap, the High-Performance Multisensor Workflow by Leica-Geosystems. In this context Tridicon 3D Editor and TextureMapper, as part of the module HxMap 3D Modeller Basic, has been renamed to HxMap 3D Editor and HxMap TextureMapper (see http://leica-geosystems.com/products/airborne-systems/software/leica-hxmap/leica-hxmap-3d-modeller-basic; last access November 2016).

User interface

The Graphic User Interface consists of the typical elements of a window-oriented software (Figure 3.1). The main window contains the title bar, menu, and toolbars, image windows and status bar. The title bar shows the name of the project as well as the name of an open data file. The largest part of the program window is occupied by image windows of the used images. Several images may be opened simultaneously, where each is displayed in its own image window. The status bar contains the X, Y, Z coordinates of the node, the selected layer (e.g. “Buildings”) and as well as the property of the selected objects (e.g. “hip roof”). The status bar is important when an object needs to be redefined. One can easily see which primitive has assigned a certain roof and one can start to convert the roof type to another desired roof type.

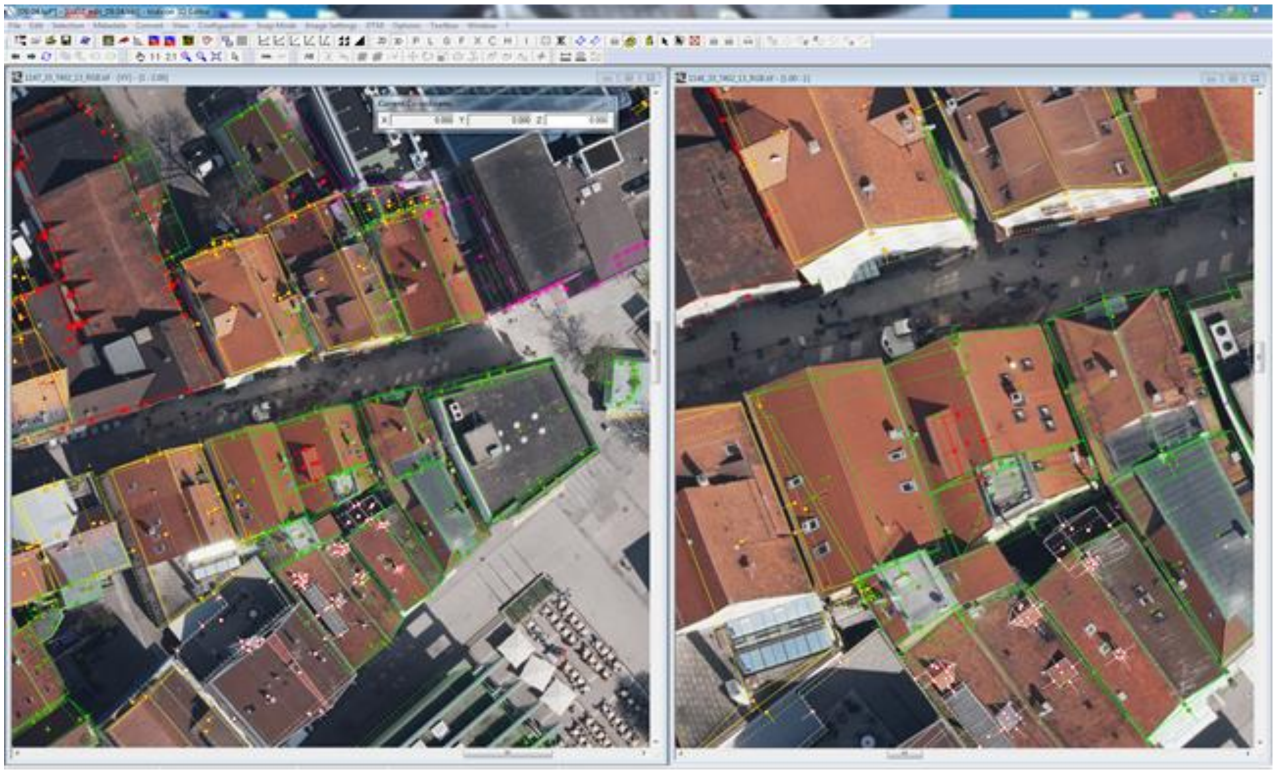


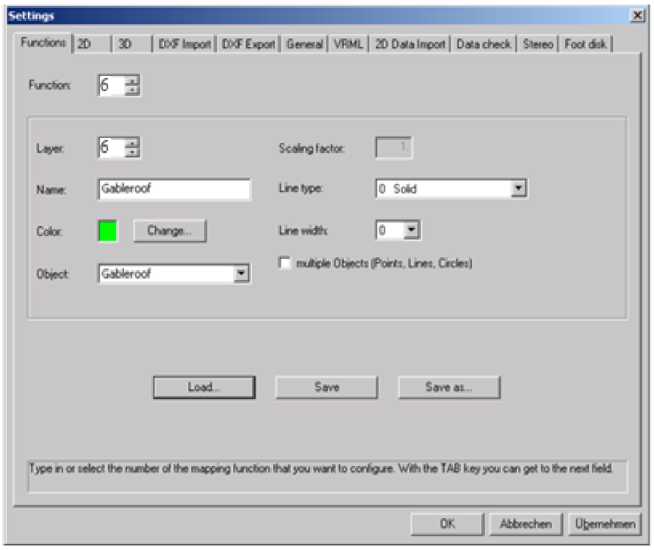
Figure 3.1: Main Window of the 3D Editor

Image Preparation

3D Editor can read the following image formats *vit*, *tiff*, *raw*, *tld* and *bmp*. It also supports automatic reading of orientation data. Therefore, the images and corresponding orientation data must be stored in the same directory. The required orientation parameter can also be inserted manually from the menu. The program also offers different options in order to enhance images by applying basic image processing algorithms, such as brightness and contrast enhancement, application of lookup tables and similar.

Mapping Functions

The predefined building primitives can be accessed by means of a list of mapping functions. For each mapping function, the user must, in addition to the number code, primarily predetermine the object type, its associated color, line type, and width. For instance, as shown in Figure 3.2, the object type "gable roof" has been assigned to mapping function 6. If a user wants to model an object, the mapping function must be activated via the number code in a toolbar. The mapping function toolbar can be seen in figure 3.3.



Function: 6

Layer: 6

Name: Gableroof

Color: 🟩

Object: Gableroof

Scaling factor: 1

Line type: 0 Solid

Line width: 0

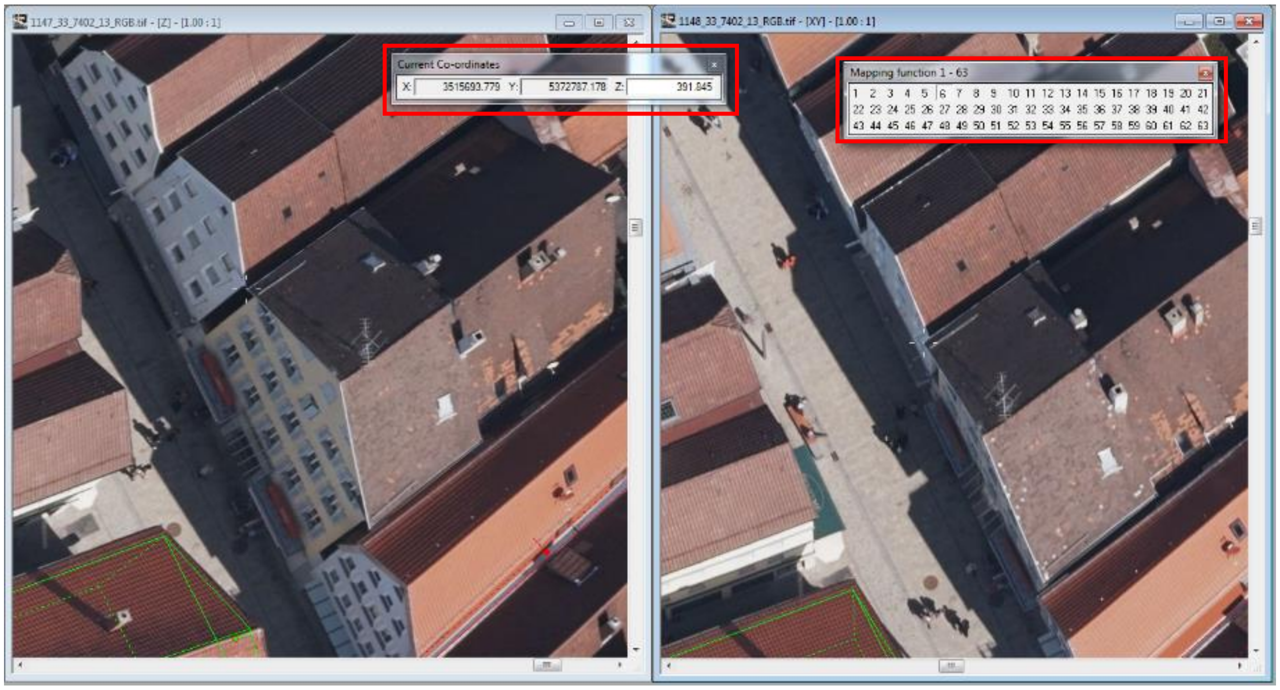
☐ multiple Objects (Points, Lines, Circles)

Type in or select the number of the mapping function that you want to configure. With the TAB key you can get to the next field.

Figure 3.2: Mapping function settings

Modeling a single building

Modeling of an object requires at least two images (Figure 3.3). To model an object, x, y image coordinates of the object must be measured in both images, and then the digital workstation applies the projection transformation in the background and calculates X, Y, Z coordinates which are displayed in Current Coordinates toolbar (Figure 3.3). In order to simplify the measuring process for the user, the measurement is split up into two step. As described by the 3D Editor's User Manual, first in one image the cursor is set onto the point to be measured (the so-called X,Y measurement according to the manual), and then in the other image by initiating the Z measurement the cursor is set onto the homologous point. In this way the respective point's 3D object coordinates X, Y, Z can be determined.



X: 3515693.779 Y: 5372787.178 Z: 391.845

Figure 3.3: Image windows and Mapping functions toolbar

The desired mapping function number must be activated in the toolbar before modeling starts. For instance, mapping function 6 („Gable roof“) is active in figure 3.4. Each object type has a constructional order, which can be seen by the operator in a line sketch where the object points are assigned an ordinal number. The user must follow this given order by measuring only the numbered points in the right sequence. Points without a number are created automatically thereby completing the respective building model.

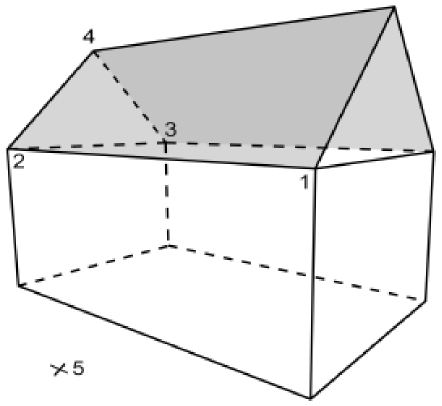


Figure 3.4. Constructional Order of Gable Roof

For better understanding, this measuring principle should be explained with the help of a gable roof, one of the 3D object types in the object catalogue of the 3D Editor (Figure 3.4). The footprint (i.e. eave line) of a gable roof is a rectangle, for whose construction only 3 points are required a fourth point is necessary to define the height and the central location of the ridge. The measurement starts at one corner of the eave line at Point 1. Point 2 is a neighboring corner on the eave line in the direction of the ridge. The corner point next to Point2, i.e. Point 3, is required for defining the other side of the rectangle. Now the rectangle can be completed automatically. Point 4 needs to be measured to define the ridge. Finally, Point 5 needs to be measured to define the building height. Figure 3.5 shows a practical example. As the just mentioned constructional order of the gable roof indicates, the modeling starts by choosing the first (eave point). Then the second point is measured parallel to the ridge, and the third point lies on the eave corner adjacent to the second point.

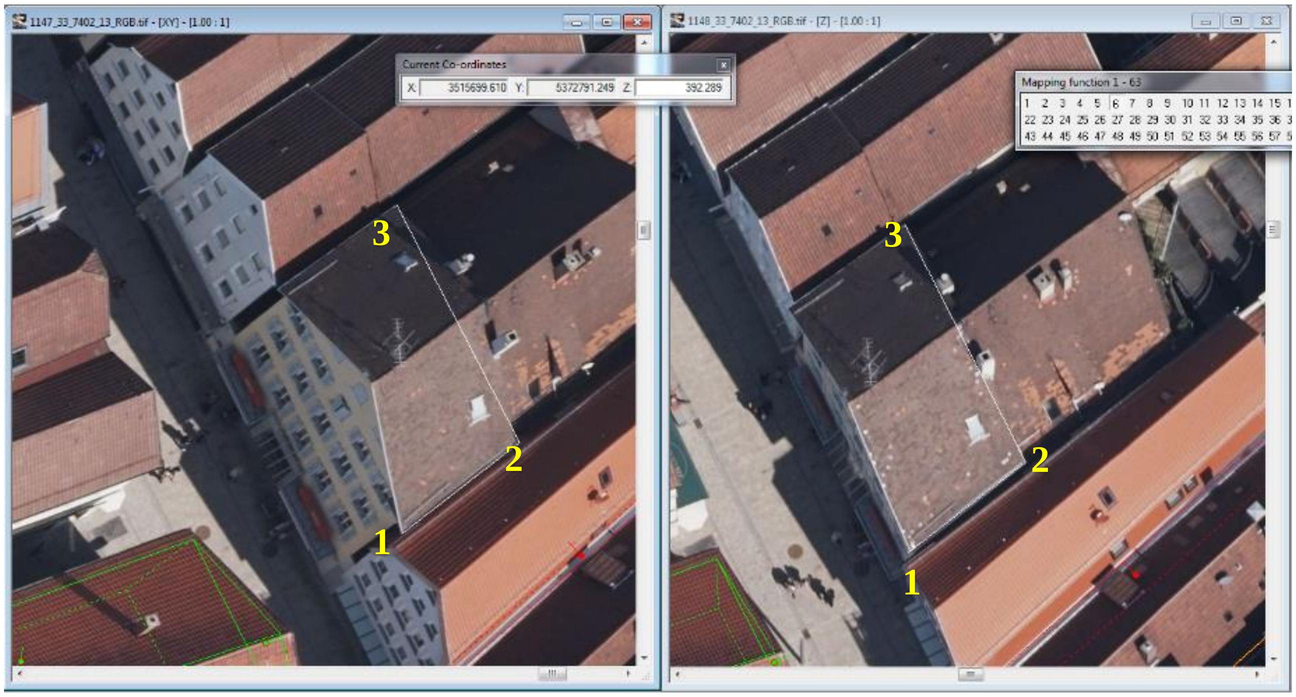

1147_33_7402_13_RGB.tif - [XY] - [1.00 : 1]
Current Co-ordinates
1148_33_7402_13_RGB.tif - [Z] - [1.00 : 1]
Mapping function 1 - 63
3
2
1
3
2
1
X: 3515699.610 Y: 5372791.249 Z: 392.289

Figure 3.5: Measuring eave points of object with respect to constructional order

The fourth eave point will be set automatically. The first ridge points need to be manually measured, and the other one will be measured automatically. The fifth point must be determined by help of the second image which can be easily accomplished by using the mouse wheel. After all points are measured as the constructional order regulates, the model looks as in Figure 3.6. If the model is not accurate enough, the position of the object can be changed by dragging the object or object part onto the correct position or moving the object or object part by a defined distance. In addition to this, there are other editing options such as changing only height, rotating, scaling or inserting a point.

The ground height can also be measured automatically from a digital terrain model (DTM). A direct display in a 3D window (City DiscovererLight[4]) allows the control of the model from all viewing directions.

[4] City DiscovererLight (by 3DCon) is a viewer for visualizing digital 3D city models

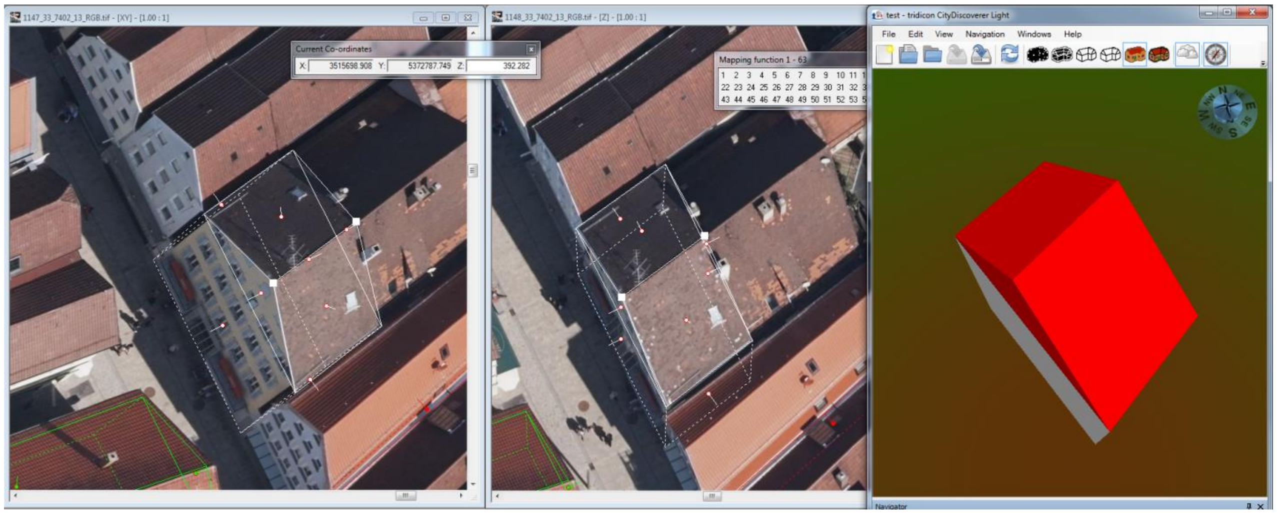

1147_33_7402_13_RGB.tif - [XY] - [1.00 : 1]
Current Co-ordinates
X: 3515698.908 Y: 5372787.749 Z: 392.282
1148_33_7402_13_RGB.tif - [Z] - [1.00 : 1]
Mapping function 1 - 63
test - tridicon CityDiscoverer Light
File Edit View Navigation Windows Help
Navigator

Figure 3.6: Model of a building with gable roof

3.4 CityGRID® Modeler

The CityGRID® Software-Suite was developed by UVM System for the acquisition and management of 3D city models. It consists of several modules for data collection, modeling, managing, visualizing and administration (Figure 3.7).

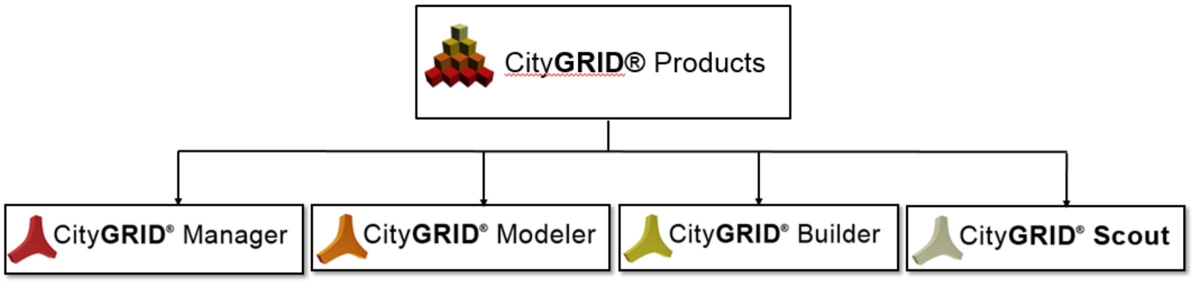


Figure 3.7: CityGRID® Products

CityGRID® Manager is a management system that is responsible for maintaining the model of the building on a database. The Manager stays in the background and provides functionalities for all CityGRID® modules. All modules of CityGRID use a common relational database (Oracle or SQL Server), where the models are stored as wire-frames (e.g. ridge lines, eave lines). The boundary representation can be automatically created from wire-frame model (Figure 3.8). The core of CityGRID is the triangulation algorithm.

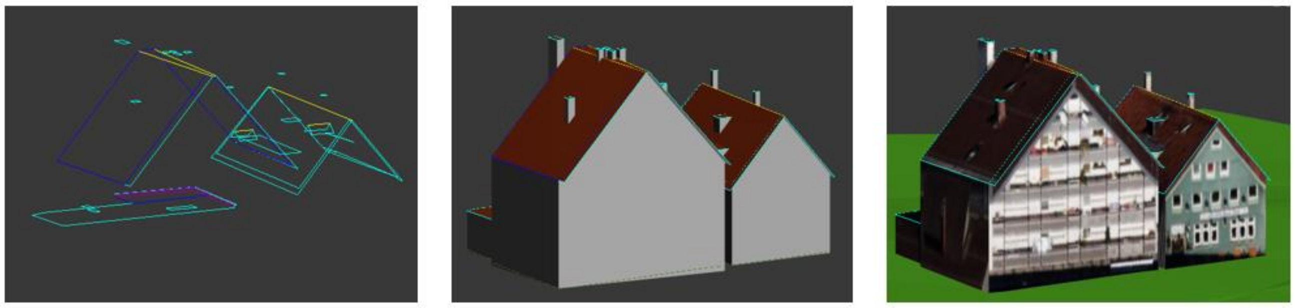


Figure 3.8: Wire-Frame model (left), the triangulated surface model (middle), the textured model (right)

CityGRID® Builder enables planers to integrate their drafts into the existing city model to generate visualizations of different planning variants and to create scenes optimized for CityGRID® Scout which provides a real-time visualization and simulation with the highly realistic appearance of urban models (Manual CityGRID Basics, 2015, S. 15).

CityGRID® Modeler is the second tool, which has been assessed in this master thesis. The Modeler a construction tool for creating,editing and texturing building model. It uses a line-oriented approach that generates 3D surface based on roof lines. Each roof or facade surface are derived by specially developed the triangulation algorithms. The prerequisite for the creation of such a model is knowledge of the lines (roof lines, building edges, floor plan) of a building (Kerschner, 2004)

The Modeler consists of available in three different modules, namely Inspector, Editor, and Texture. Inspector provides base functionality such as visualization and exporting of the model, whereas Editor includes editing tools and face generation algorithms. The module Texture is dedicated to texturing of the model.

Photogrammetric restitution

The initial tasks of the Modeler are editing of roof lines and modeling the roof surface from lines as well as adding texture to the facades and roofs. For this purposes, the Modeler uses specially developed algorithms that create 3D surfaces in boundary representation[5] based on the roof lines given in wire frame representation[6] that is extracted from the photogrammetric restitution method.

The algorithm requires well-defined roof structures in order to create 3D surfaces correctly. The most important roof structure, that face generation algorithm recognizes, is eave line (outer boundary of the roof). The eave line is demonstrated by blue lines in figure 3.9. The eave line often breaks in the vertical direction, so the special lines like upper and lower break edges have to be introduced into the mapping model (Franic et al., 2009). Violet lines in figure 3.9 show the upper and lower break edges.

[5] Boundary representation is method for representing solid objects based on vertices, edges and faces. (See https://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/model/b-rep.html, last access November 2016)
[6] Wire frame is a method that represents a solid object with lines and point (See https://www.cs.mtu.edu/~shene/COURSES/cs3621/NOTES/model/wireframe.html, last access November 2016)

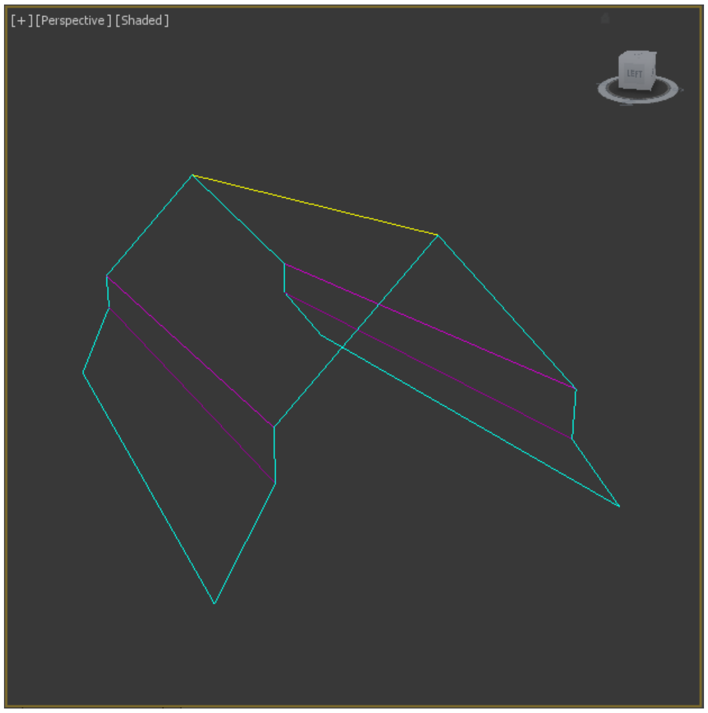


Figure 3.9: Mapping of roof structures.

General roof line classes, which are recognized by the face generation algorithm, are eave lines, ridge lines, upper break edge, lower break edge, dormer windows, facade lines, chimneys, and terrace.

Data structure

In CityGRID® buildings are organized in a multi-level hierarchy (Figure 3.10). "Model" appears at the top of this data structure hierarchy, which contains from any number of "Units". A group of buildings ("Objects") that belong together from an administrative point of view (e.g. the main building together with its garage) are represented as one "Unit" which is saved with a unique ID (Kerschner, 2004) (Figure 3.10).

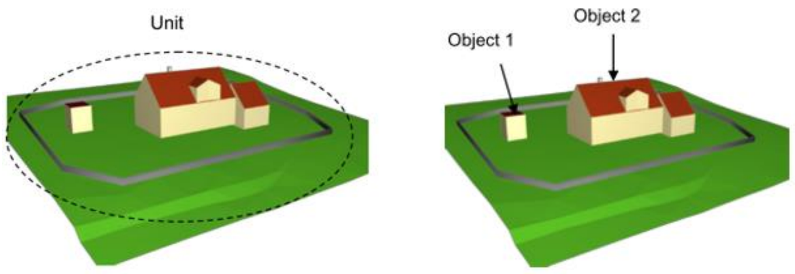


Figure 3.10: Data structure hierarchy - Unit and Objects (Manual CityGRID Basics, 2015, S. 10)

Objects symbolize a single building which consists of "Element Complex" and "Elements". An Element Complex is a group of Elements (i.e. roof, facade, protrusion, or floor)) and each Element Complex is a part of an Object (Figure 3.11). There are two types of Element Complex such as Main Element Complex (main solid of an Object) and Detail Element Complex (e.g. dormer, chimney) (Figure 3.11). Detail Element Complex can be considered as a child of its corresponding Main Element Complex.

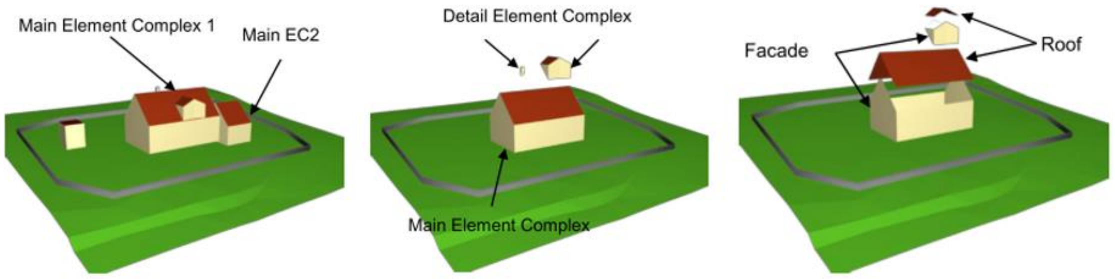


Figure 3.11: Data structure hierarchy – Element and Element Complex (Manual CityGRID Basics, 2015, S. 10)

User interface

The Modeler is a licensed plug-in Autodesk 3ds Max. Therefore Autodesk 3ds Max needs to be installed to work with CityGRID® Modeler.

Once the Modeler Plugin-in is started, a number of additional buttons and windows are appeared (Figure 3.12). The viewer contains three windows, which are 3D View, Top View, and Texture Dialog. 3D View shows the activated elements and Top View allows the user to see the model in the vertical direction from the top. Texture Dialog shows both texture image and the projected geometry for the interactive texturizing.

Next to the viewer, Main Window is located. Main Window consists of Main Dialog, Hierarchy Dialog, Layer Dialog and Properties Dialog. Main Dialog enables loading single units and multiple units. Hierarchy Dialog displays loaded units in hierarchy tree form. Layer Dialog shows the layers of the element complex which is selected in the Hierarchy Dialog. Finally, Properties Dialog shows information about selected element properties such as face generation type, default color.

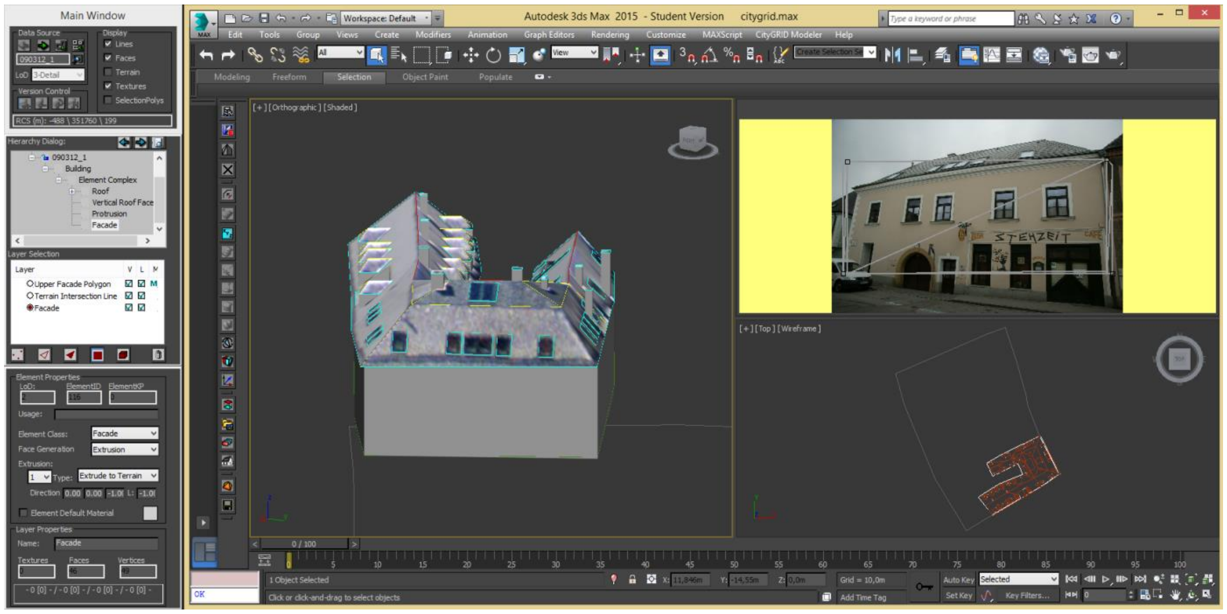

Main Window
Data Source: 090312_1, LoD: 3-Detail
Display: Lines [✓], Faces [✓], Terrain, Textures [✓], SelectionPolys
RCS (m): -488 \ 351760 \ 199
Hierarchy Dialog: 090312_1 – Building – Element Complex – Roof, Vertical Roof Face, Protrusion, Facade
Layer Selection: Upper Facade Polygon, Terrain Intersection Line, Facade
Element Properties: LoD: 2, ElementID: 116, ElementKP: 0
Element Class: Facade
Face Generation: Extrusion
Extrusion: 1, Type: Extrude to Terrain
Direction: 0.00 0.00 -1.0 L: -1.0
Layer Properties: Name: Facade
Textures: 0, Faces: 46, Vertices: 49
Autodesk 3ds Max 2015 - Student Version citygrid.max

Figure 3.12: the User interface of CityGRID® Modeler.

Modeling a single building

As it has been mentioned before, modeling buildings with the Modeler are based on face generation algorithm. The face generation algorithm generates the roof by triangulation inside of its eave lines where ridge or other roof lines are used as constraint edges (Figure 3.13).

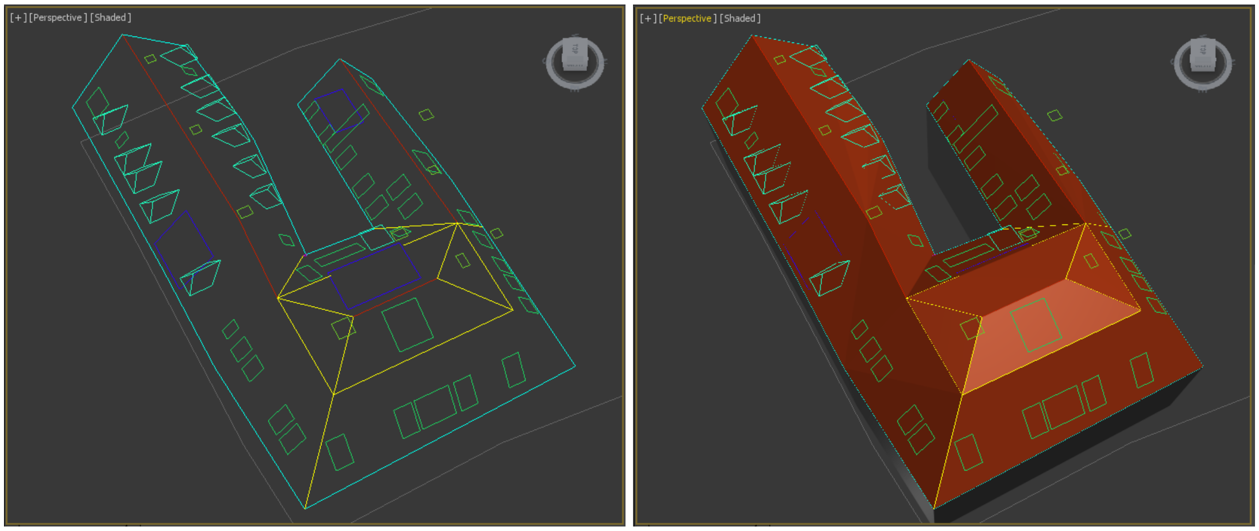


Figure 3.13: Roofline structure (e.g. Blue lines are eave lines, red lines are ridge lines, and yellow lines are general roof line)

In some cases, the Modeler can be used to correct the dataset that derived from photogrammetric restitution. Typical editing actions include moving lines into the proper layers, close topological connections, insert further roof edges and so on (Figure 3.14). The aim is to edit lines until the derived roof faces are correct (Kerschner, 2004).

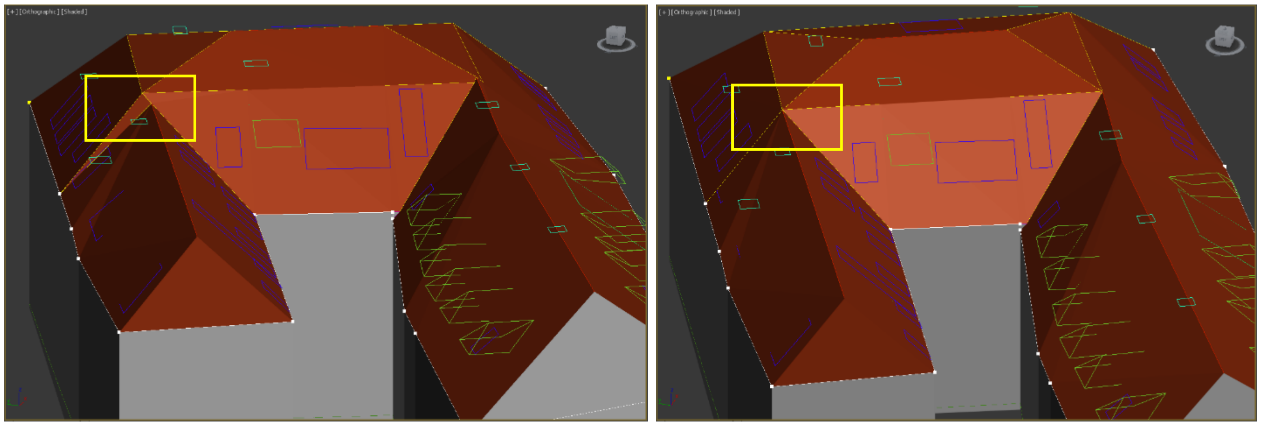


Figure 3.14: The correction of the error on the ridge line

Every time the geometry of an object is edited and triangulated, which is necessary to perform the changes on the faces, the system creates a new version of the model and stores it in the database providing the editing history (Figure 3.15).

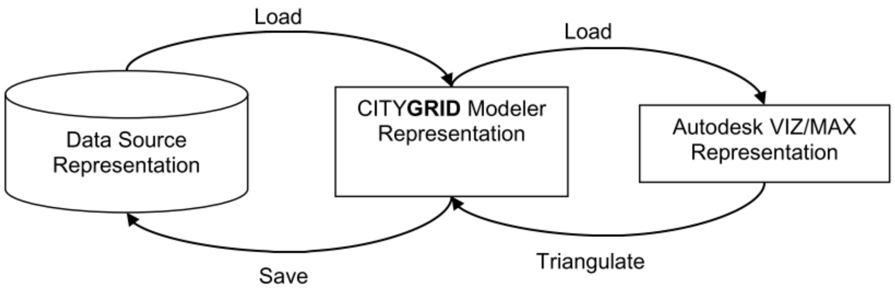

Load
Load
Data Source Representation
CITYGRID Modeler Representation
Autodesk VIZ/MAX Representation
Save
Triangulate

Figure 3.15: Basic working principle of CityGRID Modeler

The faces of the facade are generated by extruding the roof outer eave lines to the ground till it intersects with the lowest point of the terrain model Figure 3.16. In order to bring the model to a higher level of detail, the roof detailed such as dormer windows, terrace and chimneys can also model by face generation algorithm.

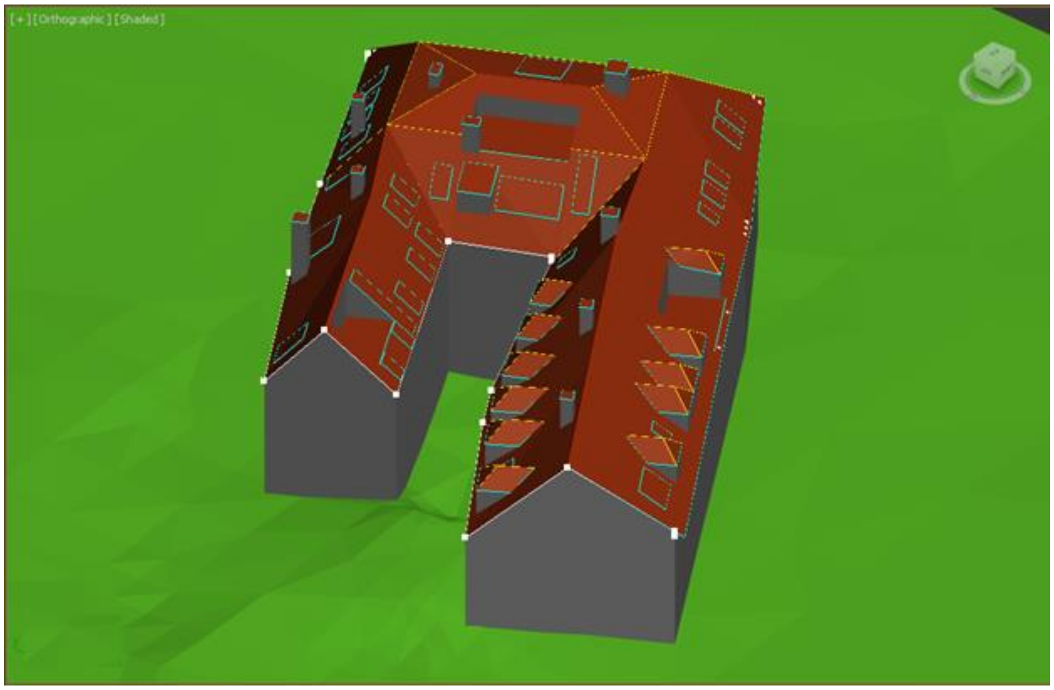


Figure 3.16: Building model with dormer windows, terrace, and chimneys

3.5 SketchUp by Trimble

In 2000, SketchUp was introduced by Last Software as a 3D modeling software which is widely used in various fields of architecting, modeling and designing like medical simulation, robotics, civil, mechanical, film, video, and game designing. Google bought the company in 2006 and that brought a new dimension to the software, which enables placement of the models into Google Earth. In 2012, SketchUp has been acquired by Trimble Inc., a provider of location-based solutions.

SketchUp features a sketch-based modeling approach which means the user can quickly draw a 2D shape and convert it to 3D model. This approach makes use of SketchUp easy for the users with no experienced in 3D modeling software.

In SketchUp, edges and faces are fundamental geometry type for modeling. Each object in a model is constructed by edges and faces. For instance, the basic cube on the left top in figure 3.17, which is modeled in SketchUp, consists of 12 edges and 6 faces. Edges are always straight lines in SketchUp. Even arcs and circles are composed of small straight lines. Faces are the surface of the model. A face in SketchUp is constructed by at least three coplanar edges that surround it. Faces in SketchUp are always flat. Curved or no coplanar faces are created by multiple faces.

SketchUp has an inference engine, which streamlines generating accurately and quickly models. The inference engine basically finds out the consequential relationship between points, line, edges, and surfaces. As its name suggests, it infers and gives cues and hints to the user by tooltips, when it finds a consequential relationship in the surrounding of the cursor. Figure 3.17 shows examples of tooltips.

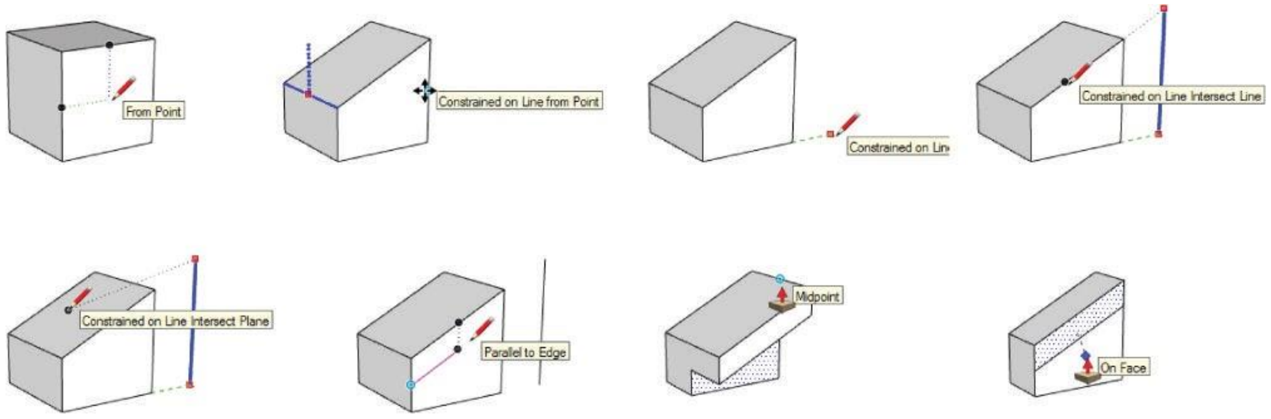

From Point
Constrained on Line from Point
Constrained on Lin
Constrained on Line Intersect Line
Constrained on Line Intersect Plane
Parallel to Edge
Midpoint
On Face

Figure 3.17: Some of the inference engine of SketchUp[7]

SketchUp is very versatile in modeling. Both, modelling the outside of buildings and their interior is possible. All possible objects such as vehicles, animals, furniture and lot of other can be modeled by mean of SketchUp (Figure 3.18).

[7] Source of the figure: http://www.sketchup-ur-space.com/2013/april/SketchUp-interference.html; last access November 2016.

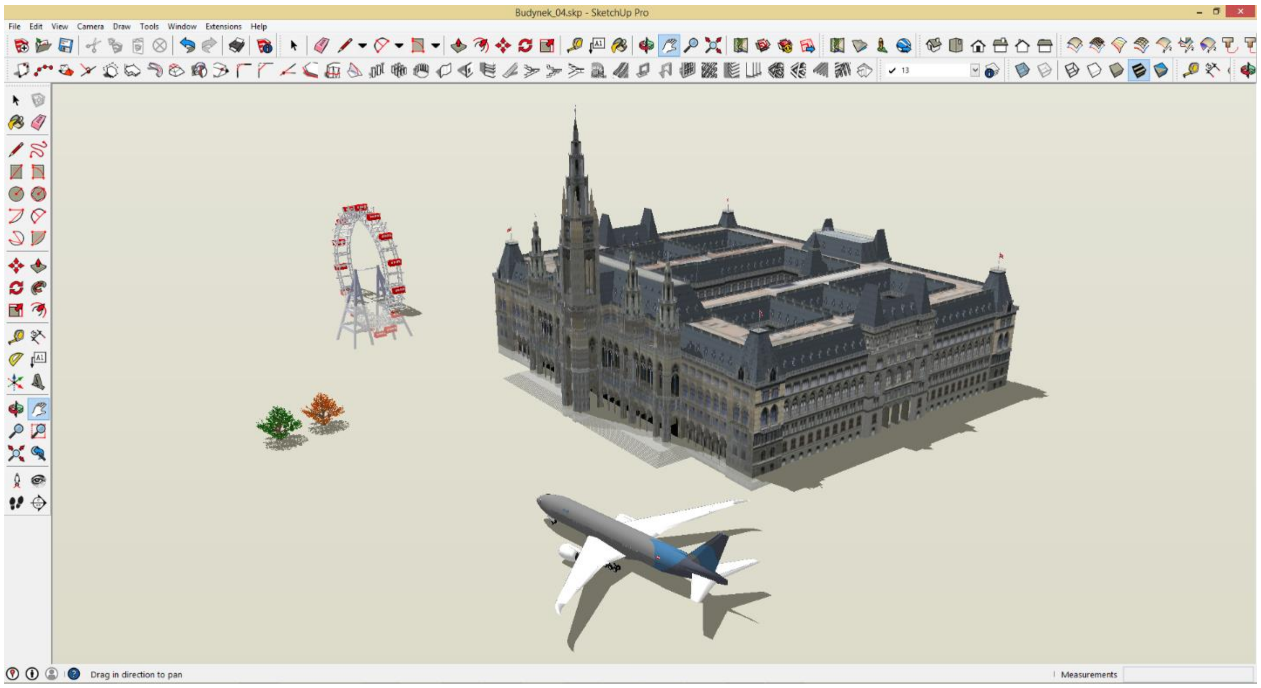

Budynek_04.skp - SketchUp Pro
File Edit View Camera Draw Tools Window Extensions Help
Drag in direction to pan
Measurements

Figure 3.18: Different modeled objects by SketchUp[8]

There are two versions of SketchUp, “Make” and “pro” versions. SketchUp Make is a free downloadable version for non-commercial purposes. SketchUp pro is a professional/commercial version. Therefore, it has advanced features. The significant difference between two versions is import and export options.SketchUp Make version can export the models in *kmz* and *collada* while SketchUp Pro version can export the models in *3ds*, *wrl*, *xsi*, *dwg*, *dwf*, *fbx*, and *obj* file formats.Add-on programs can be created by using Ruby programming language, or various types of the plugin can be freely downloaded via 3D Warehouse of SketchUp. In the thesis, SketchUp Pro 2015 version has been used as well as the different plugins which will be explained in Chapter 5.

User interface

Most functions and toolbars can be found via the menu bar (Figure 3.19). Two main toolbars are Getting Start Toolbar and Large Tool Set that consist of the following functions: drawing, editing, moving, rotating, scaling and other construction tools as well as camera tools.

[8] The objects in the figure are freely downloadable via https://3dwarehouse.sketchup.com; last access November 2016.

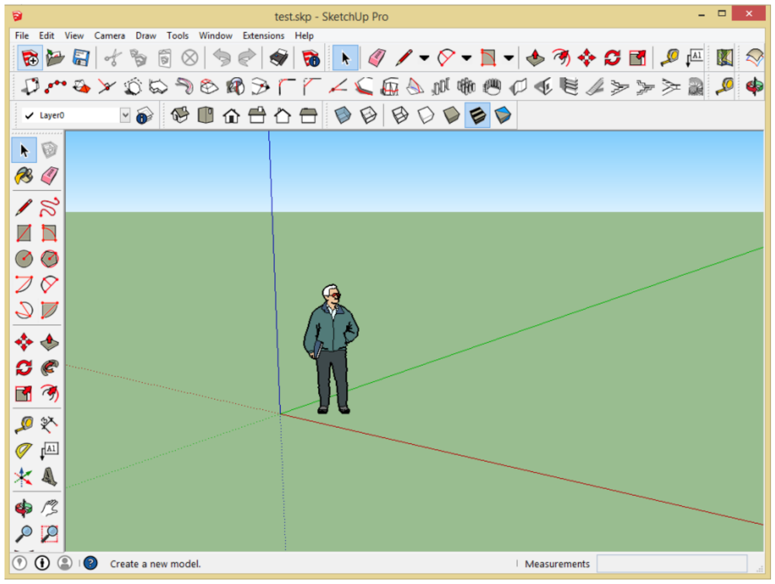


Figure 3.19: User interface SketchUp Pro 2015

The other practical tools are view and style. The view enables the change the prefixed camera position quickly as iso, top, front, right, back and left. While modeling, the mesh structure can be very important, and the style tool allows to change it easily.

The modeling window consists of a 3D space environment, in which the objects are created or edited. The 3D coordinate system is identified by three axes which have different colors (Figure 3.20). The ground plane is defined by the red and green axes and the intersection of three lines define the origin.

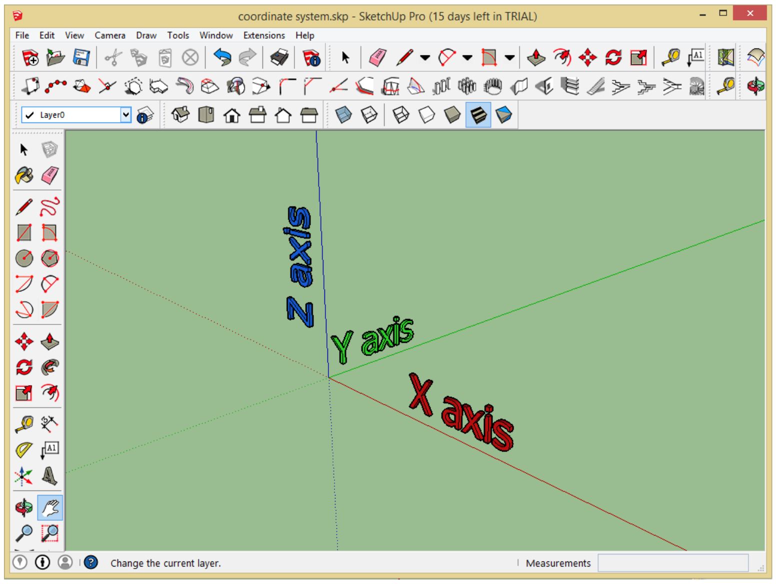
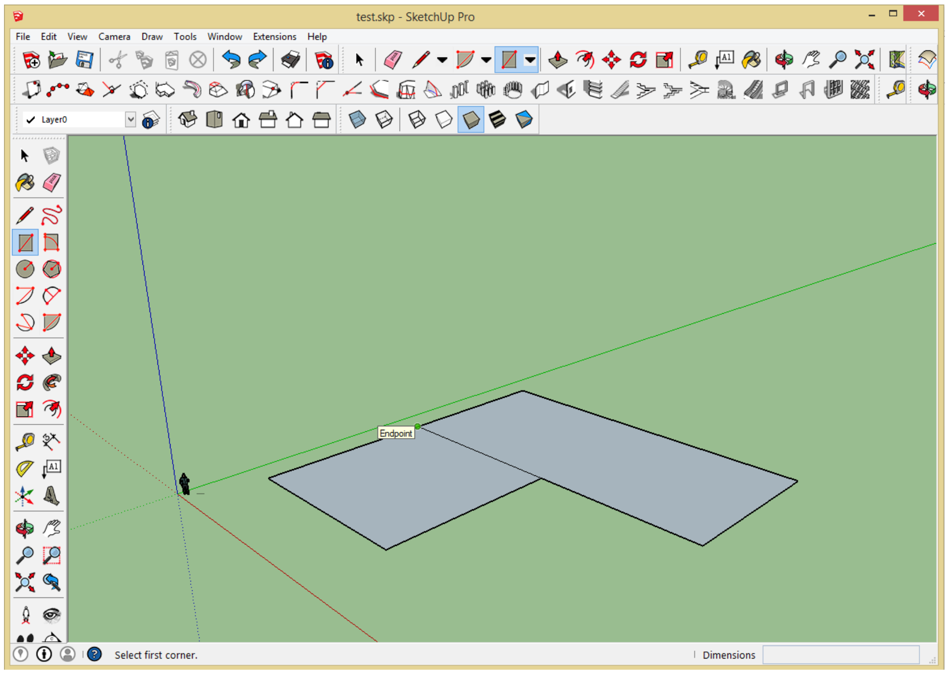

coordinate system.skp - SketchUp Pro (15 days left in TRIAL)
File Edit View Camera Draw Tools Window Extensions Help
Layer0
Z axis
Y axis
X axis
Change the current layer.
Measurements

Figure 3.20: SketchUp X-Y-Z coordinate system

Modeling a single building

In this chapter, the modeling of a building with basic tools of SkecthUp is demonstrated. The modeling of the building is started by drawing two rectangular faces (2D shape) by using

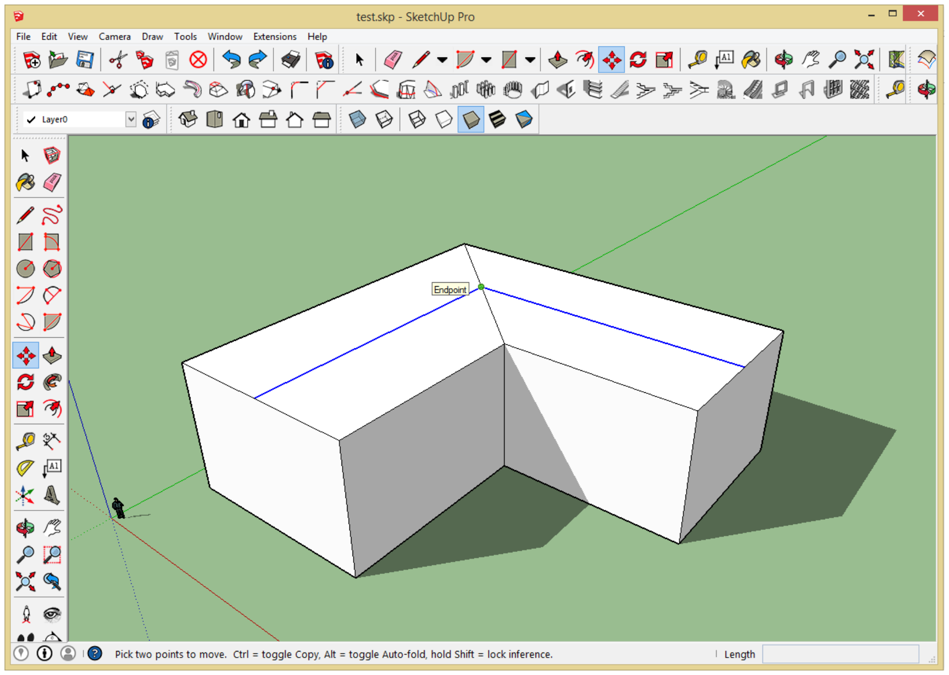
Drawing tools (Figure 3.21). The intersected line can be deleted if one wants to obtain only one faces.



Select first corner.

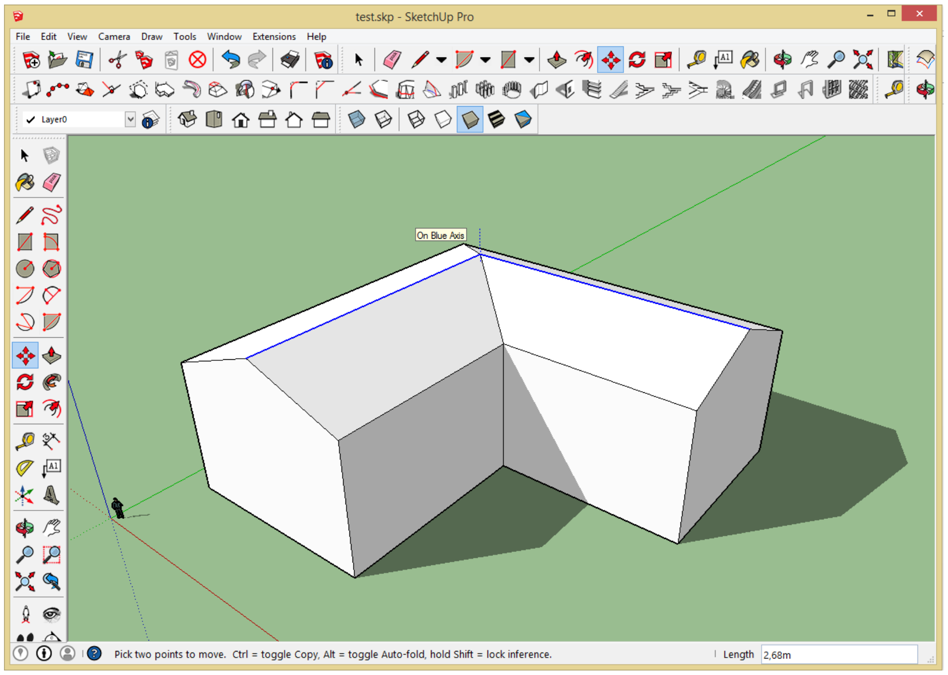
Figure 3.21: Sketching 2D Objects

By means of the Push/Pull tool, the L-Shaped faces are converted to building block (from 2D to 3D) (Figure 3.22). On the top of the block, the ridge lines and valley are drawn in order to form the roof. The ridge lines are selected and moved up with the Move tool and hereby the gable roof is generated (Figure 3.23).



Pick two points to move. Ctrl = toggle Copy, Alt = toggle Auto-fold, hold Shift = lock inference.

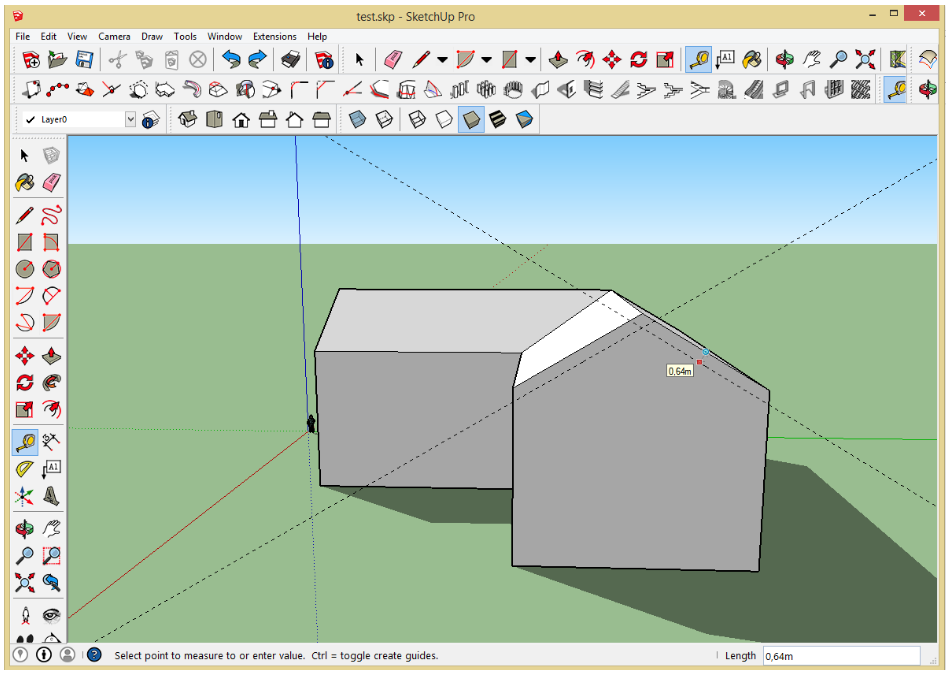
Figure 3.22: Converting from 2D Object to 3D



Pick two points to move. Ctrl = toggle Copy, Alt = toggle Auto-fold, hold Shift = lock inference.

Figure 3.23: Modeling the roof form

Roof details can be modeled with a combination of the tool sets – Offset and Push/Pull. Tape measure tool can be used to measure the thickness of the roof (Figure 3.24) and Offset and Push/Pull enables modeling the roof overhang (Figure 3.25).



Select point to measure to or enter value. Ctrl = toggle create guides.

Figure 3.24: Measuring the roof thickness

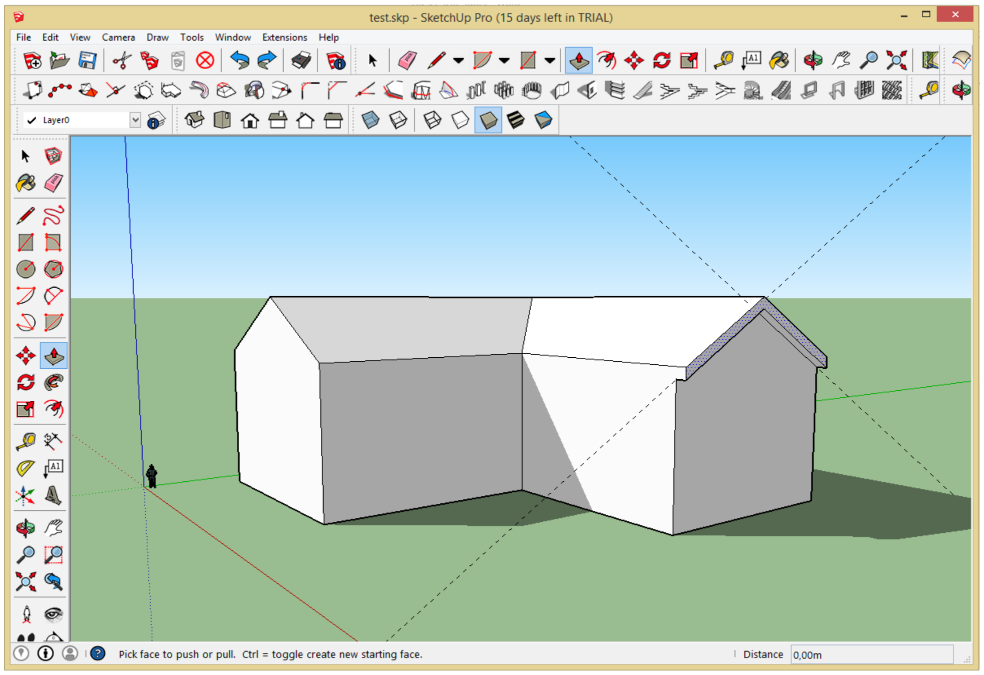

test.skp - SketchUp Pro (15 days left in TRIAL)
File Edit View Camera Draw Tools Window Extensions Help
Layer0
Pick face to push or pull. Ctrl = toggle create new starting face.
Distance 0,00m

Figure 3.25: Modeling the roof overhang

The door, windows, and chimney are created by drawing rectangle faces and using Push/Pull tools to convert rectangle faces to 3D block. The model can be further detailed with other basic tools of SketchUp as well as commercial /non-commercial plugins.

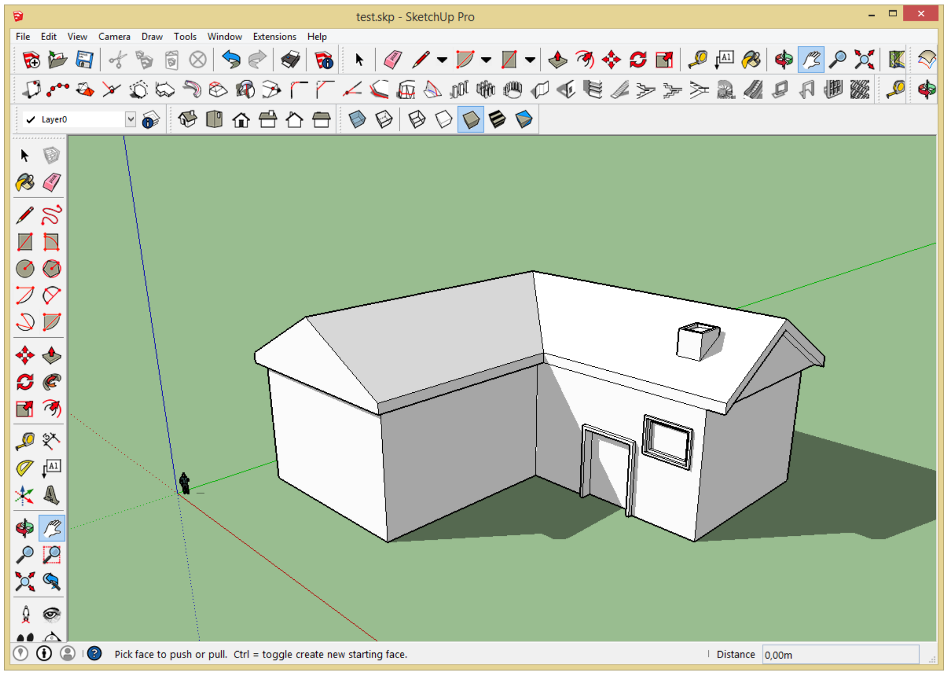

test.skp - SketchUp Pro
File Edit View Camera Draw Tools Window Extensions Help
Layer0
Pick face to push or pull. Ctrl = toggle create new starting face.
Distance 0,00m

Figure 3.26: Final 3D Model

The objects, which require being modeled multiple times (e.g. fences of the garden door), can be modeled as a component. A component can be copied, and the copies will be linked to the original component that is called instances. If the original component is changed, then all instances will be changed.

CHAPTER 4

4 STUDY AREA AND DATA DESCRIPTION

A case study in this thesis has been performed with the intention to assess the afore mentioned three different software packages for creating 3D city model. In this chapter, the datasets, which have been used by the software, and study area are presented.

4.3 Study Area

The geographic area used in this case study is Marktplatz, Reutlingen (Figure 4.1), which is a city in Baden-Württemberg, Germany and lies in the Southwest corner of Germany. The city is circa 87.06 km² and has circa 12,500 inhabitants (2014).

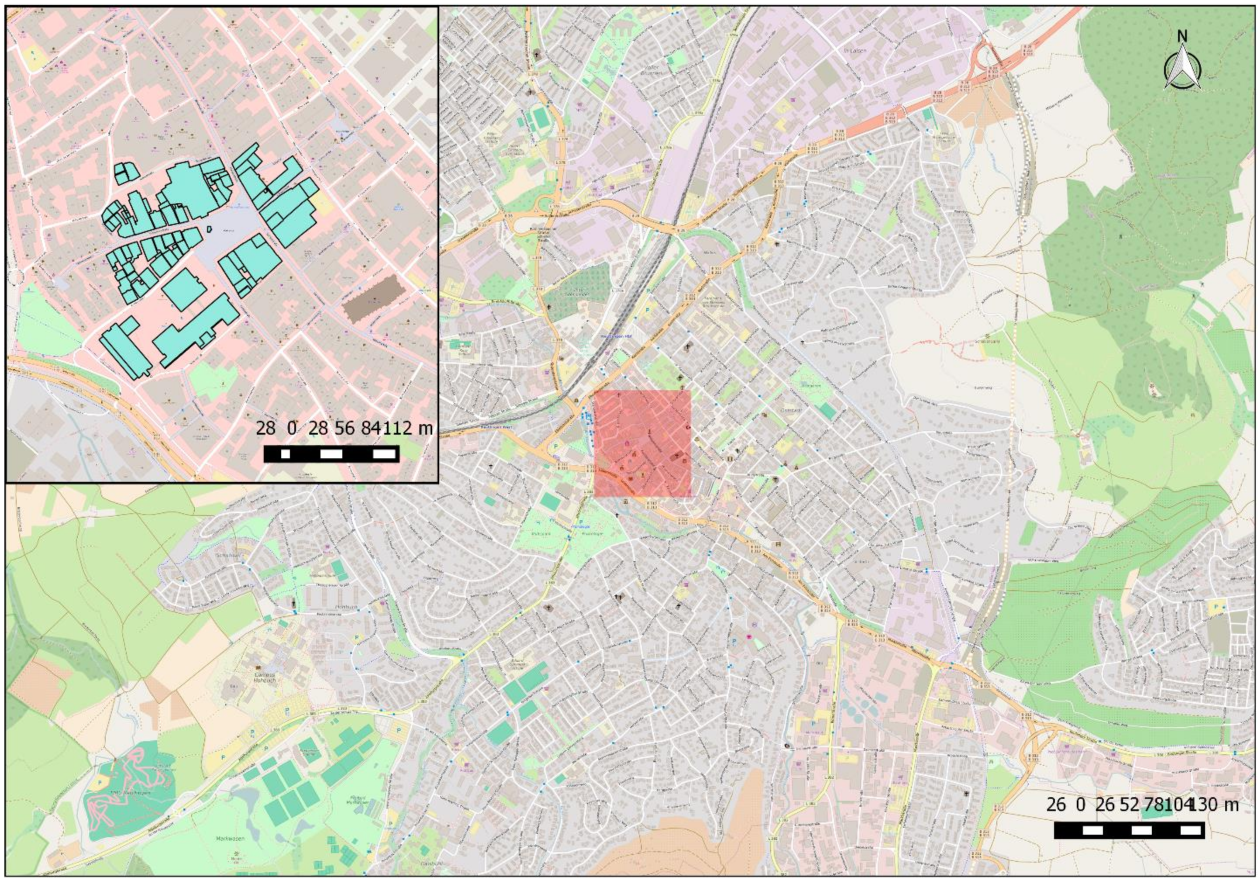


Figure 4.1: Case Study Area – Marktplatz, Reutlingen

4.4 Data

4.4.1 Aerial Imagery

In the case study, 17 nadir aerial images have been used. The images were captured by UltraCam Eagle camera from Vexcel Imaging GmbH (located in Graz, Austria) on 5 August 2013. It delivers large frame format (20010 pixels across-track and 13080 pixels long track) panchromatic images and simultaneously recorded four channels (red, green, blue and NIR) at a frame format of 4360 pixels across-track and 6670 pixels long track. For both the panchromatic and the color images the physical pixel size is 5.2 μm by 5.2 μm. The radiometric resolution of the sensor in each channel is >12-bit, the analog-to-digital conversion works at 14-bit and the digital format of the entire processing makes use of the 16-bit data format.

Each image overlaps the next image within a flight strip by 60% (forward overlap or longitudinal overlap) and adjacent strips overlap by 60% (lateral overlap) (Figure 4.2).The 60/60 overlapping imagery enables to get a view from four different perspectives. This is essential for texturing purposes, as every facade is visible at least in one image.

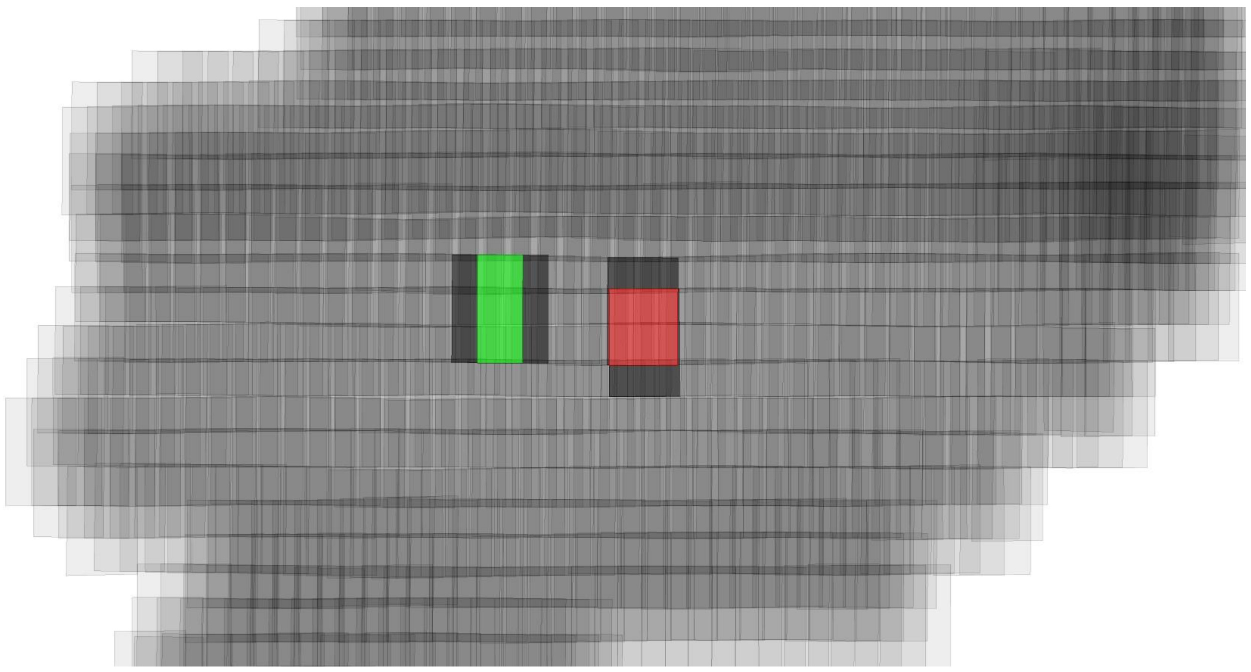
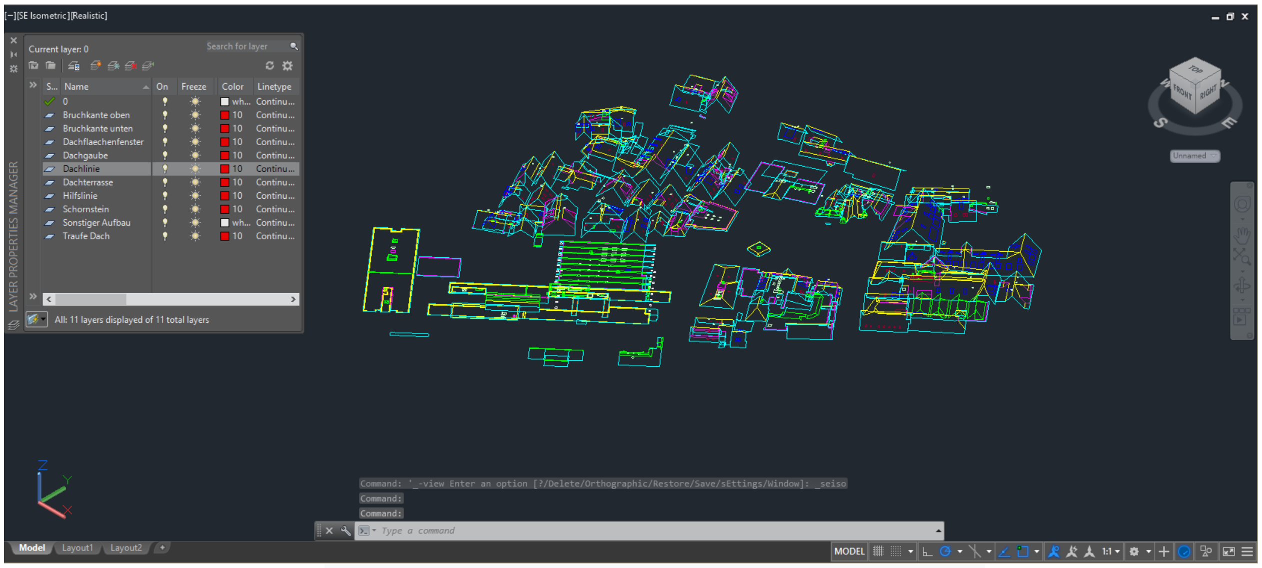


Figure 4.2: Block of aerial photographs of the study area. Green color indicates longitudinal overlap and red indicates lateral overlap

4.4.2 Roof lines

The roof lines (ridge, eave lines, break lines or other roof lines) have been determined through photogrammetric restitution of aerial images by GEOGIS Trade Inc. (located Ankara, Turkey) and corrected by UVM Systems GmbH (located in Klosterneuburg, Austria) in spring 2015. (Figure 4.3). The dataset has not only the geometry of the roof structure but also the semantics of roof structure. The layer window in Figure 4.3 shows all roof lines types of the dataset. The dataset is obtained in *dwg* format.



S...	Name	On	Freeze	Color	Linetype
✓	0	●	☀	⬜ wh...	Continu...
	Bruchkante oben	●	☀	🟥 10	Continu...
	Bruchkante unten	●	☀	🟥 10	Continu...
	Dachflaechenfenster	●	☀	🟥 10	Continu...
	Dachgaube	●	☀	🟥 10	Continu...
	Dachlinie	●	☀	🟥 10	Continu...
	Dachterrasse	●	☀	🟥 10	Continu...
	Hilfslinie	●	☀	🟥 10	Continu...
	Schornstein	●	☀	🟥 10	Continu...
	Sonstiger Aufbau	●	☀	⬜ wh...	Continu...
	Traufe Dach	●	☀	🟥 10	Continu...

All: 11 layers displayed of 11 total layers

```
Command: '_-view Enter an option [?/Delete/Orthographic/Restore/Save/sEttings/Window]: _seiso
Command:
Command:
Type a command
```

Figure 4.3: Roof lines dataset visualization in AutoCAD

4.4.3 Footprint

Quite often the building footprint (building ground plan) outlines the building structure and is identical to the precise location of the exterior walls. In the case of a simple roof shape (like a flat roof), the building footprint can also be an indicator for the boundary of the roof face. Hence, building footprints dataset are combined with other data source to improve the reconstruction of the 3D building model. The corresponding footprints are provided in *shp* format by Amtliche Liegenschaftskatasterinformationssystem (ALKIS®), year N/A (Figure 4.4).

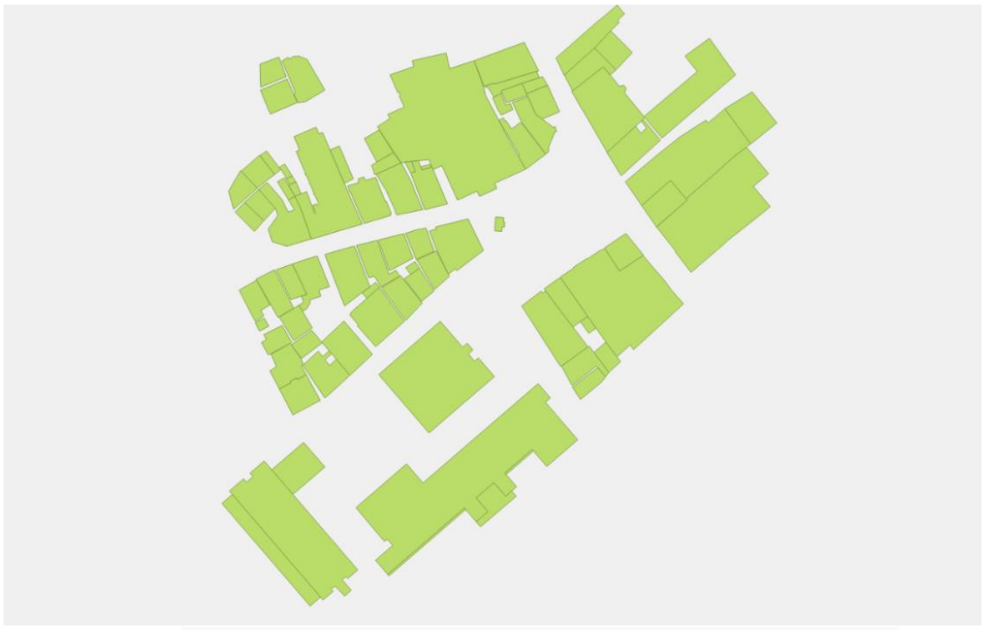


Figure 4.4: Footprints of the buildings in the case study area

4.4.4 Digital Terrain Model

The digital terrain model of the case study area is obtained from City Reutlingen in both *TIN* and *XML* format (Figure 4.5).

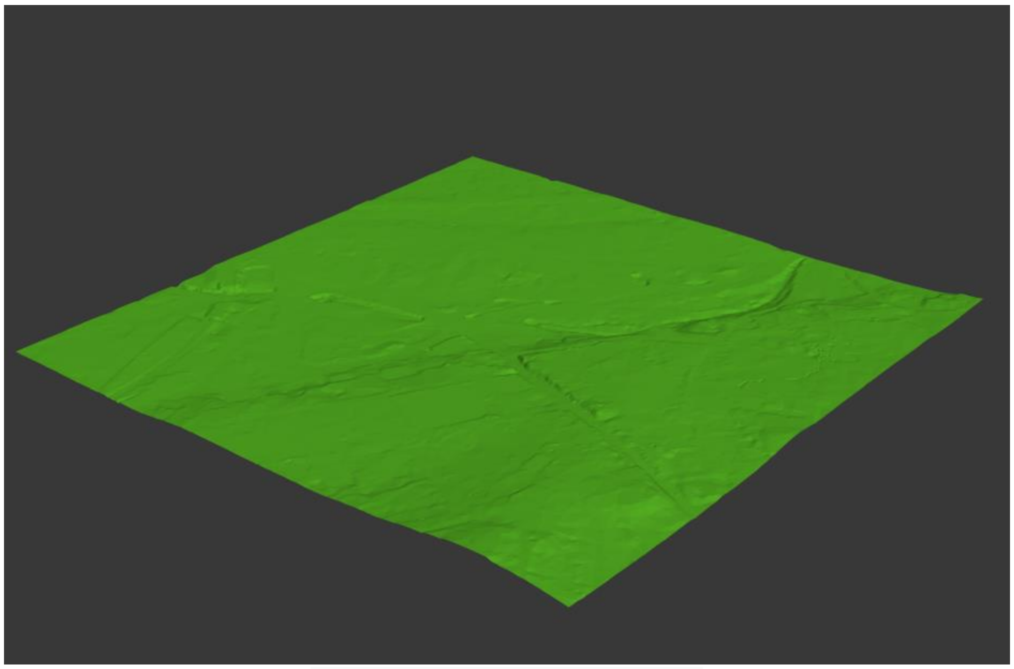


Figure 4.5: Digital terrain model

CHAPTER 5

5 EXEMPLARY 3D CITY MODELING

In the following the focus will be laid on 3D modelling in practice. With the help of each modelling tool, a 3D city model has been created as detailed as (Figure 5.1) the tool allows and the model has been texturized with aerial images. Each tool requires different processes and utilizes a different combination of data. In this chapter, those different processes will be presented.

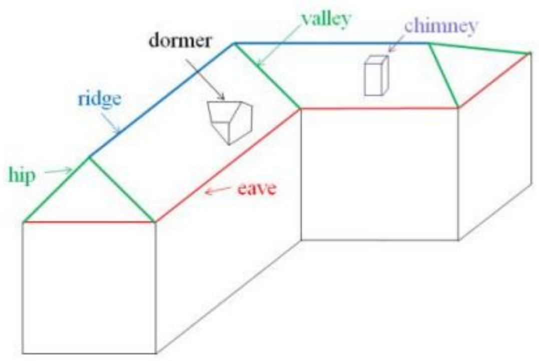


Figure 5.1 Roof components which should be reconstructed for the highest roof LODs (Nex and Remondino, 2012)

5.3 3D City Modeling with 3D Editor

The primary aim of the case study with 3D Editor was utilizing the combination of roof lines, footprint, and DTM datasets in order to create a 3D City model. However, the processing of roof lines in *dxf* format by means of 3D Editor was not possible. Tridicon data format (tdc/op3d) contains not only geometry but also information regarding roof types such as roof surface, wall surface or gable roof surface. Since such information was not stored in the roof line dataset as 3D Editor requires, another process follows in order to create 3D city model (Figure 5.2).

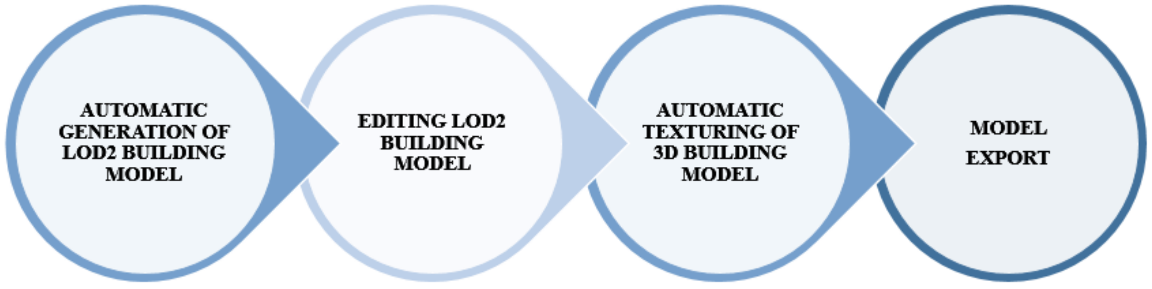

AUTOMATIC GENERATION OF LOD2 BUILDING MODEL
EDITING LOD2 BUILDING MODEL
AUTOMATIC TEXTURING OF 3D BUILDING MODEL
MODEL EXPORT

Figure 5.2: Implementation process

5.3.1 Generation LOD2 by mean of City Modeller

The CityModeller is developed by 3DCon GmbH as a solution for the automatic derivation of 3D city model in Level of Detail 2. The software uses aerial images as well as the building footprints to generate LOD2 building. Furthermore, digital terrain or point clouds may be added (Figure 5.3).

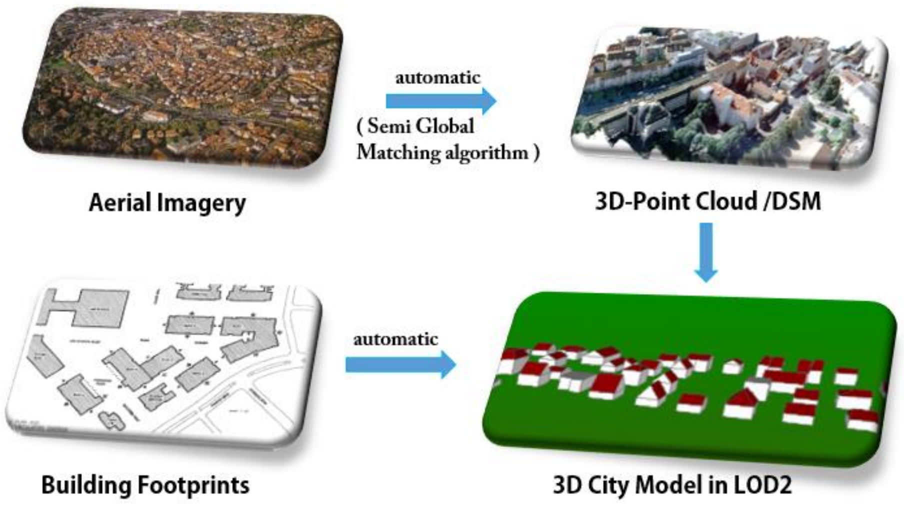


Figure 5.3: Working principle of CityModeller

The software generates 3D point clouds from aerial images and uses a detailed roof library (pattern recognition) in order to automatically extract LOD2 models (Figure 5.4 & 5.5).

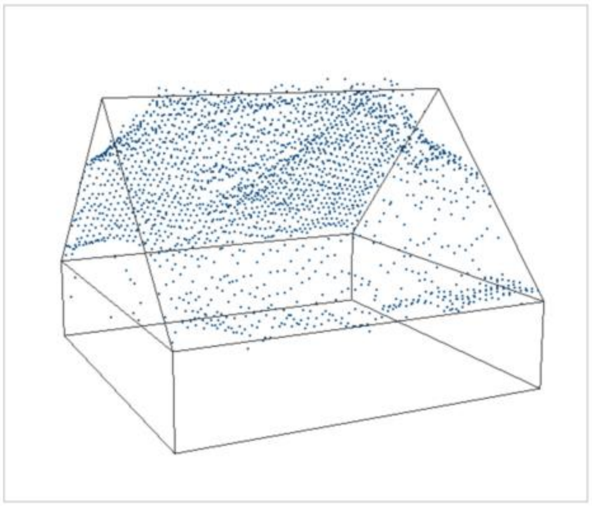


Figure 5.4: Gable roof

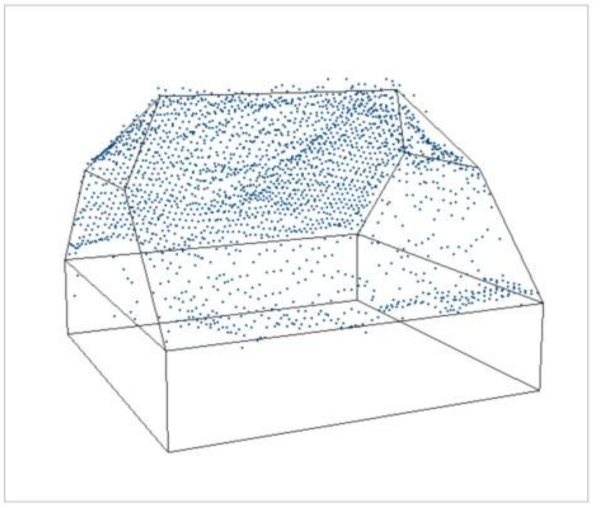
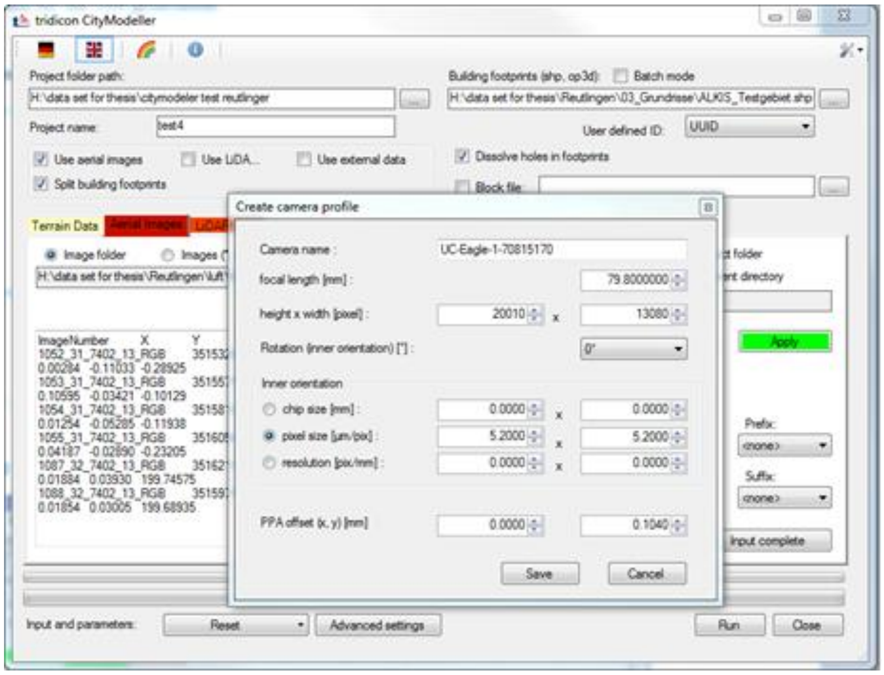


Figure 5.5: Hipped-gable roof

To accomplish the aim of the case study, the process has started by creating LOD2 by mean of CityModeller. The process starts with data preparation and import. Aerial images, building footprint, and DTM have been imported respectively. For the automatic derivation of 3D buildings, the program needs image pyramids and per image an orientation file in indigenous *gori* format (CityModeler Manual, 2009 -2015, S. 20). As there was no orientation file in *gori* format available, they have been created by GoriCreator, which CityModeller provides for this purpose (Figure 5.6). Required information for the orientation was provided by the camera calibration report (by Vexcel Imaging GmbH, Feb 2013).The program creates the orientation files for the standard rotation angles and for the horizontal and vertical arrangement of lines and sample. Afterward, the orientation preview window opens in which the orientation model can be selected that matches the input data (CityModeler Manual, 2009 -2015, S. 23).



Project folder path: H:\data set for thesis\citymodeler test reutlinger
Project name: test4
Building footprints (shp, op3d): H:\data set for thesis\Reutlingen\03_Grundrisse\ALKIS_Testgebiet.shp
Batch mode: ☐
User defined ID: UUID
☑ Use aerial images ☐ Use LiDA... ☐ Use external data
☑ Split building footprints
☑ Dissolve holes in footprints
☐ Block file:
Image folder: H:\data set for thesis\Reutlingen\luft

```
ImageNumber        X        Y
1052_31_7402_13_RGB        35153
0.00284  -0.11033  -0.28925
1053_31_7402_13_RGB        35155
0.10595  -0.03421  -0.10129
1054_31_7402_13_RGB        35158
0.01254  -0.05285  -0.11938
1055_31_7402_13_RGB        35160
0.04187  -0.02890  -0.23205
1087_32_7402_13_RGB        35162
0.01884  0.03930  199.74575
1088_32_7402_13_RGB        35159
0.01854  0.03005  199.68935
```

Create camera profile
Camera name: UC-Eagle-1-70815170
focal length [mm]: 79.8000000
height x width [pixel]: 20010 x 13080
Rotation (inner orientation) [°]: 0°
Inner orientation
chip size [mm]: 0.0000 x 0.0000
[✓ pixel size [µm/pix]: 5.2000 x 5.2000]
resolution [pix/mm]: 0.0000 x 0.0000
PPA offset (x, y) [mm]: 0.0000 0.1040
Prefix: <none>
Suffix: <none>
Input and parameters:

Figure 5.6: Creating orientation file in GORI format

The accepted image formats are *vit* and *tif*. The data import process was completed by adding DTM in *las* format. Before the project is executed, the further function can preferably be set in order to optimize the model. Those functions are RoofSplitter, Roof Options, and Buildings Adjustment.

In the case study, in order to get better roof detection, RoofSplitter has been activated, which allows the footprint split into individual parts, and the algorithm detects the roof type based on those smaller parts of the footprint. RoofSplitter let users choose between three different operation modes: automatically, interactively or pre-processing. The interactive and pre-processing operation modes allow users to check and adapt the footprint splitting whereas the automatic operation mode splits the building footprint automatically. In order to avoid any manual interaction, Roof Splitter operation mode has chosen automatically in the case study. After all, the program has been executed.

According to the log file, the automatic derivation has successfully completed in 53minutes and 4 seconds, and 149 roof forms were detected (Table 5.1). The footprint dataset has initially 81 objects, which are split during the roof detection that the eventual number of individual objects yields 149. The visualization of the result can be seen in Figure 5.7 and 5.8.

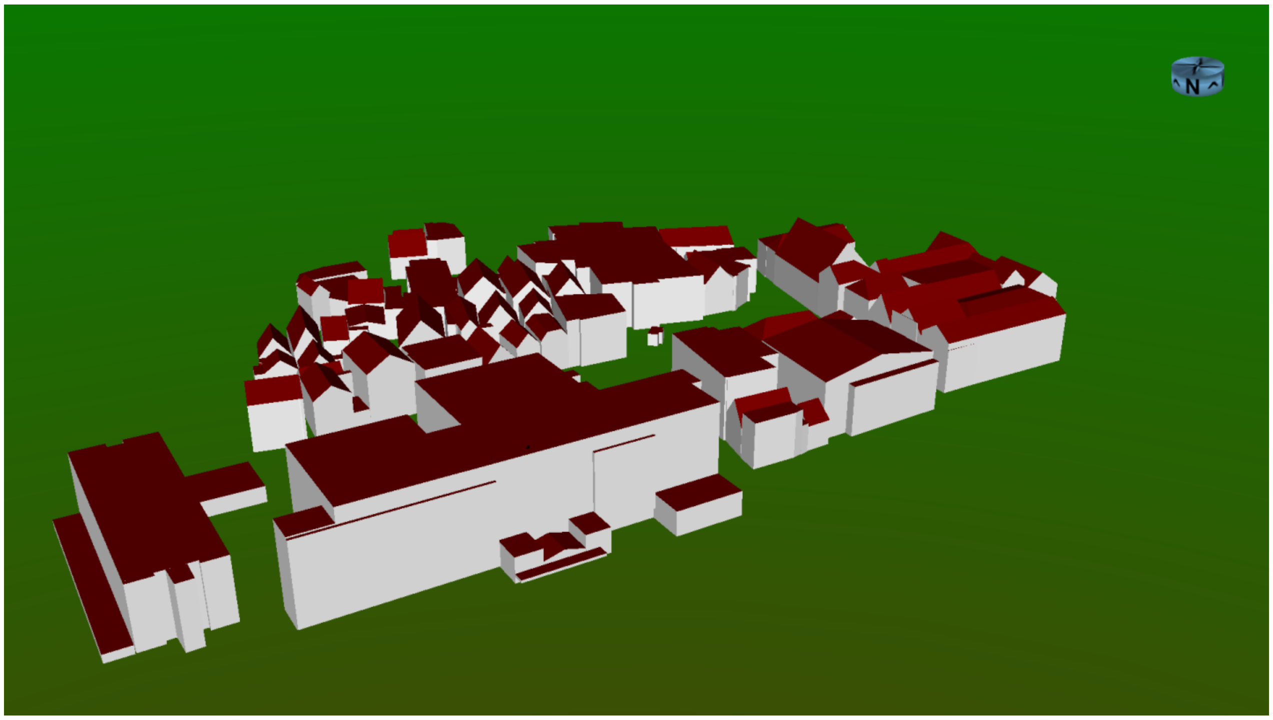


Figure 5.7: Visualization of LOD2 Model

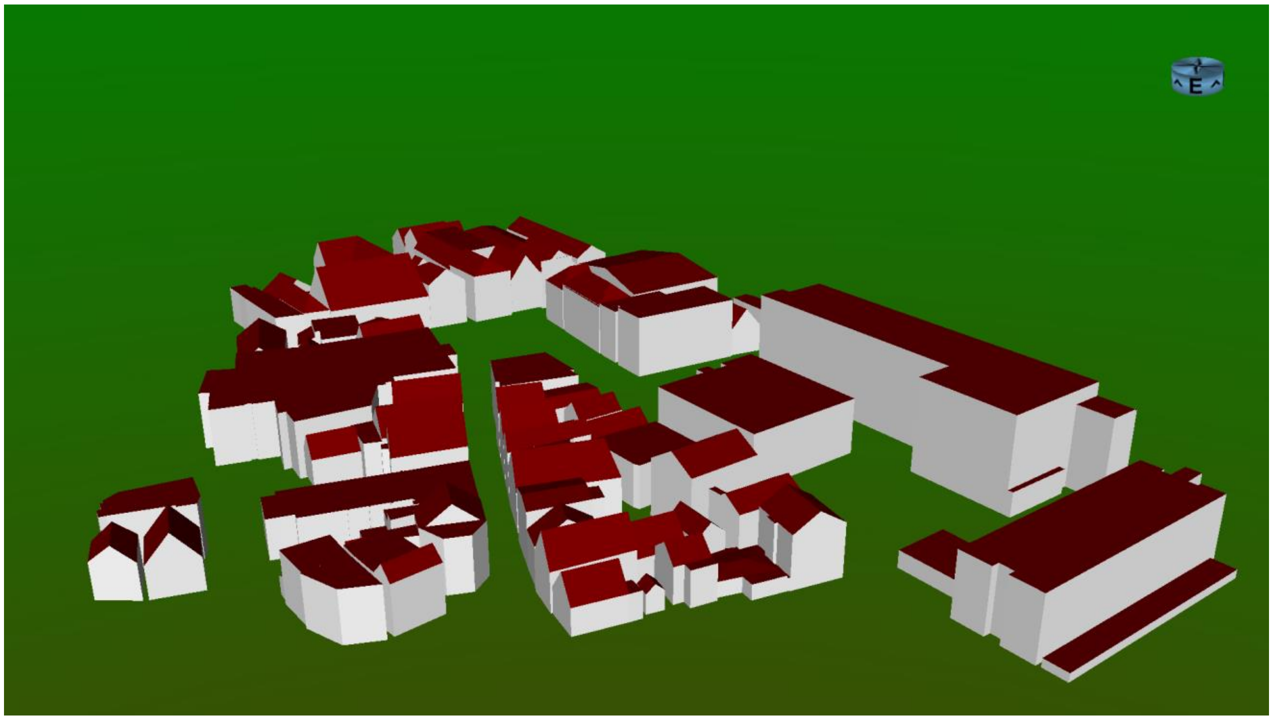


Figure 5.8: Visualization of LOD2 Model

Roof form	Number
Lean-to roof	4
Gable roof	41
Hip roof	3
Gable and hip roof	6
Cut gable roof	2
Flat roof	87
Flat roof (LOD1)	6
Not found	0
Sum	149

Table 5.1: Result of the automatic derivation of roof form

As it can be seen in figure 5.9, the result of the automatical detection is colored depending on how certainty the automatism has detected the correct roof. The quality value is measured in percent and describes the percentage of all points within a footprint which contribute to the detected roof shape. For instance, if 100 of the point cloud fall into a certain footprint and 70 of these describe a certain roof form, the quality value of this roof form within this footprint will be 70% (CityModeler Manual, 2009 -2015, S. 94).

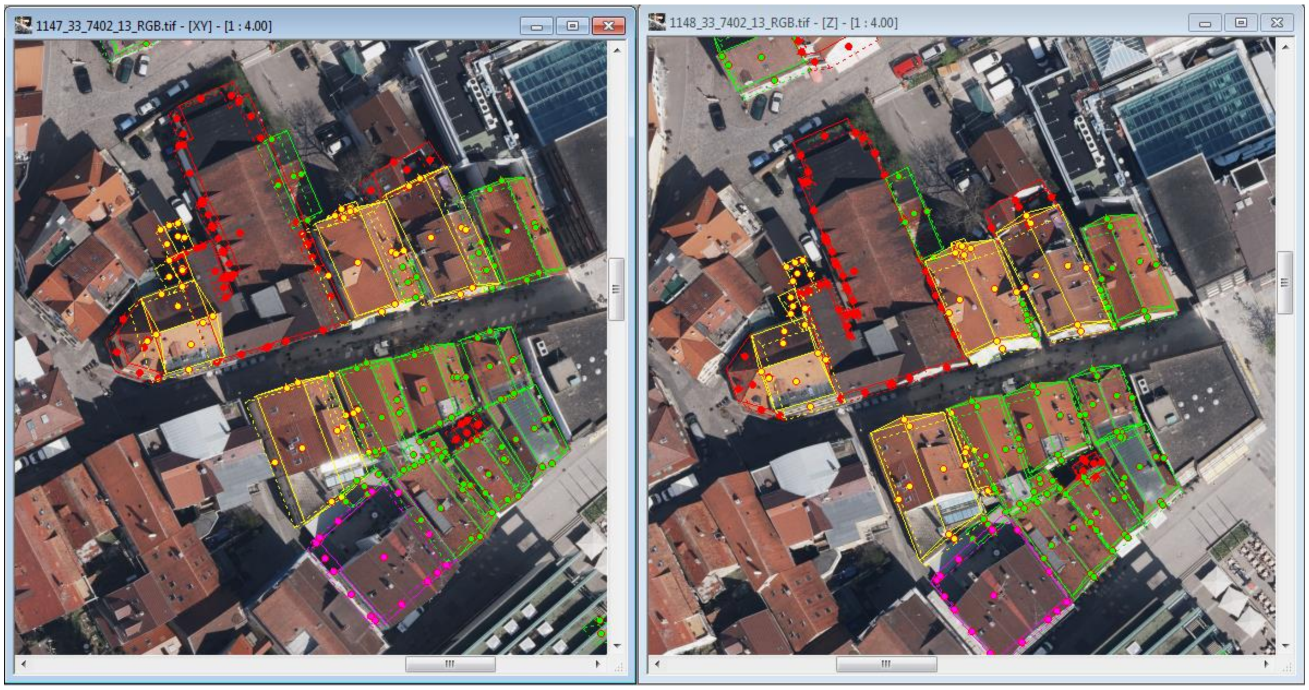
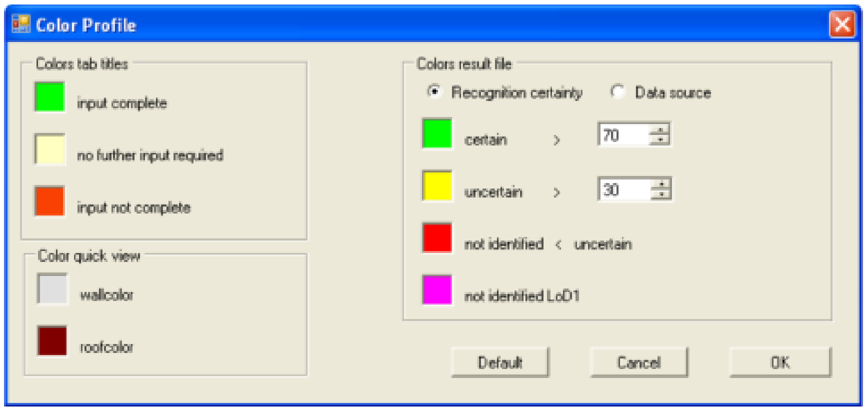

1147_33_7402_13_RGB.tif - [XY] - [1 : 4.00]
1148_33_7402_13_RGB.tif - [Z] - [1 : 4.00]

Figure 5.9: Recognition certainty of LOD2 Buildings, visualization by CityModeller

The color profile of the recognition certainty of detected roofs as well as the color of roofs and the walls of the displayed building may be set by the user in advance (Figure 5.10).


Color Profile

Colors tab titles
- 🟩 input complete
- 🟨 no further input required
- 🟥 input not complete

Color quick view
- ⬜ wallcolor
- 🟥 roofcolor

Colors result file
(● Recognition certainty) ○ Data source
- 🟩 certain > 70
- 🟨 uncertain > 30
- 🟥 not identified < uncertain
- 🟪 not identified LoD1

Figure 5.10: Color profile of recognition certainty

5.3.2 Editing LOD2 Building Models with 3D Editor

The model, which is automatically created in the previous step, consists of building models with standard roof structures (LOD2). In this step, the 3D Editor has been used to correcting LOD2 model where editing is required. Roof details such as a dormer, chimney, terrace into the model in order to get closer to reality. In addition to this, the roof overhang extraction has been accomplished for the couple buildings.

CORRECTING ROOF TYPES

The building in figure 5.11 has been generated by City Modeller as a building with flat roof type. In the reality, the building has cut gable roof with terrace. The generated model has been split into two parts so that each part can separately be modeled by selecting different predefined object types. One part has been regenerated as a terrace. So the roof type remained as the flat roof but the height of the roof has been changed. The second part has been converted to cut gable roof. Finally, the chimney has been modeled in order to get closer to as close to reality as possible. The result of the adjustment can be seen in figure 5.12.

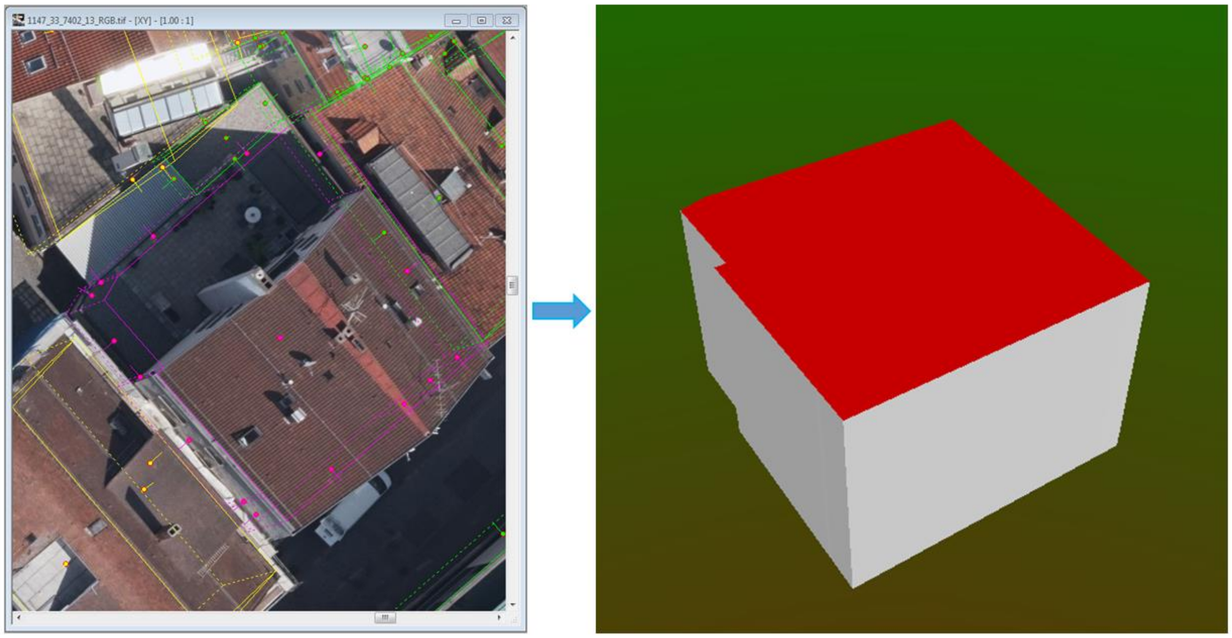


Figure 5.11: Building with flat roof is generated by City Modeller

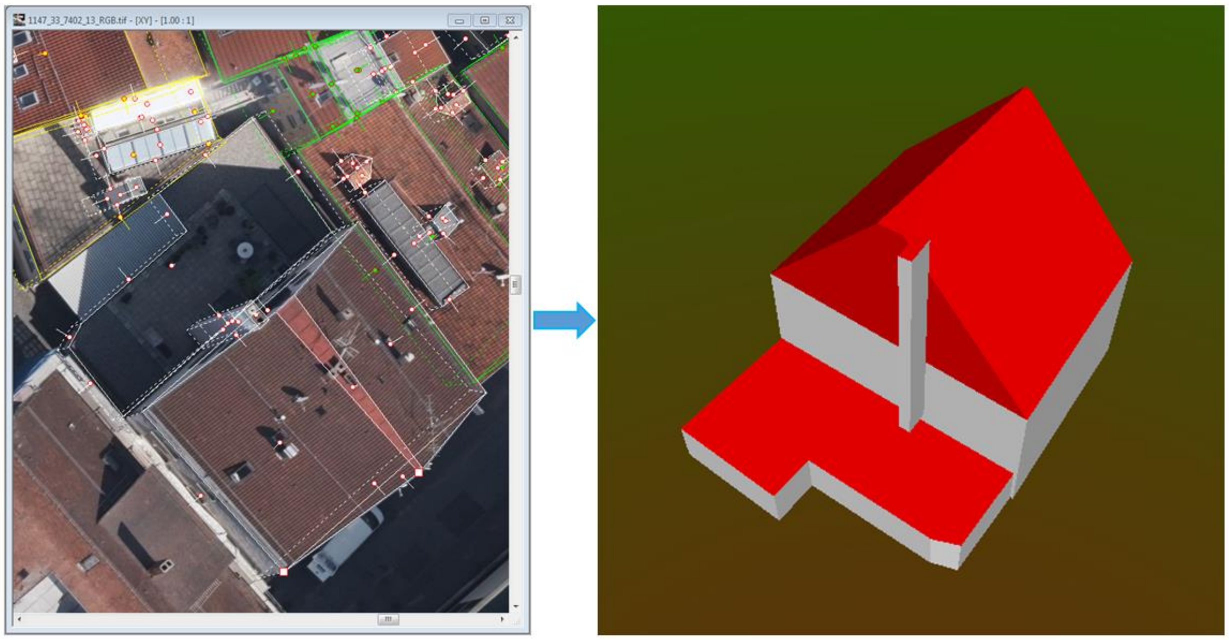


Figure 5.12: The model after adjustment

ROOF DETAILS GENERATION

The roof type of the building in Figure 5.13 is generated correctly. However, the outlines (eave lines) of the roof were shifted. Hence, the locations of the eave lines have been corrected and afterwards, the roof details has been generated by using different available object types such as lean-to roof, gable roof (Figure 5.14).

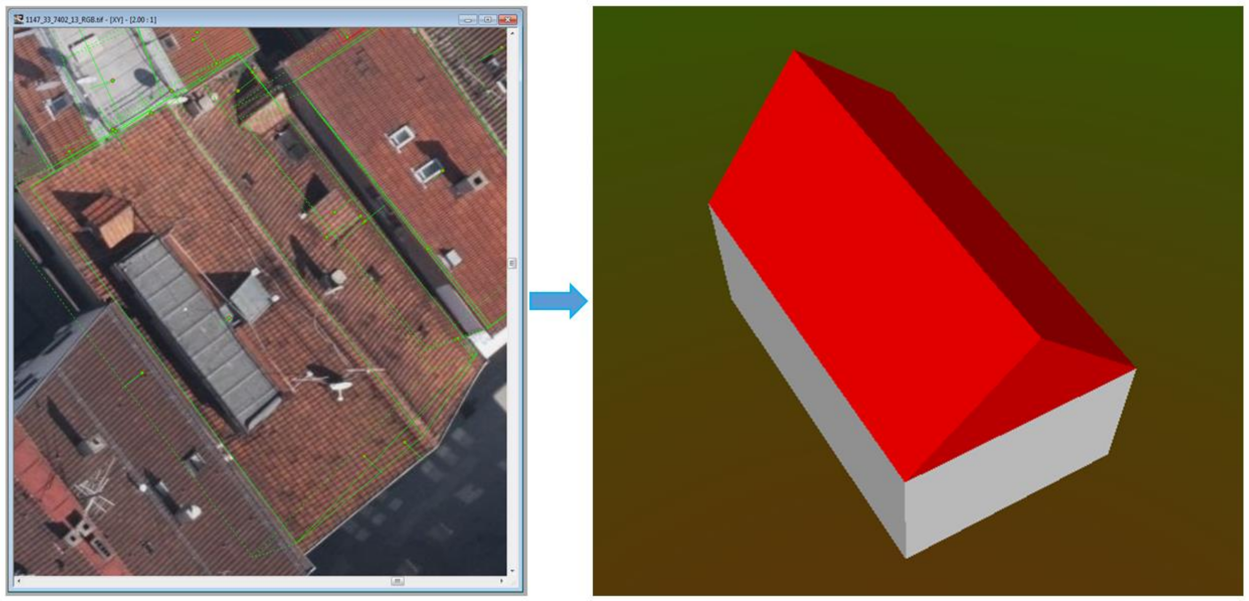


Figure 5.13: Building with gable roof generated by City Modeller

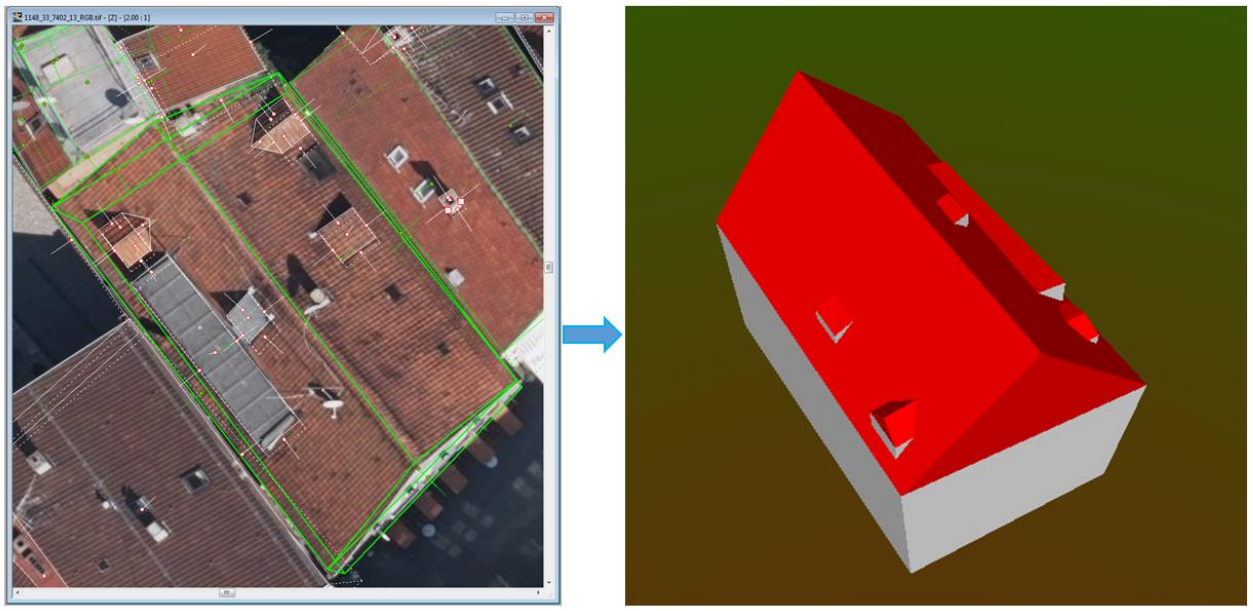


Figure 5.14: The model after adjustment

IRREGULAR BUILDINGS

Pattern recognition of CityModeller software is having difficulty with irregular buildings. The building in Figure in 5.15 has a roof type which is a combination of tent roof and irregular gambrel roof. CityModeller accomplished to split to the roof where two different roof types apply. However, the roof types could not be generated correctly. The front part of the building has been initially generated as a flat roof and it has been corrected by converting the flat roof into the polytent roof and the back part of the roof has been also converted from gable roof to gambrel roof (Figure 5.16). As the building has an irregular gambrel roof, all eave points in the constructional order of the object must be measured manually.

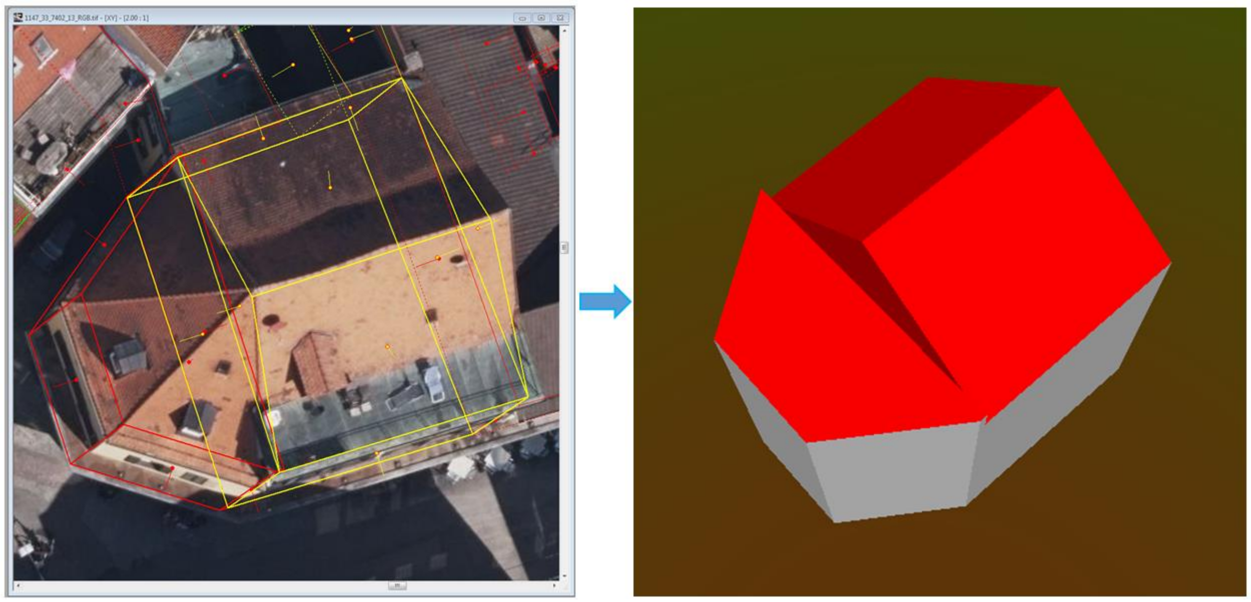


Figure 5.15: Building with flat roof and gable roof by CityModeller

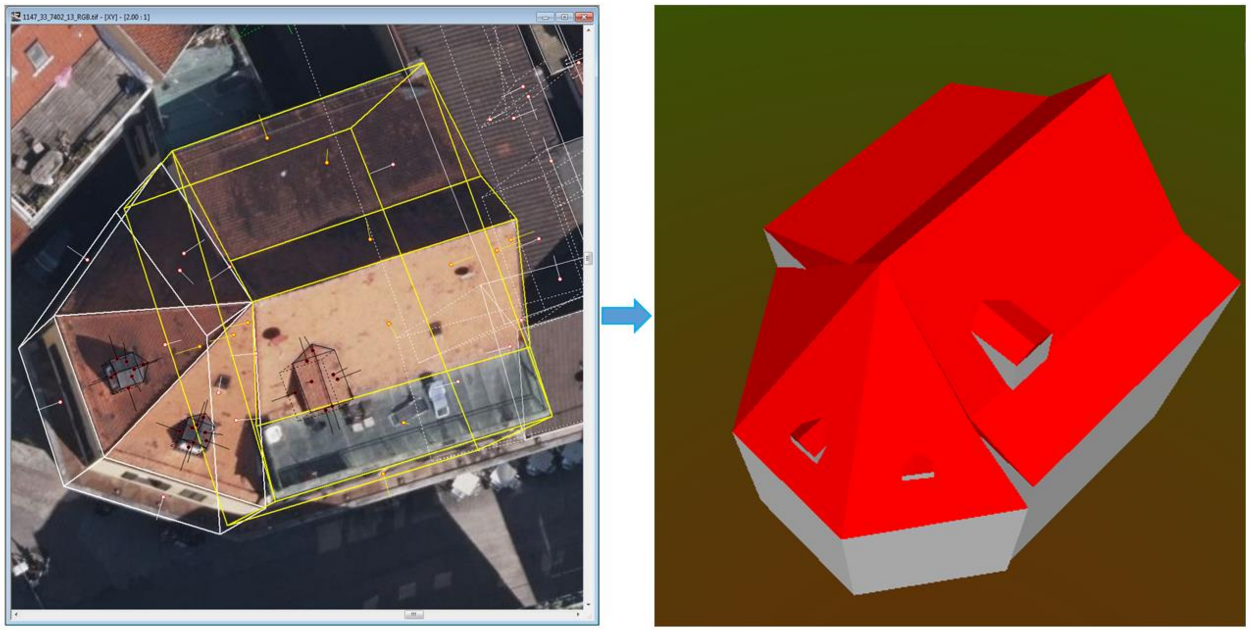


Figure 5.16: The model after adjustment

ROOF OVERHANG GENERATION

In the aerial images, roof overhangs are often obscured by the roof. Thus, 3D Editor cannot generate the overhangs from aerial images.

Export software, which is further explained in next section, has a component that enables exporting the model from op3d/tdc to CityGML. This component gives the possibility of intersecting 3D buildings with a 2D data source such as footprint. Footprints indicate the legal border of the outer walls of the building. CityGML Export module utilizes this information to generate roof overhang. If the roof outlines are bigger than the building footprints, the module cuts off the roof faces during intersection and generates the roof overhang (Figure 5.17).

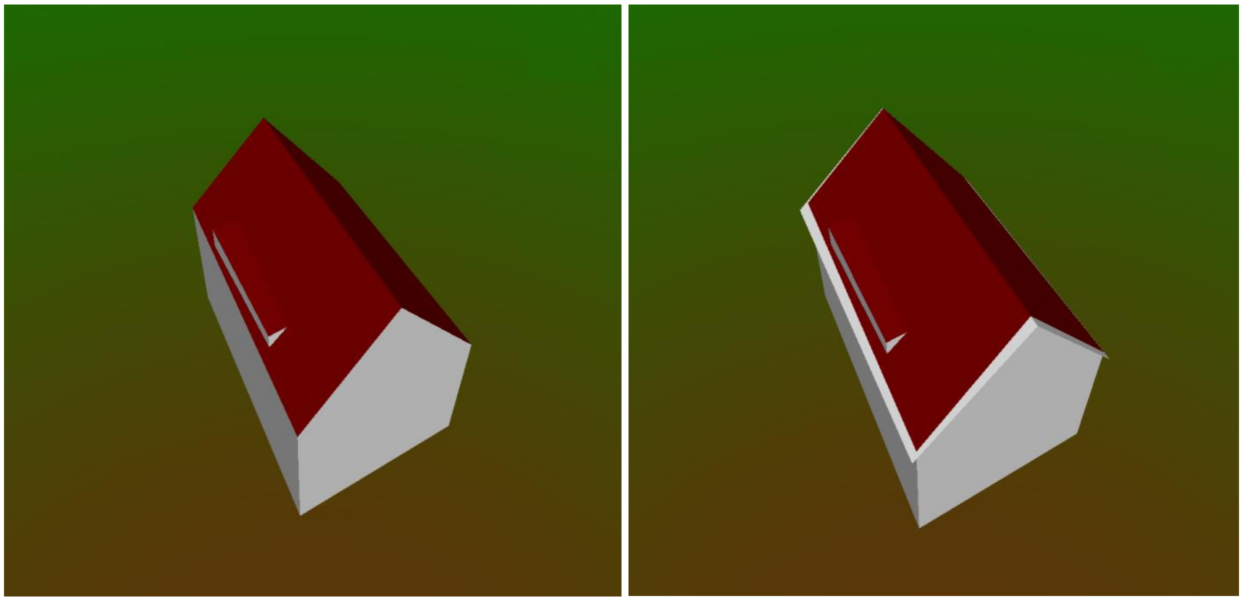
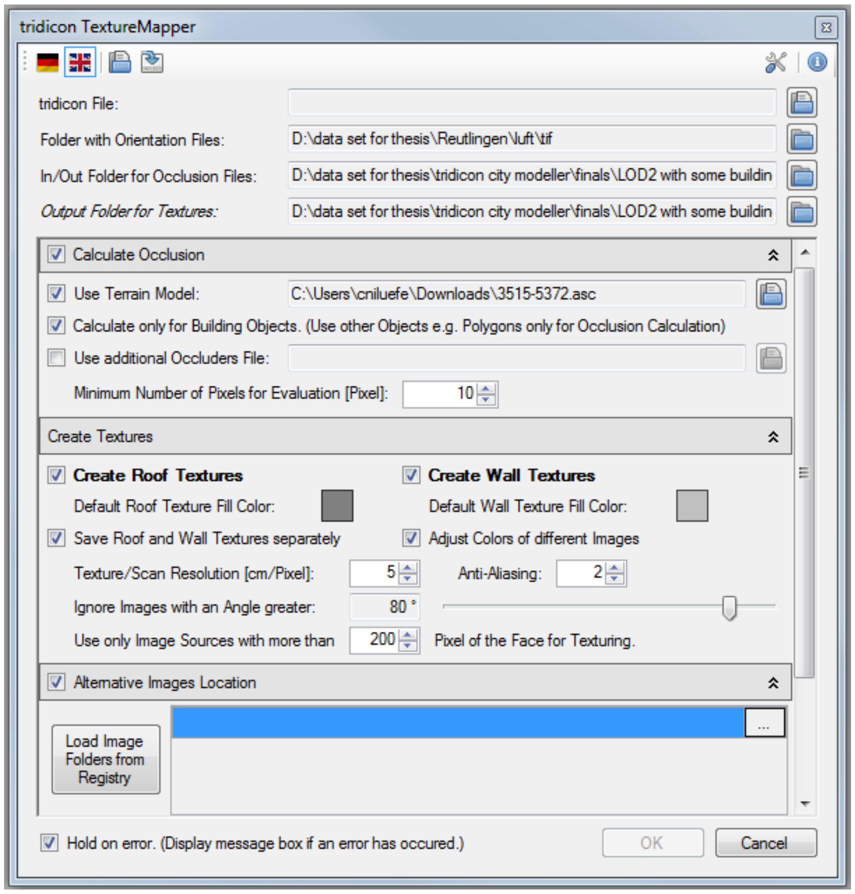


Figure 5.17: The building in op3d format (left), The building in CityGML format after interesting with footprint (right)

5.3.3 Texture Generation

The generated 3D city model is texturized by TextureMapper. TextureMapper is software that automatically textures the 3D building models from nadir or oblique aerial images.

The automatic texturing process of with the TextureMapper is divided into two sub-processes: occlusion calculation and determination of the texture. In the first subprocess, the program detects how much of which faces of the loaded 3D model is visible in which image. (TextureMapper Manual, 2006 -2015, S. 11). The program firstly eliminates the images on which the faces are visible at all and then beyond the images on which the faces are visible, the program calculates how many pixels of the image show a particular face and the incidence angle at which light strikes on the face of the model. In the second subprocess, textures for the faces are determined with respect to the result of occlusion calculation.


tridicon TextureMapper

tridicon File:

Folder with Orientation Files: D:\data set for thesis\Reutlingen\luft\tif

In/Out Folder for Occlusion Files: D:\data set for thesis\tridicon city modeller\finals\LOD2 with some buildin

Output Folder for Textures: D:\data set for thesis\tridicon city modeller\finals\LOD2 with some buildin

☑ Calculate Occlusion

☑ Use Terrain Model: C:\Users\cniluefe\Downloads\3515-5372.asc

☑ Calculate only for Building Objects. (Use other Objects e.g. Polygons only for Occlusion Calculation)

☐ Use additional Occluders File:

Minimum Number of Pixels for Evaluation [Pixel]: 10

Create Textures

☑ **Create Roof Textures**

Default Roof Texture Fill Color:

☑ Save Roof and Wall Textures separately

Texture/Scan Resolution [cm/Pixel]: 5

Ignore Images with an Angle greater: 80 °

Use only Image Sources with more than 200 Pixel of the Face for Texturing.

☑ **Create Wall Textures**

Default Wall Texture Fill Color:

☑ Adjust Colors of different Images

Anti-Aliasing: 2

☑ Alternative Images Location

Load Image Folders from Registry

☑ Hold on error. (Display message box if an error has occured.)

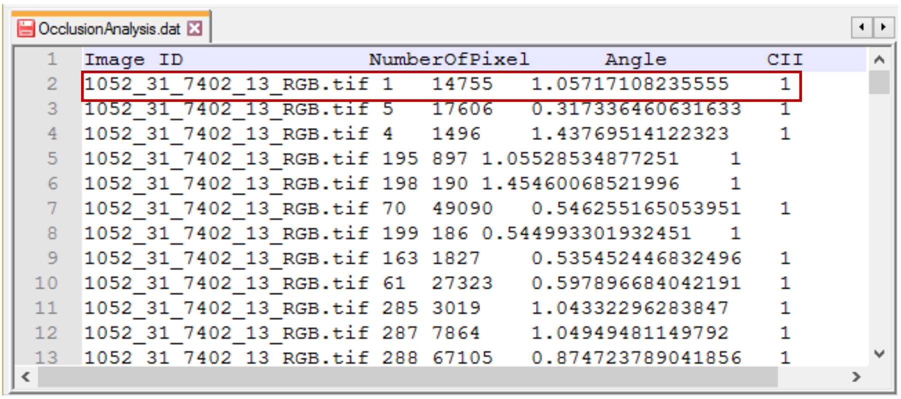
OK Cancel

Figure 5.18: User Interface of the TextureMapper

The texturing process starts by loading the required datasets to the program. As shown in figure 5.18, the path to the 3D building model file and to images with the orientation file must be provided. The following images formats are supported: *tiff, tif, vit, tga, png, jpg, jpeg, tdci, bmp, ads, jpe, psi, pmi, ljpg, dib,* and *img*. Additionally, digital terrain data may be added. The occlusion files and the folder for texture are created automatically where the model is stored.

Initially, the default values for different parameters are set in the user interfaces. Depending on the used images, the parameters such *Minimum number of pixels for a face*, *Texture Resolution* or *Maximum incidence angle* can be changed. TextureMapper allows separate texturing of walls and roof surfaces as well. Therefore, the parameter can also be set differently for the roof and wall.

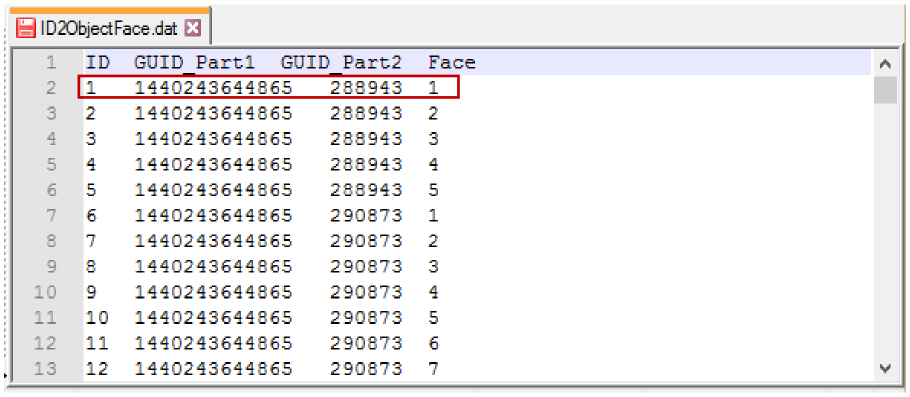
In the case study, the default values of the parameter are used, as they are often the optimal values. As first occlusion calculation has been completed and then textures have been created in png format. Some results of occlusion calculation can be seen in Figure 5.19. The column *Image* is the name of the images, *ID* is the id of the faces to be textured. *NumberOfPixel* indicates how many pixels of the images are visible on the face and *Angle* refers to the incident angle.



```
Image ID                    NumberOfPixel     Angle            CII
1052_31_7402_13_RGB.tif 1   14755   1.05717108235555    1
1052_31_7402_13_RGB.tif 5   17606   0.317336460631633   1
1052_31_7402_13_RGB.tif 4   1496    1.43769514122323    1
1052_31_7402_13_RGB.tif 195 897 1.05528534877251    1
1052_31_7402_13_RGB.tif 198 190 1.45460068521996    1
1052_31_7402_13_RGB.tif 70  49090   0.546255165053951   1
1052_31_7402_13_RGB.tif 199 186 0.544993301932451   1
1052_31_7402_13_RGB.tif 163 1827    0.535452446832496   1
1052_31_7402_13_RGB.tif 61  27323   0.597896684042191   1
1052_31_7402_13_RGB.tif 285 3019    1.04332296283847    1
1052_31_7402_13_RGB.tif 287 7864    1.04949481149792    1
1052_31_7402_13_RGB.tif 288 67105   0.874723789041856   1
```

Figure 5.19: Example results of occlusion calculation

In addition to this, another file (ID2ObjectFace.dat), which shows the generated texture for every face of the model, is created (Figure 5.20). *ID* column of this file enables to communicate with OcclusionAnalysis.dat file (Figure 5.19), *GUID* columns refer the corresponding building, and *Face* column is the identifier for the face in a particular building. The file name of the textures contains both parts of GUIDs and the Face number. By looking at those both files, one can see, which face will be textured by which image. As an example, the building with GUID 1440243644865_288943 has 5 faces.The face number ID 1 is visible in the image 1052_31_7402_13_RGB.tif. The face is covered by 14755 pixels of this image. Therefore, a texture is created by those pixels and is named as 1440243644865_288943_1.png.



```
ID  GUID_Part1    GUID_Part2   Face
1   1440243644865   288943   1
2   1440243644865   288943   2
3   1440243644865   288943   3
4   1440243644865   288943   4
5   1440243644865   288943   5
6   1440243644865   290873   1
7   1440243644865   290873   2
8   1440243644865   290873   3
9   1440243644865   290873   4
10  1440243644865   290873   5
11  1440243644865   290873   6
12  1440243644865   290873   7
```

Figure 5.20: Example results of occlusion calculation

The quality of texturing is highly dependent on both qualities of the images and orientation parameter. For instance, the wall faces can better be textured with oblique images or high overlapping images.

TextureMapper offers to work in batch process when series of the model need to be texturized.

5.3.4 Model Export

3D Editor generates the 3D building models in *op3d/tdc* format and enables exporting them as *dxf*, *vrml 97* and *shp format.*

Furthermore, the Export software makes possible the exporting of the data format *op3d/tdc* into various formats. The software consists of following export components:

- *tdc2CityGML:* Exports the model to CityGML format. The program also allows 3D building models to intersect with the 2D data source and a terrain model. Export of textured model is also supported. If there is no texture available, the model can be exported with synthetic solid colors. tdc2CityGML component enables working with batch mode.
- *tdc2Collada:* Exports the model to collada format.
- *tdc2EsriGDB:* Exports the model to Esri Geodatabase format. Both 32-bit and 64-bit version are available. The exporter allows specifying wall and roof color or searching for textures.
- *tdc2KML:* Exports the model to kml format.
- *tdc2KMZ:* Exports the model to kmz format. The model can be exported with natural colors.
- *tdc2myVR:* Exports the model to myVR format.
- *tdc2OBJ:* Exports the model to obj format. The model can be exported with natural colors.
- *tdc2Shape:* Exports the model to shp format. The model can be exported both as 3D or 2D shp file. Export with natural colors is available.
- *tdc2VRML:* Exports the model to VRML format.The model can be exported with textures. If there is no texture available, export of the model is possible with natural colors or random colors.

5.3.5 Results

The following sequence of figures (Figure 5.21 to 5.32) shows visualizations of the modelling results from CityModeller and 3D Editor.

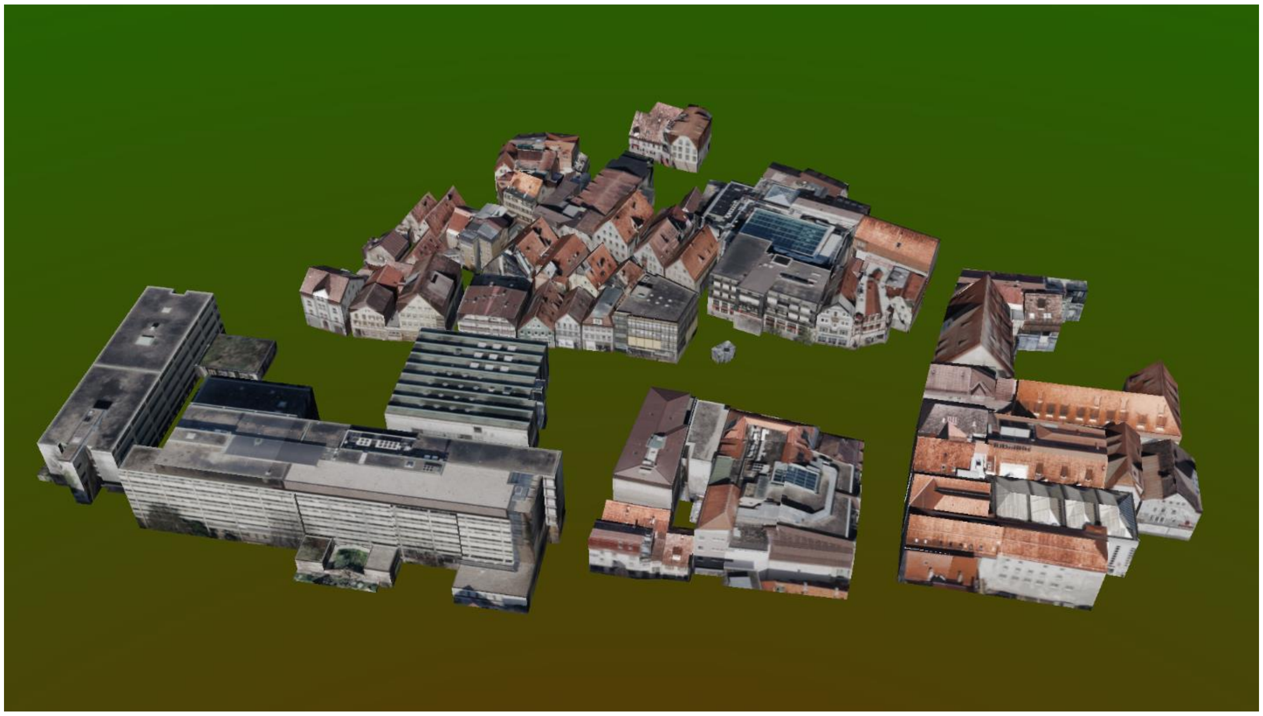


Figure 5.21: LOD2 Model generated by City Modeller and texturized by TextureMapper

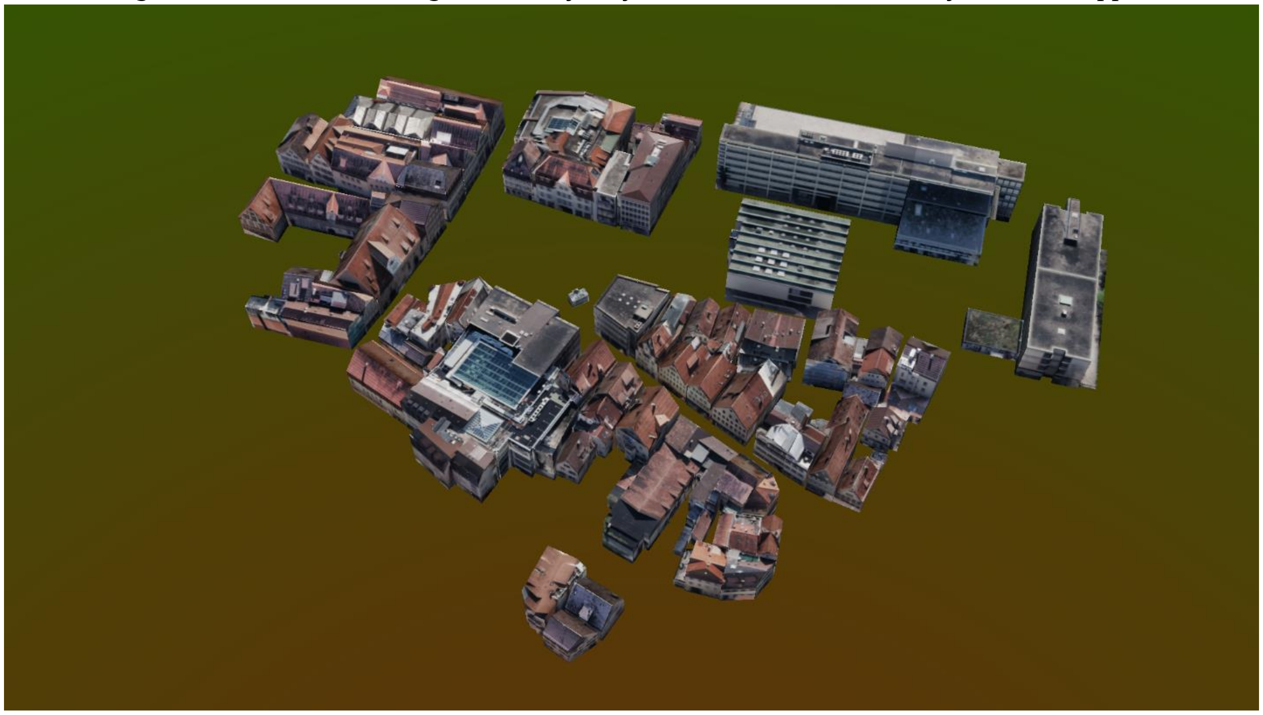


Figure 5.22 LOD2 Model generated by City Modeller and texturized by TextureMapper

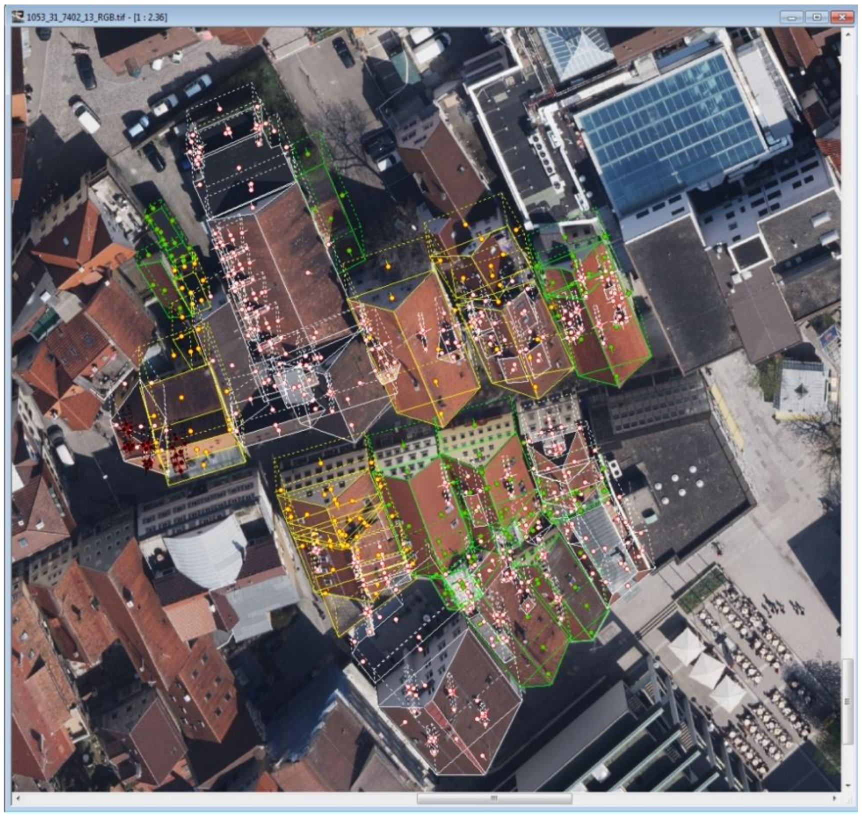


Figure 5.23: The modeled buildings in case study, visualization in City Modeller

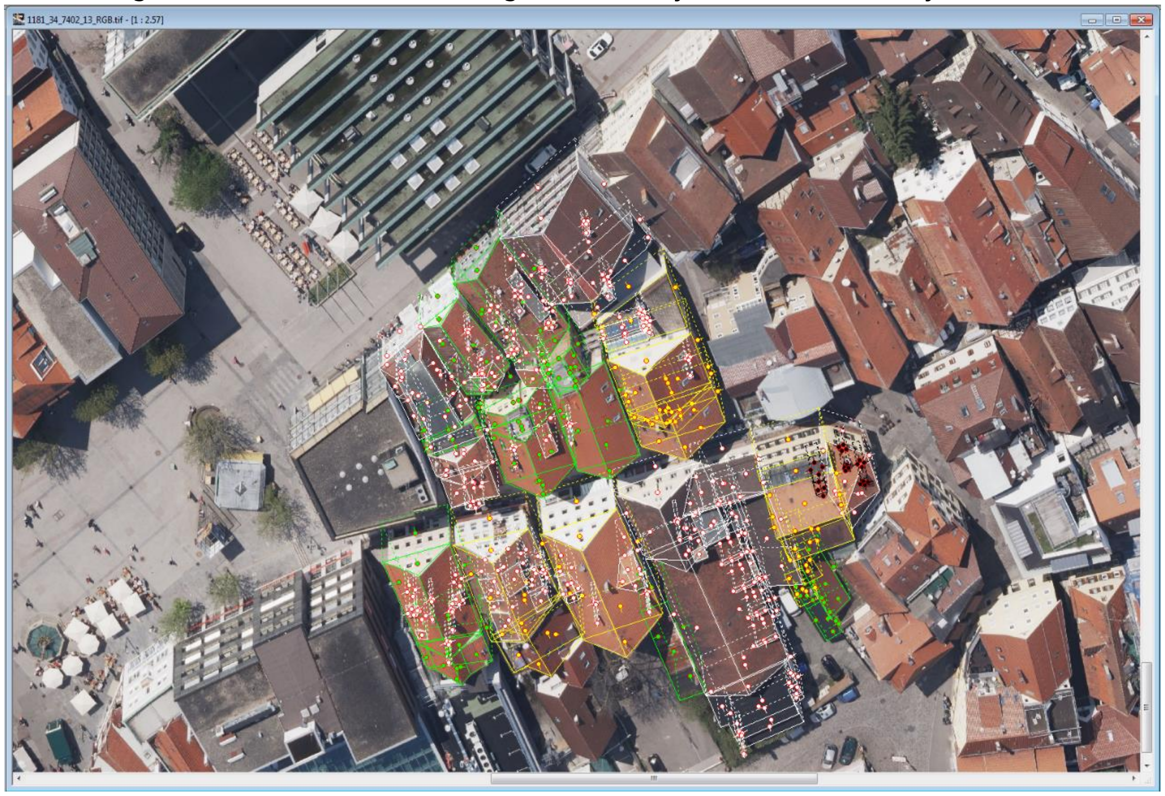


Figure 5.24: The modeled buildings in case study, visualization in City Modeller

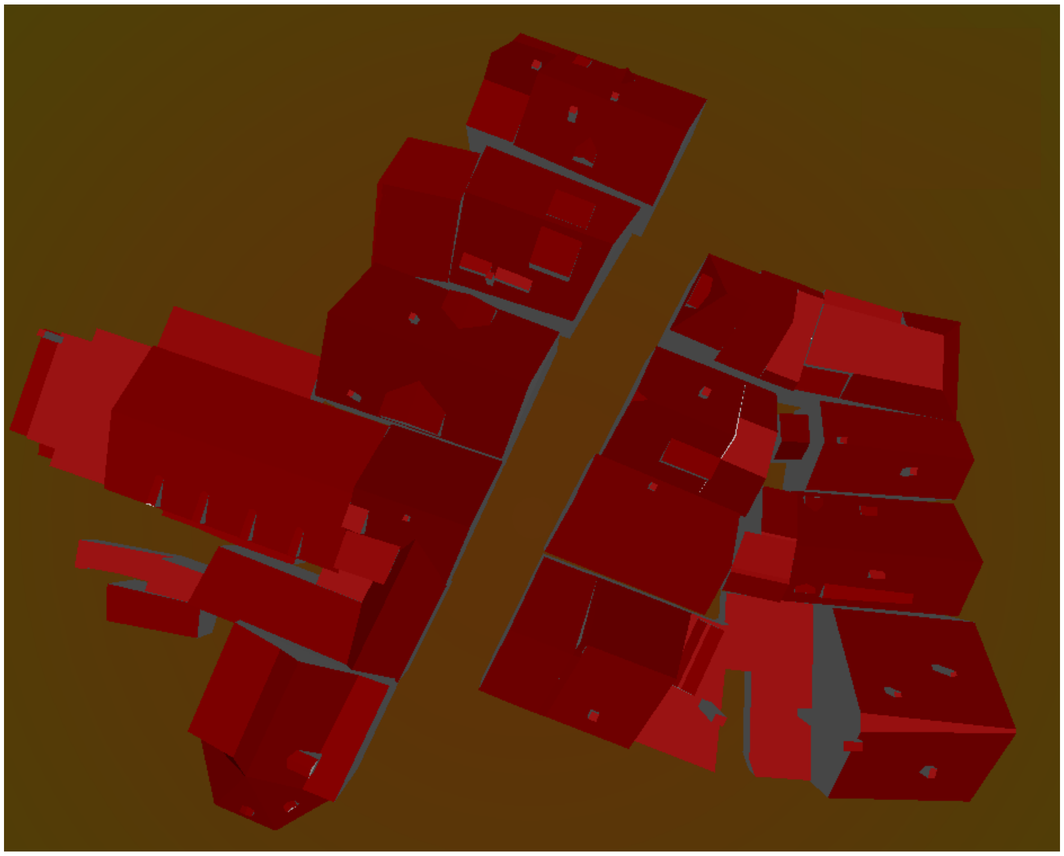
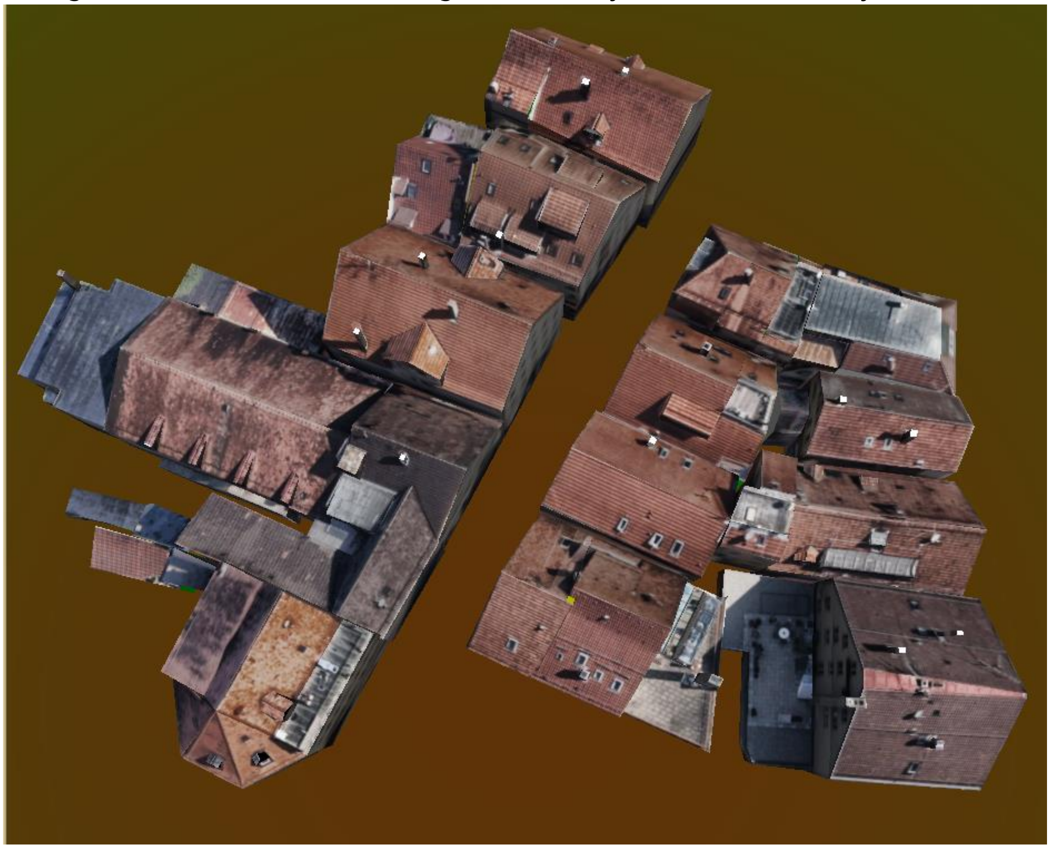


Figure 5.25: The modeled buildings in case study, visualization in CityDiscoverer



5.26: The modeled and texturized buildings in case study, visualization in CityDiscoverer

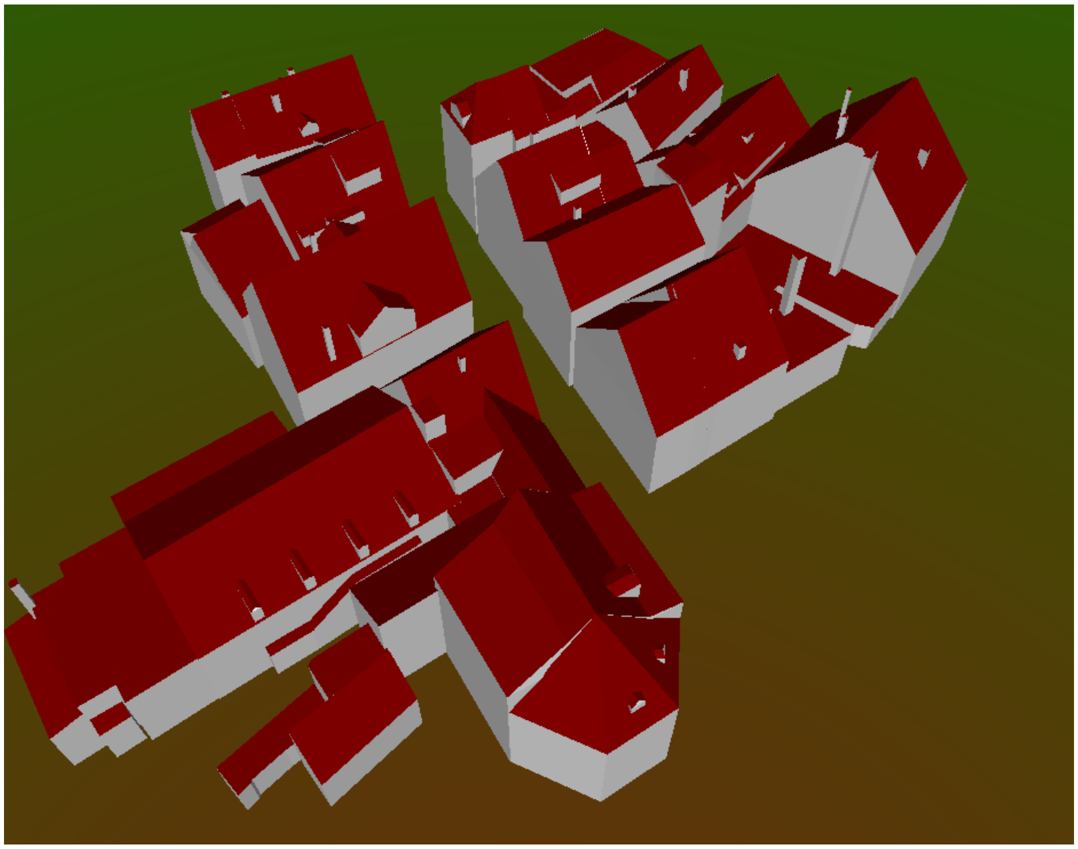


Figure 5.27: The modeled buildings in case study, visualization in CityDiscoverer

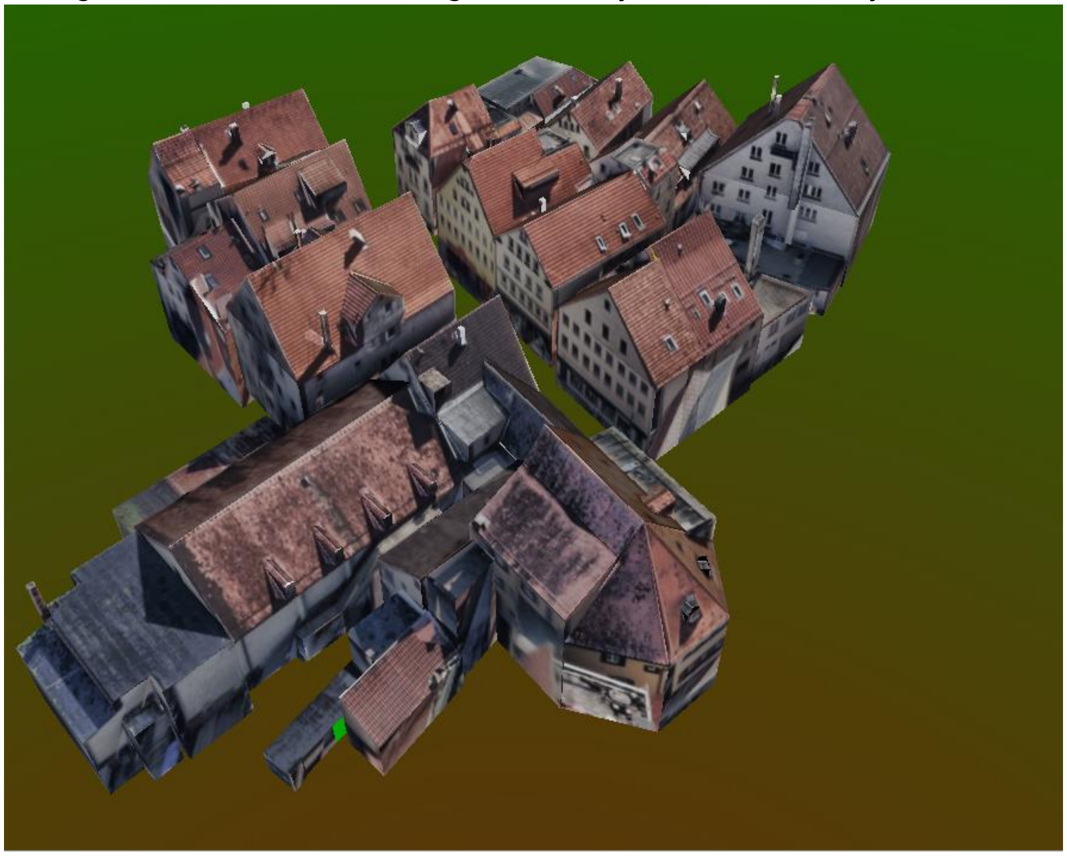


Figure 5.28: The modeled and texturized buildings in case study, visualization in CityDiscoverer

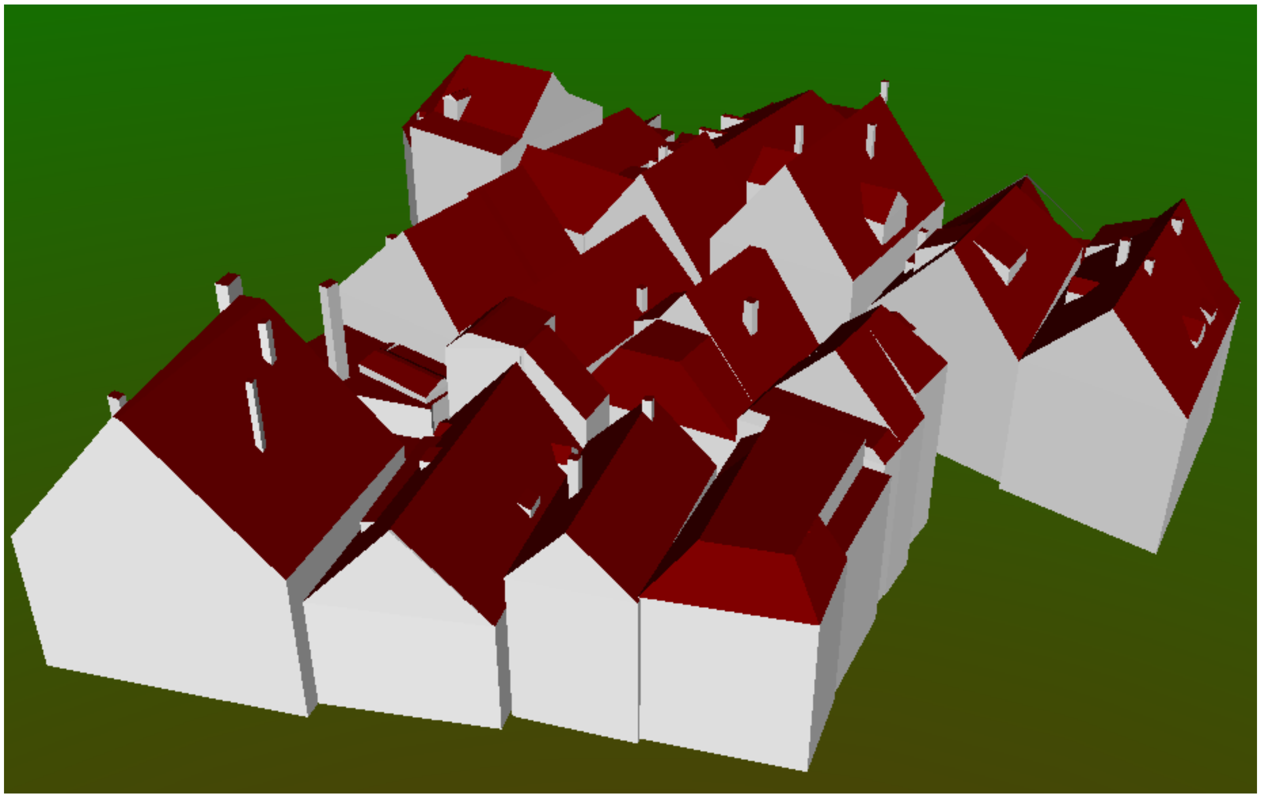


Figure 5.29: The modeled buildings in case study, visualization in CityDiscoverer

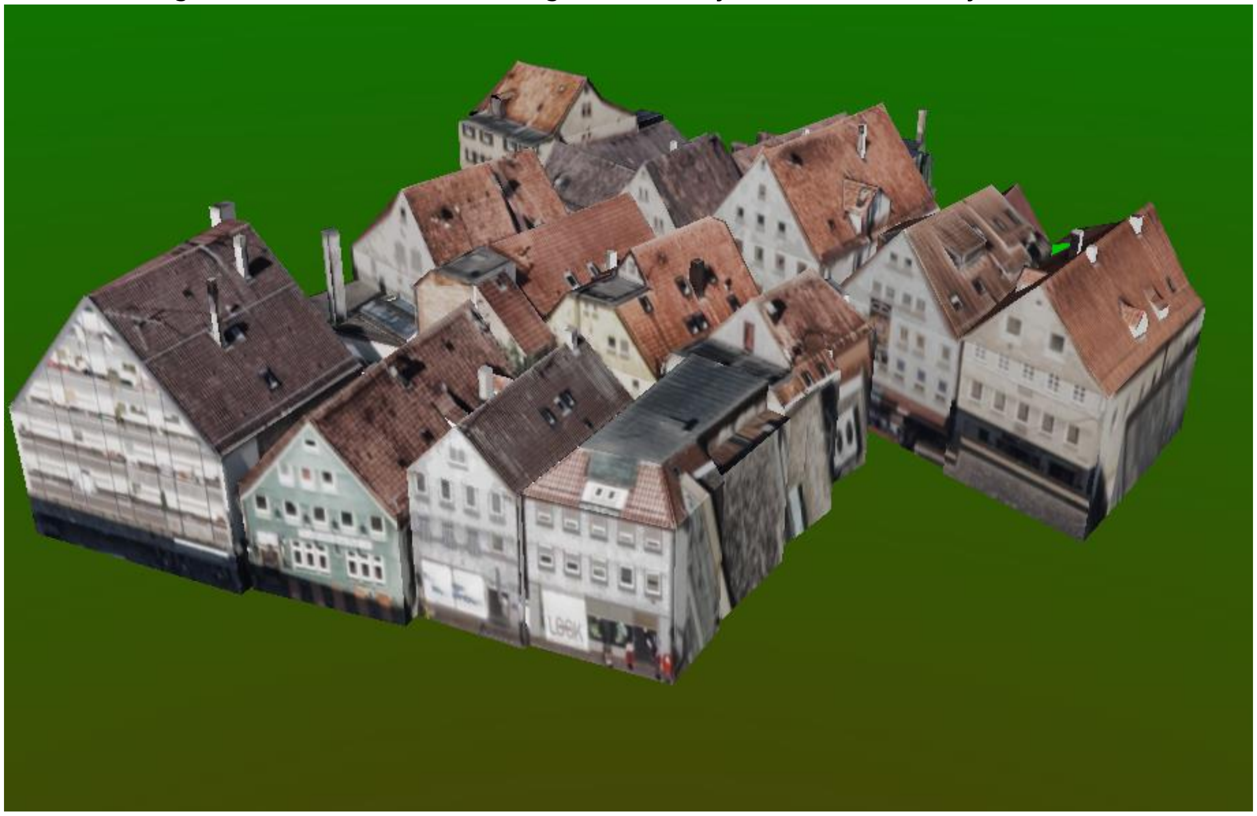


Figure 5.30: The modeled and texturized buildings in case study, visualization in CityDiscoverer

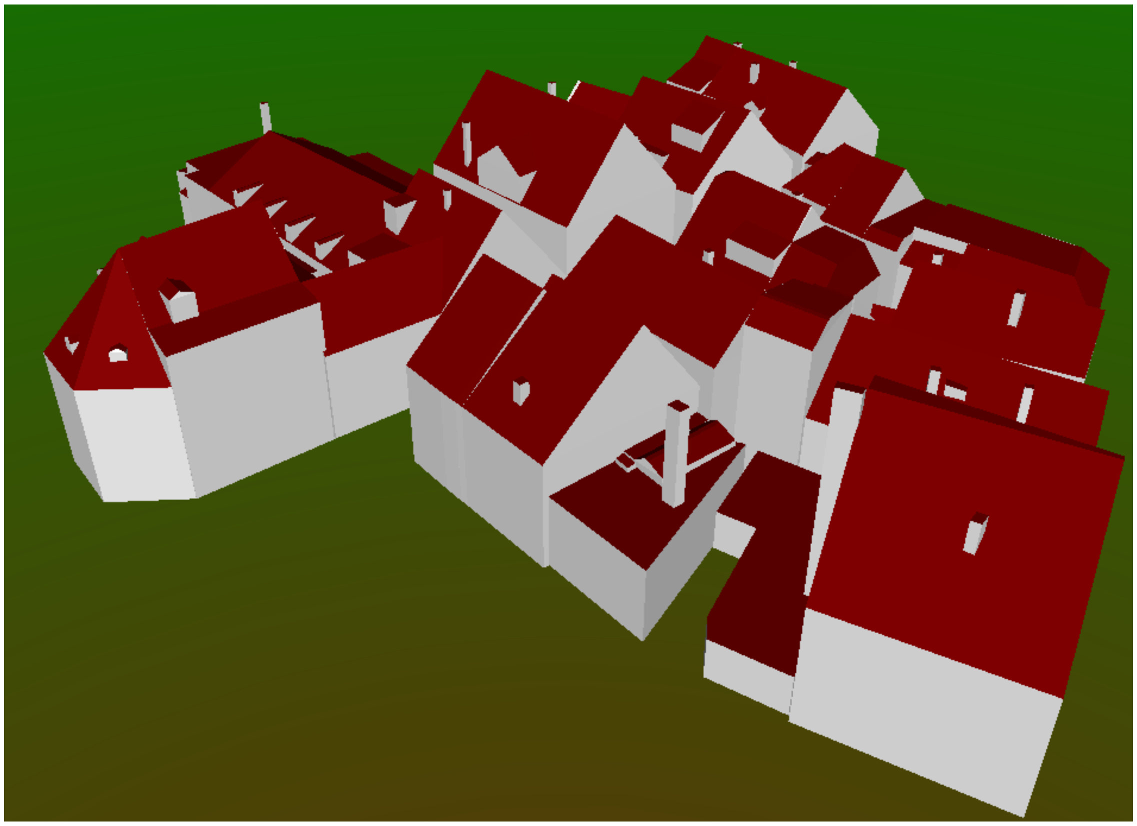


Figure 5.31: The modeled buildings in case study, visualization in CityDiscoverer

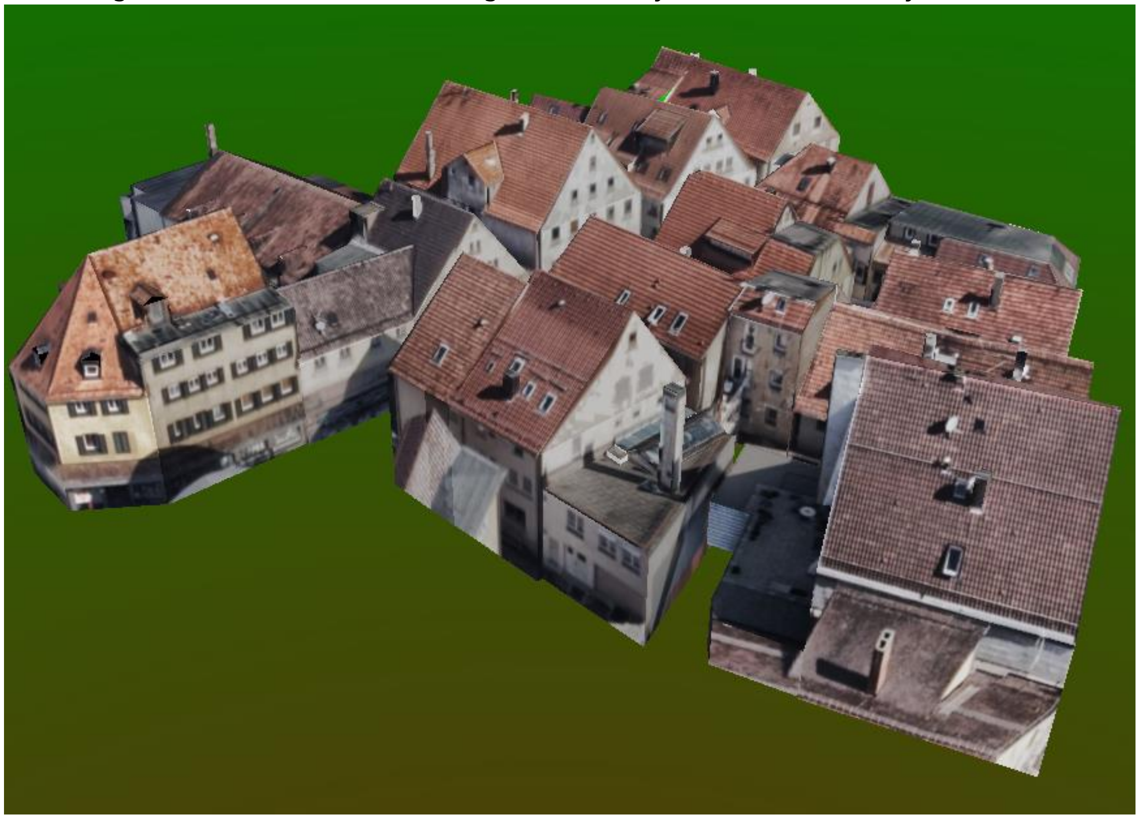


Figure 5.32: The modeled and texturized buildings in case study, visualization in CityDiscoverer

5.4 3D City Modeling with CityGRID® Modeler

In this section, the implementation process to generate a city model with CityGRID Modeler is described (Figure 5.33).

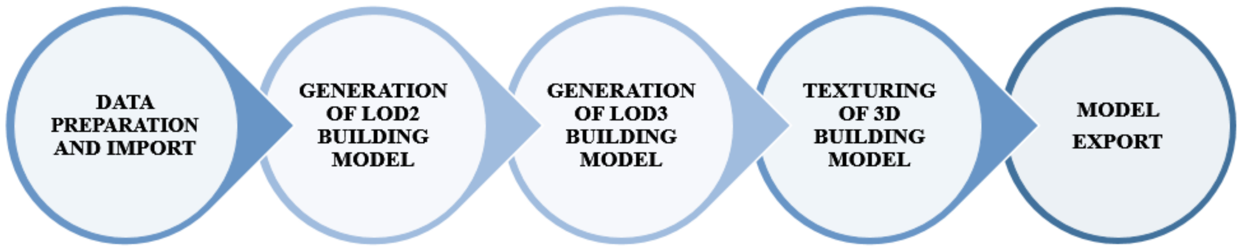

DATA PREPARATION AND IMPORT
GENERATION OF LOD2 BUILDING MODEL
GENERATION OF LOD3 BUILDING MODEL
TEXTURING OF 3D BUILDING MODEL
MODEL EXPORT

Figure 5.33: Implementation process

5.4.1 Preparation of the dataset

The core of CityGRID Modeler is the triangulation algorithm, which requires well-defined roof structures in order to create 3D surfaces correctly. Therefore, photogrammetric interpretation operator should consider of not only the geometry of the roof structure but also the semantics of roof structure. By taking this criterion into account, the roof line data set has been generated in *dwg* format with the help of photogrammetric compilation as it has been mentioned in Chapter 4. The basic data format in CityGRID is *xml*, hence the *dwg* format has been converted into in *xml* by use of CityGRID Converter. CityGRID Converter software helps to convert *cad* format into *xml* format. During the conversion, the Converter organizes each object semantically and hierarchically. For instance, outer eave, general roof lines, ridge, break line, dormer window or chimney are assigned to roof element whereas facade lines are assigned to facade element (Figure 5.34).

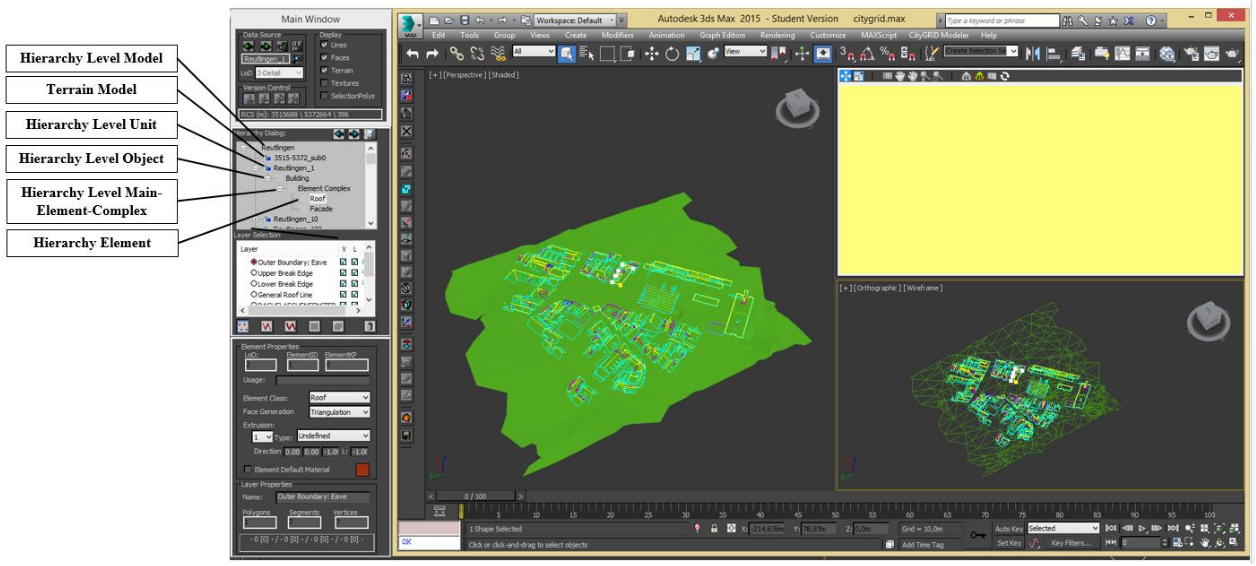


Figure 5.34: Visualization and hierarchic structure of the model after conversion to xml format

The Converter can also import a terrain model concurrently so that the buildings can get real height after the triangulation algorithm has been executed.

5.4.2 Model Generation

The surface of the models has been derived automatically from the lines by Triangulate tool in the Modeler (Figure 5.35). The tool triggers the 3D triangulation algorithm for generating face generation. Different face generation rules are applied to the roof, the facade, the protrusion, to detail elements and to overbuild objects of a building (i.e. roof details) (Systems, UVM, 2015, p. 12). The faces of the roof are generated by triangulation inside eave lines (outer boundary of roof shape). The eave lines must be a close polygon.

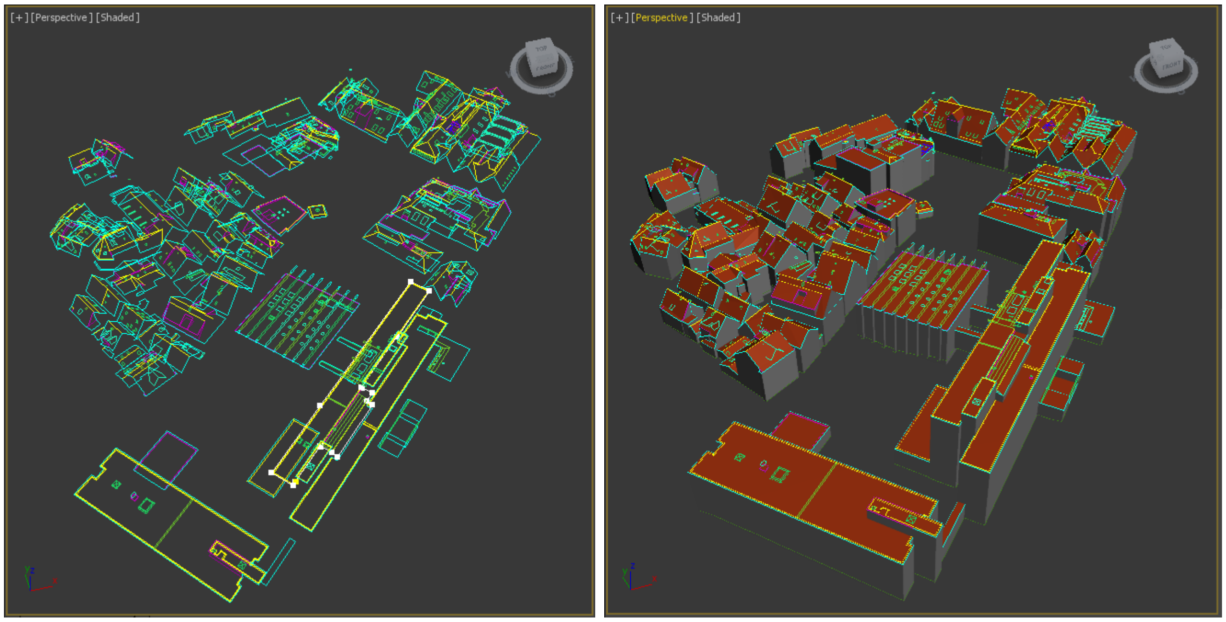


Figure 5.35: Roof and facade face generation from roof lines

The face of the facade of the buildings is generated horizontally from eave lines with a height of 50 m (default value). If the building model lies on a terrain model, the faces can be generated from eave lines to the lowest intersection point with the terrain.

EDITING THE GEOMETRY

Triangulation algorithm may generate faces incorrectly as a consequence of geometry error of building lines (Figure 5.36). According to the Modeler Manual, typical editing tasks are to merge non-snapped vertices, adjust intersection lines, complete missing lines and detach lines to Detail elements. If a loaded unit has such geometry error, a warning window comes out after the unit is triangulated. The warning window indicates geometry errors and possible solutions.

The buildings in case study have been checked against such error. Lines with error have been edited in such cases until the face generation algorithm creates correct faces. For instance, the upper break line does not end in yellow colored roof line in figure 5.37, therefore the roof surface has a gap. The problem has been solved by moving the end point of the upper break lines. The editing the process finishes by triangulating the building (Figure 5.38).

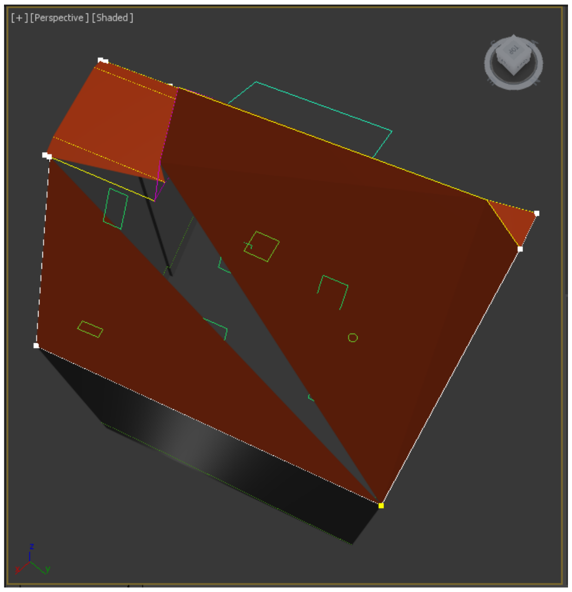


Figure 5.36: Incorrect roof face

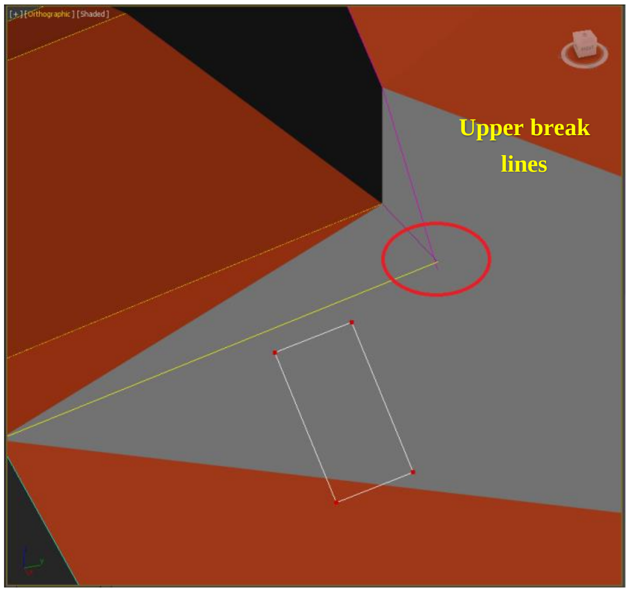


Figure 5.37: Geometry error of roof lines

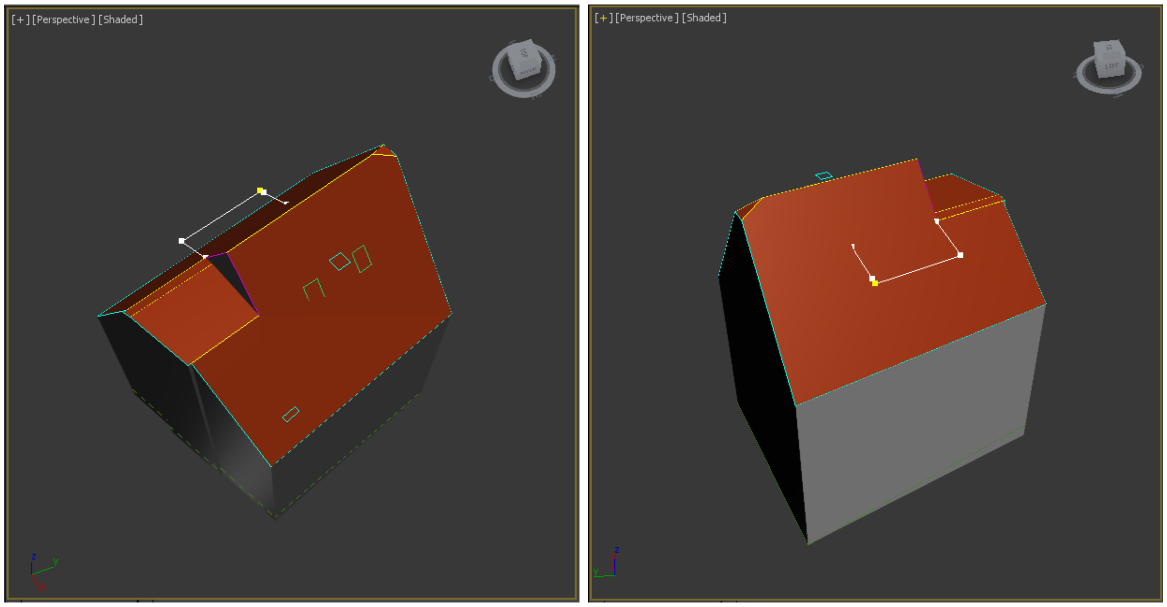
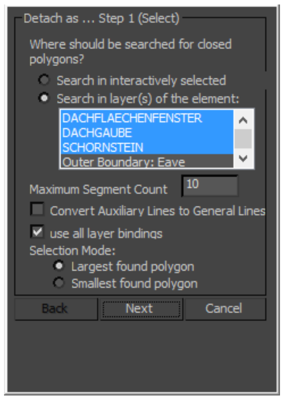


Figure 5.38: The building model after correction

ROOF DETAILS GENERATION

The special editing tool "Detach new complexes" allows generating roof details such as dormers, chimney. The lines, which are to be detached as roof details, must have a closed outline, as the automatic algorithm searches for closed polygons in the given layer or selected lines (Figure 5.39). The lines, which lie on those closed polygons, are suggested as Detail Element Complex. If suggested closed polygon is reasonable, the faces of the roof details can be generated by Triangulate tool. The triangulation algorithm extrudes facade down to the corresponding roof. In Figure 5.40 and 5.41, the buildings before and after roof detail generation can be seen.


Detach as ... Step 1 (Select)

Where should be searched for closed polygons?

- Search in interactively selected
- [✓ Search in layer(s) of the element:]
  - DACHFLAECHENFENSTER
  - DACHGAUBE
  - SCHORNSTEIN
  - Outer Boundary: Eave

Maximum Segment Count 10

☐ Convert Auxiliary Lines to General Lines

☑ use all layer bindings

Selection Mode:

- [✓ Largest found polygon]
- Smallest found polygon

Figure 5.39: Detach new complexes properties windows. In blue highlighted layers, the closed polygons are searched.

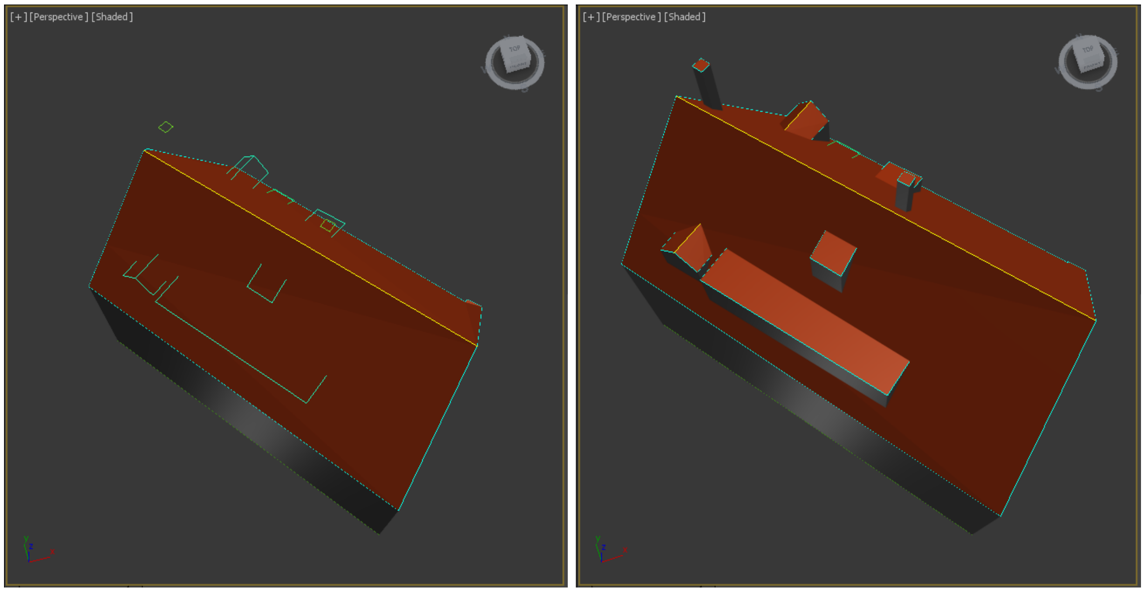


Figure 5.40: The building before and after roof detail generation

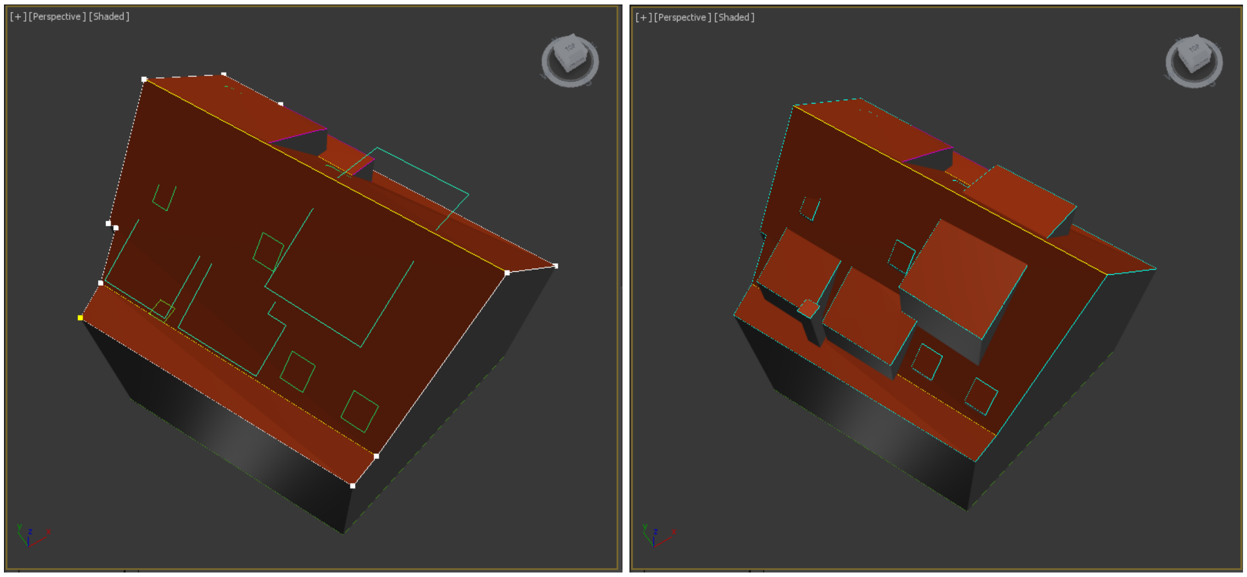
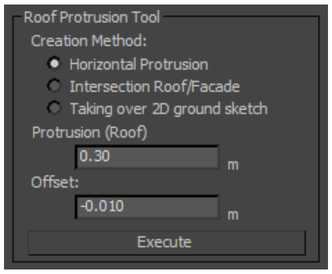


Figure 5.41: The building before and after roof detail generation

ROOF OVERHANG GENERATION

Roof Protrusion Tools enables to separate eave lines and the upper edge of the wall so that roof overhangs can be generated. The tool offers different methods for calculation of roof overhangs (Figure 5.42). *Horizontal Protrusion* is the first method which the selected line segments will be shifted respectively the given amount. The second method is *Intersection Roof/Facade*. The methods shift the selected segment with respective to the given amount and intersect vertically with the roof. The last method is *Taking over the 2D ground sketch*. The method intersects the given 2D line information, that is stored as a separate layer, with a roof, where it´s lying inside of the eave line.


Roof Protrusion Tool

Creation Method:

[✓ Horizontal Protrusion]

Intersection Roof/Facade

Taking over 2D ground sketch

Protrusion (Roof)

0.30 m

Offset:

-0.010 m

Execute

Figure 5.42: Roof Protrusion Tools Windows and the offered methods

Taking over 2D ground sketch method has been carried out in the case study to generate roof overhang (Figure 5.43, 5.44). In order to accomplish this method, the footprint as 2D ground information had to correctly be matched with 3D buildings. This process requires further software to handle such as Feature Manipulation Engine (FME) software[9].

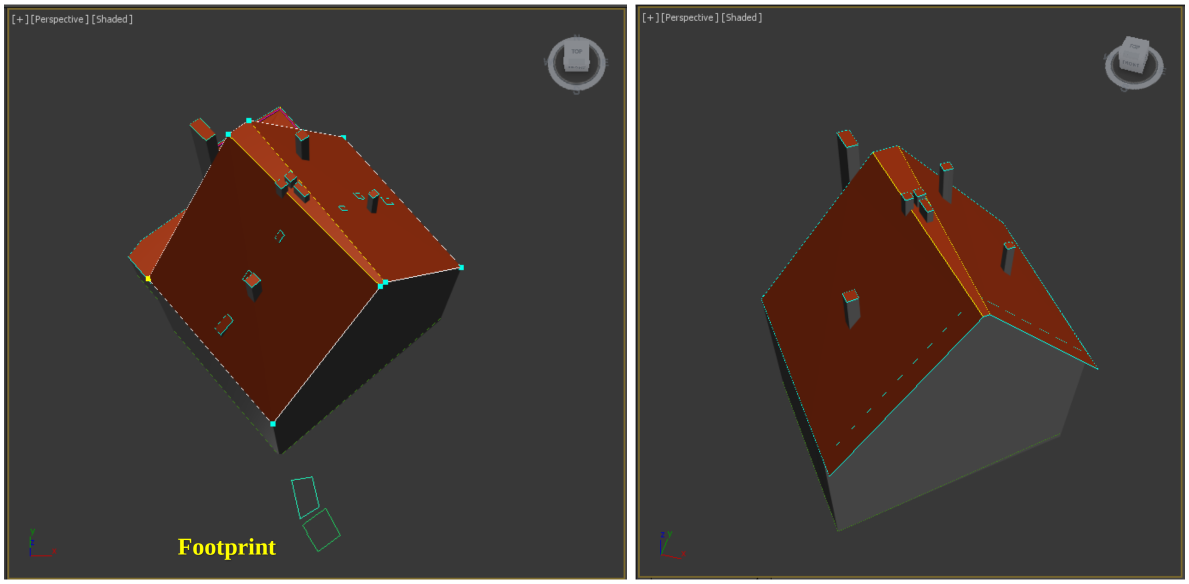


Figure 5.43: The building before and after roof overhang generation

[9] FME software is a data manipulation and transformation tool. (See http://www.safe.com/; last access December 2016).

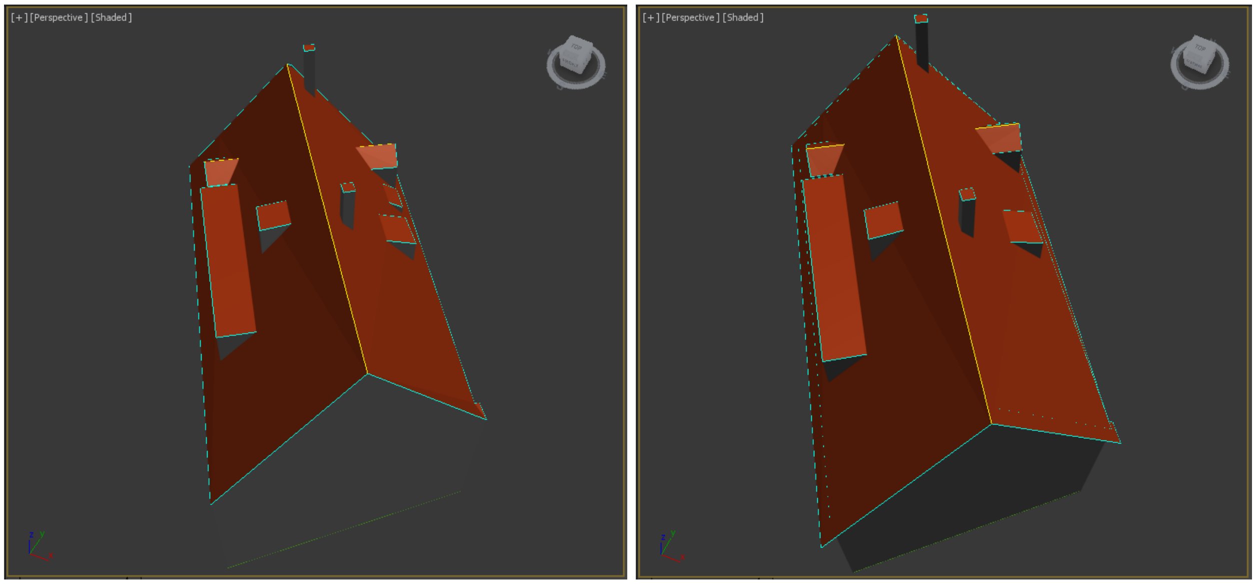
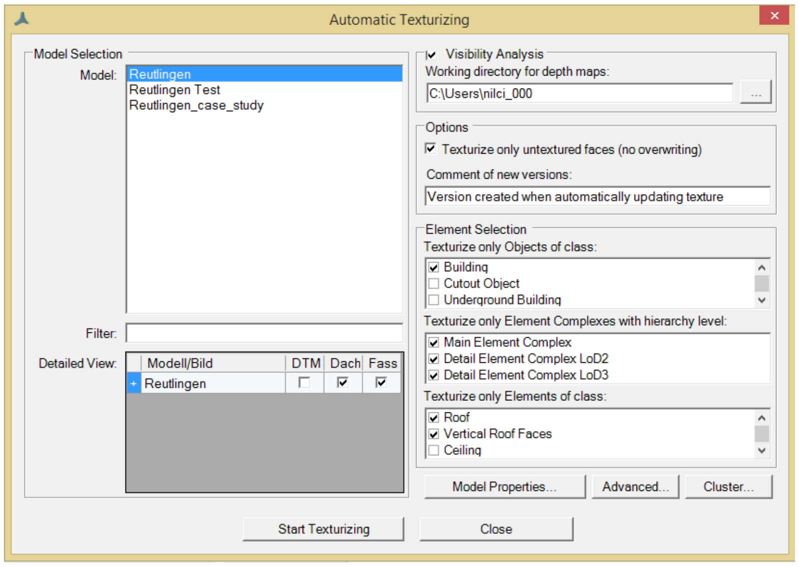


Figure 5.44: The building before and after roof overhang generation

5.4.3 Texture Generation

CityGRID Administrator has been utilized to automatically texturize the model. CityGRID Administrator is a UVM product for data administration. The program establishes a connection to a database (MS SQL or Oracle), which enables to import the models into database and export the data in various format. Furthermore, automatic building texturing is introduced by this program (Figure 5.45). The pre-requisites for automatic texturing are geo-referenced aerial images that are captured at the high overlap. Throughout, the program searches ideal aerial image for a specific polygon to texturize. It is done for roof and wall element separately.

Texturing has been started by importing the building model into the database via CityGRID Administrator. The images are not directly stored in the database but in specified texture folder. An xml file has been created, which refer to the path to aerial images as well as their inner and outer orientation parameter. Based on the parameter, the program finds automatically faces with corresponding parts of aerial images. The result of the automatically texturing process can be seen in Section 5.2.5.



Automatic Texturizing

Model Selection

Model: Reutlingen, Reutlingen Test, Reutlingen_case_study

Filter:

Detailed View:

Modell/Bild	DTM	Dach	Fass
Reutlingen	☐	☑	☑

☑ Visibility Analysis

Working directory for depth maps: C:\Users\nilci_000

Options

☑ Texturize only untextured faces (no overwriting)

Comment of new versions: Version created when automatically updating texture

Element Selection

Texturize only Objects of class:

- ☑ Building
- ☐ Cutout Object
- ☐ Underground Building

Texturize only Element Complexes with hierarchy level:

- ☑ Main Element Complex
- ☑ Detail Element Complex LoD2
- ☑ Detail Element Complex LoD3

Texturize only Elements of class:

- ☑ Roof
- ☑ Vertical Roof Faces
- ☐ Ceiling

Figure 5.45: *Automatic Texturizing Window of CityGRID Administrator*

There are two other possibilities for texturing the model directly in CityGRID Modeler: interactively texturing via freely-taken picture and semi-automatically texturing via oriented images.

For interactively texturing, the face to be texturized as well as the images to be mapped must be selected. In the texture window (yellow window in Figure 5.46), the mesh of the geometry of the face and texture image will be combined with selection.

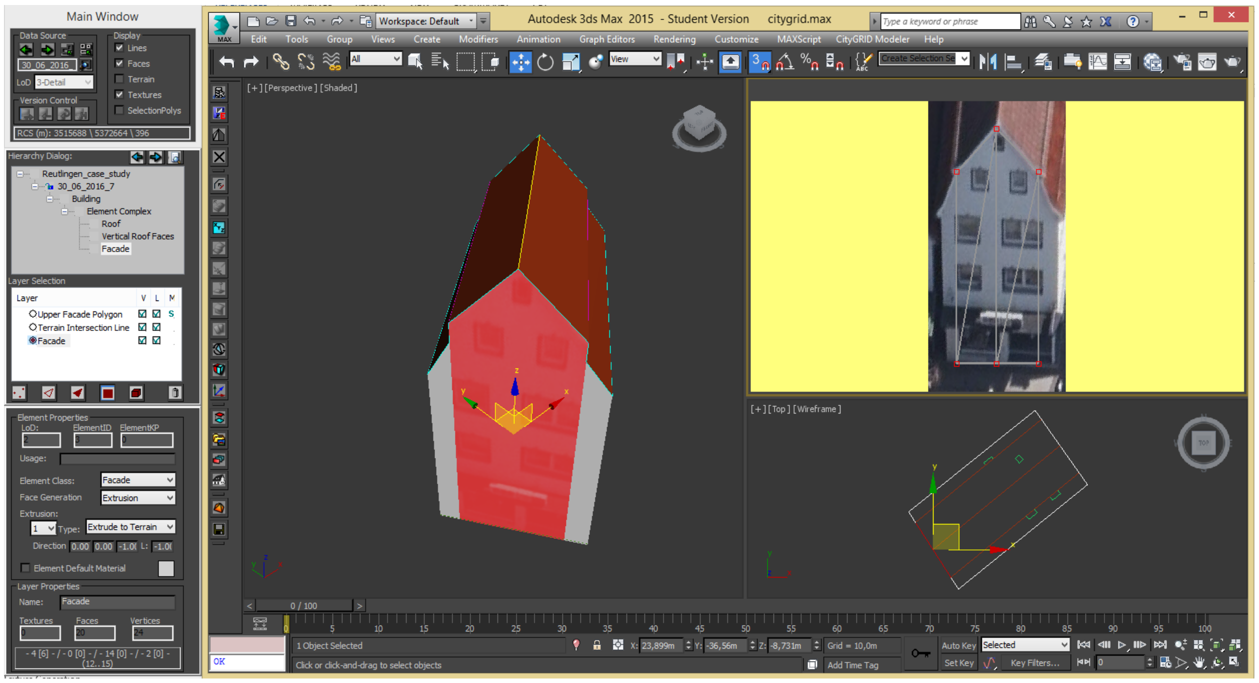


Figure 5.46: *Interactive Texturizing in CityGRID Modeler*

The mesh always has four points which represent the vertices of the selected face. In the texture window, the points of mesh should be dragged to the corresponding positions in the texture image and the final positions of the points should be saved so that the texture coordinates will be saved in the data source.

For the semi-automatically texturing approach, the images that will be used for the texture must be imported with their perspective orientation parameter and/ or with their geo-reference. Based on this,selected faces will be mapped semi-automatically with oriented images.

5.4.4 Model Export

The models are generated in xml format by CityGRID. However, the program enables exporting the model into followings formats*: wrl*, *dxf*, *kml*, and *CityGML*. If the model is stored in the database, there is a possibility to export model directly from the database into *wrl*, *dxf*, *kml*, and *CityGML* format via CityGRID Administrator.

5.4.5 Results

The following sequence of figures (Figure 5.47 to 5.54) shows visualizations of the modelling results from CityGRID Modeler.

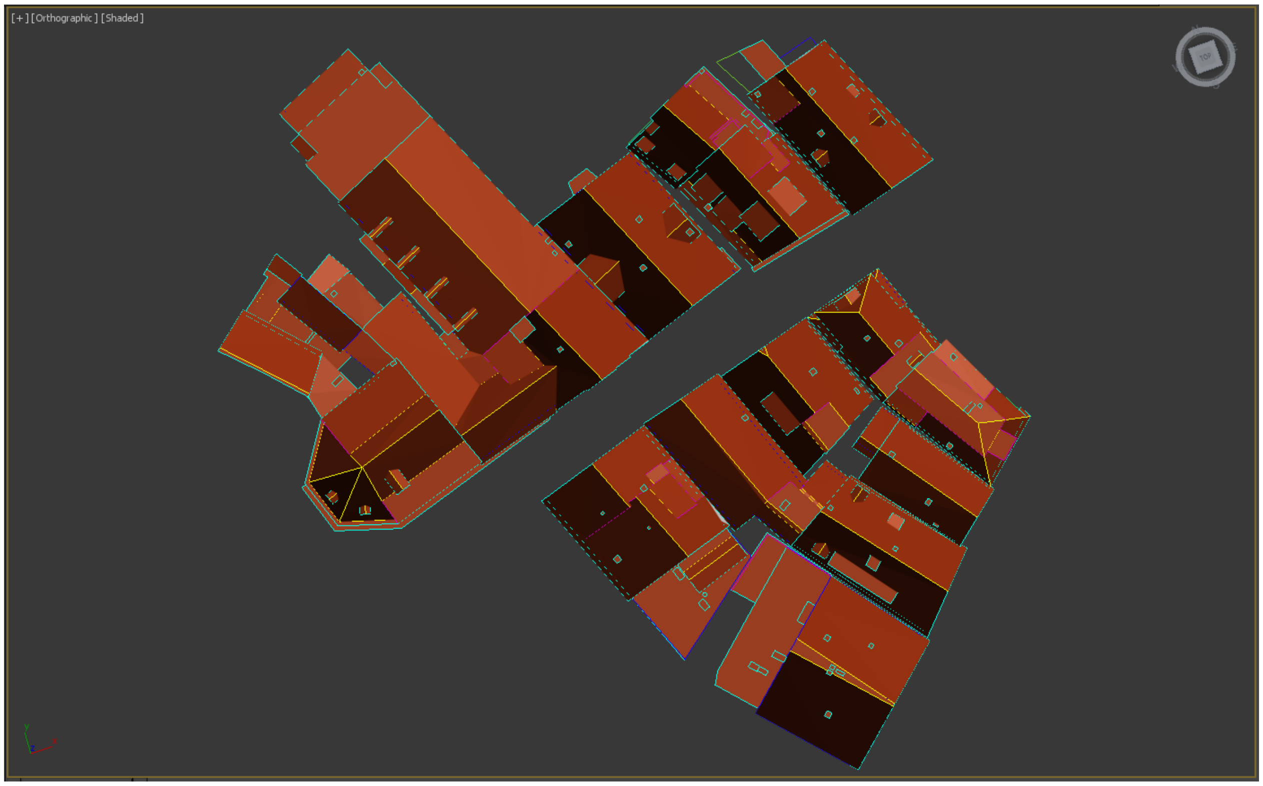


Figure 5.47: The modeled buildings in case study, visualization in Autodesk 3ds Max

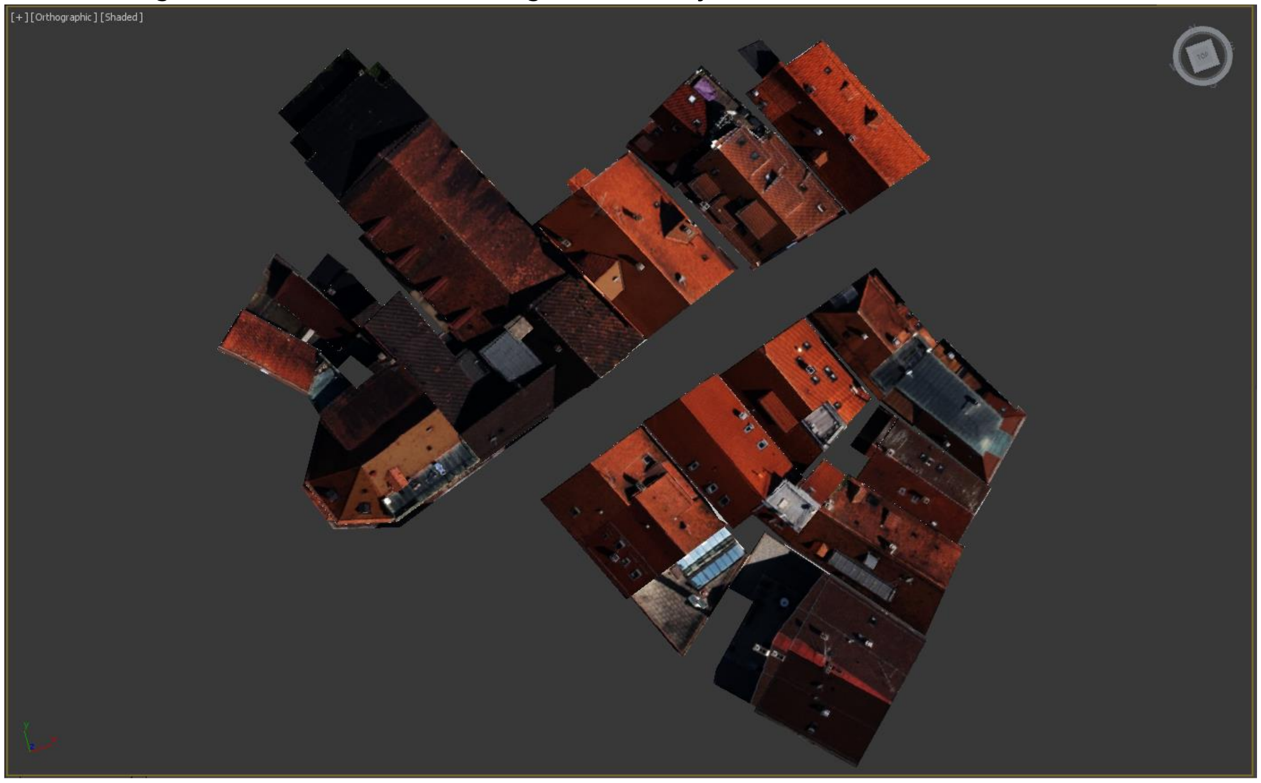


Figure 5.48: The modeled and texturized buildings in case study, visualization in Autodesk 3D Max

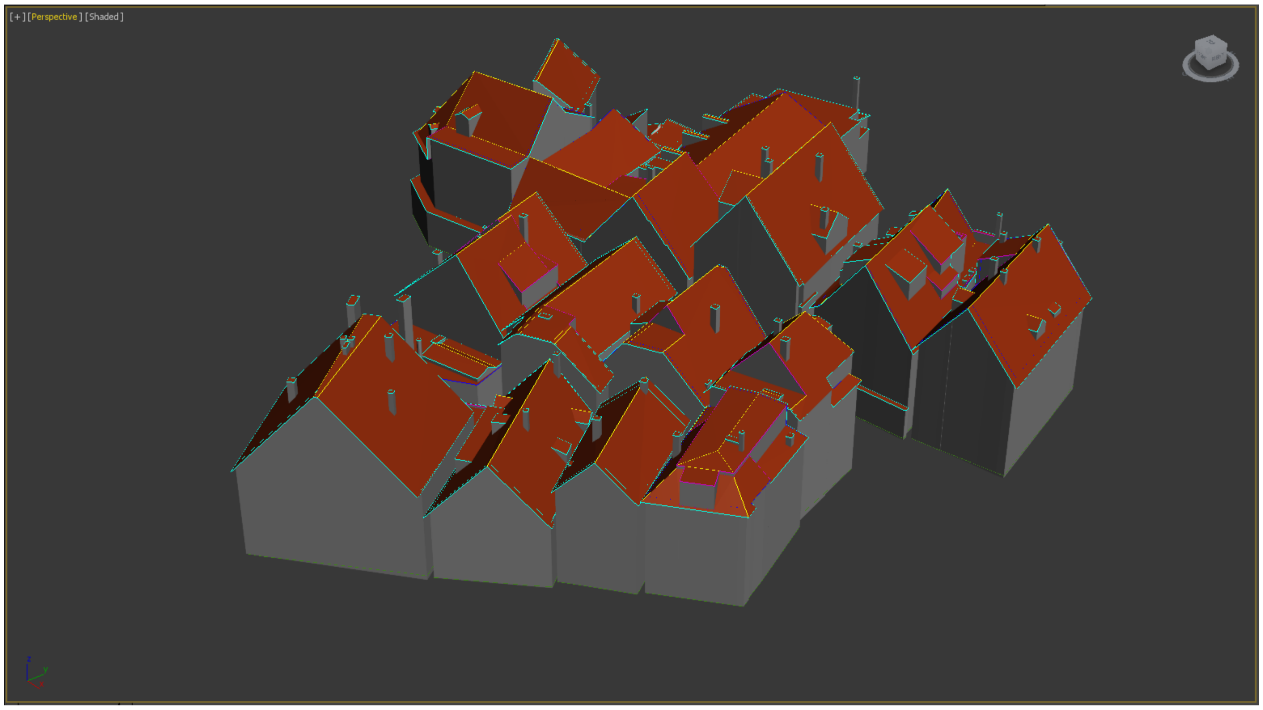


Figure 5.49: The modeled buildings in case study, visualization in Autodesk 3D Max

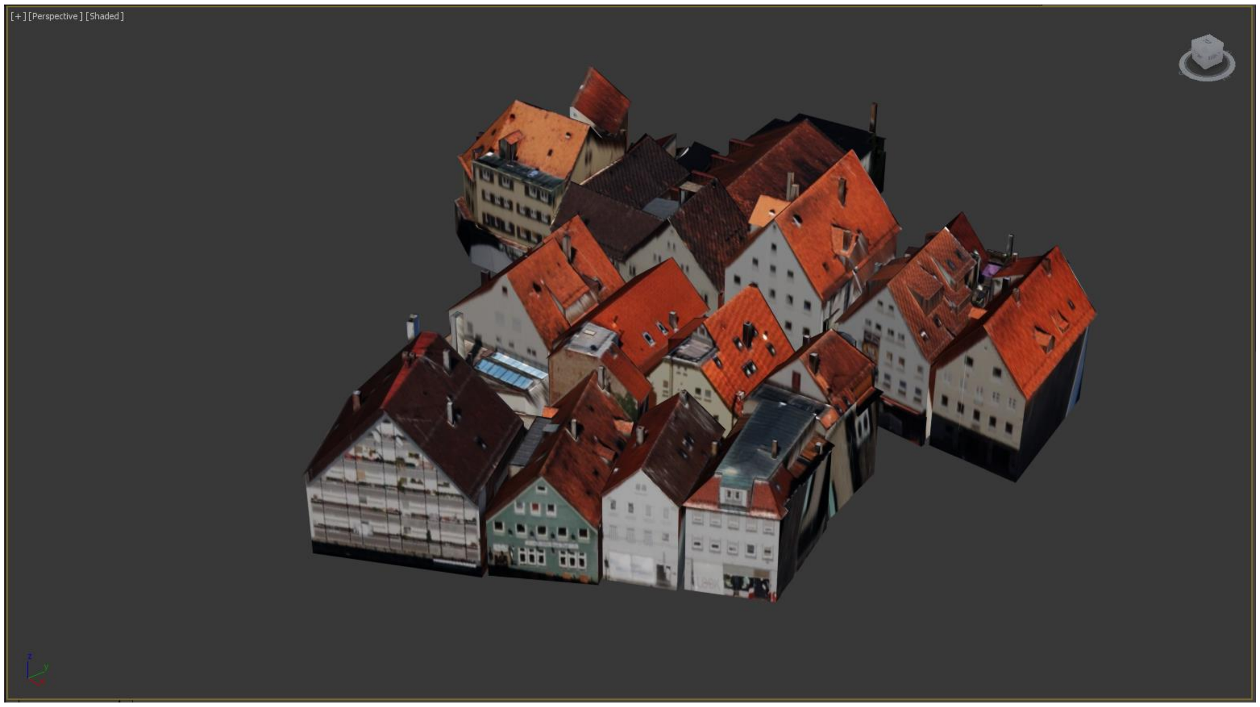


Figure 5.50: The modeled and texturized buildings in case study, visualization in Autodesk 3D Max

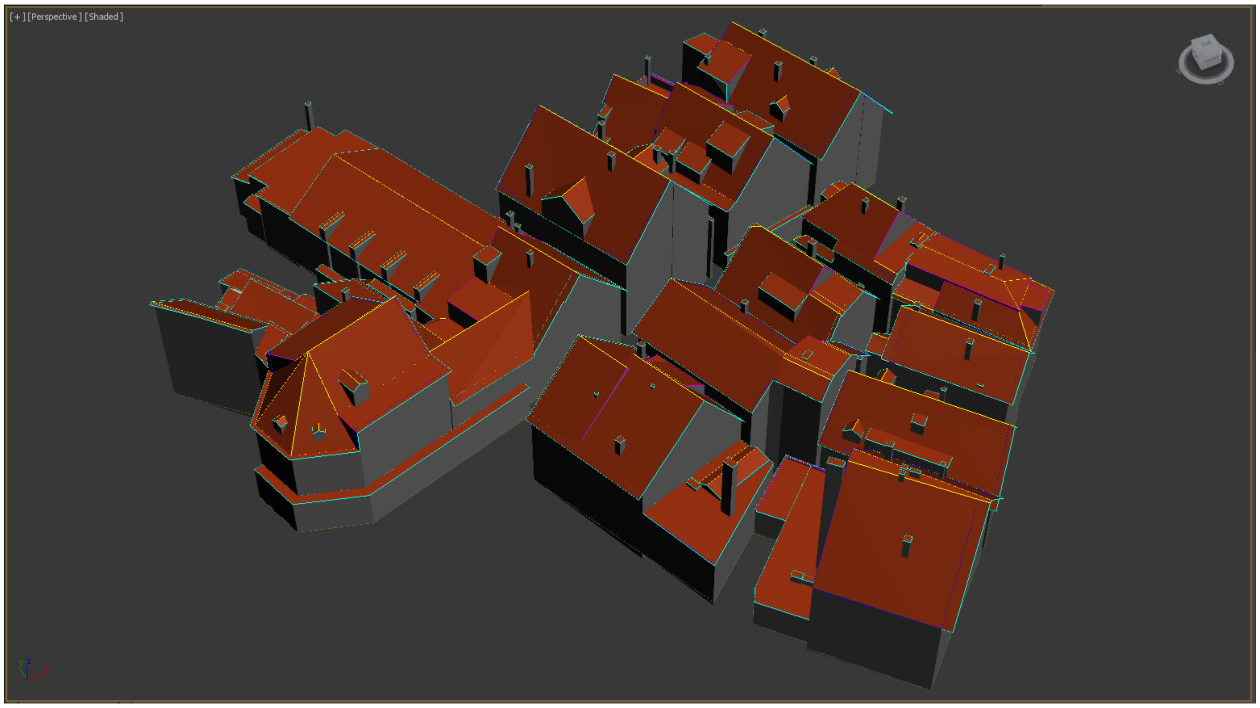


Figure 5.51: The modeled buildings in case study, visualization in Autodesk 3D Max

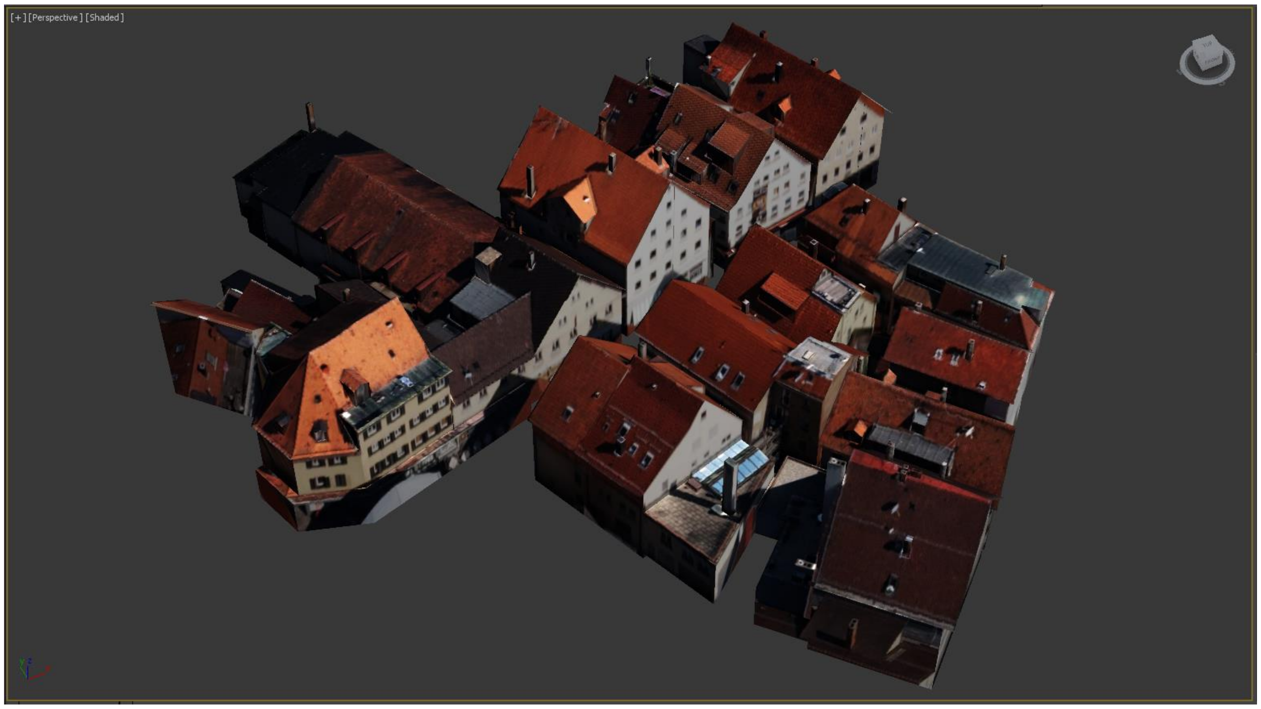


Figure 5.52: The modeled and texturized buildings in case study, visualization in Autodesk 3D Max

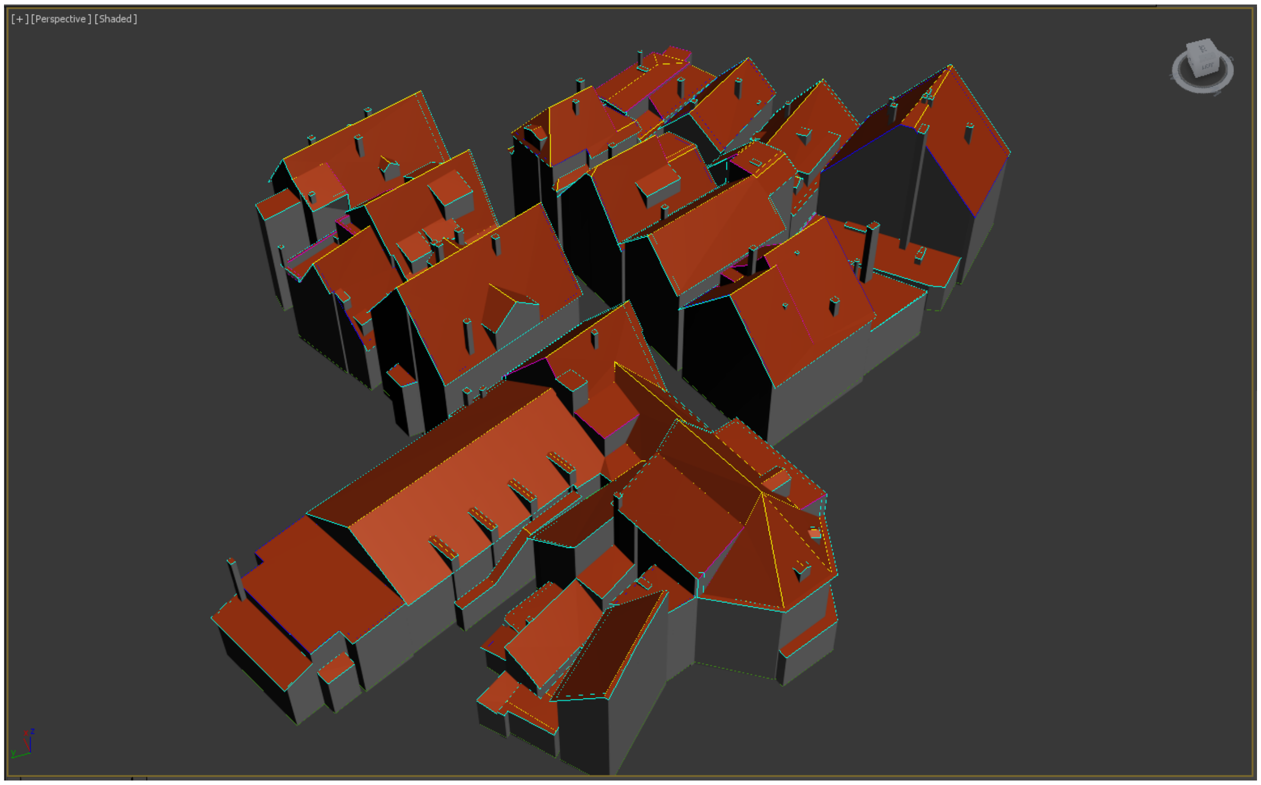


Figure 5.53: The modeled buildings in case study, visualization in Autodesk 3D Max

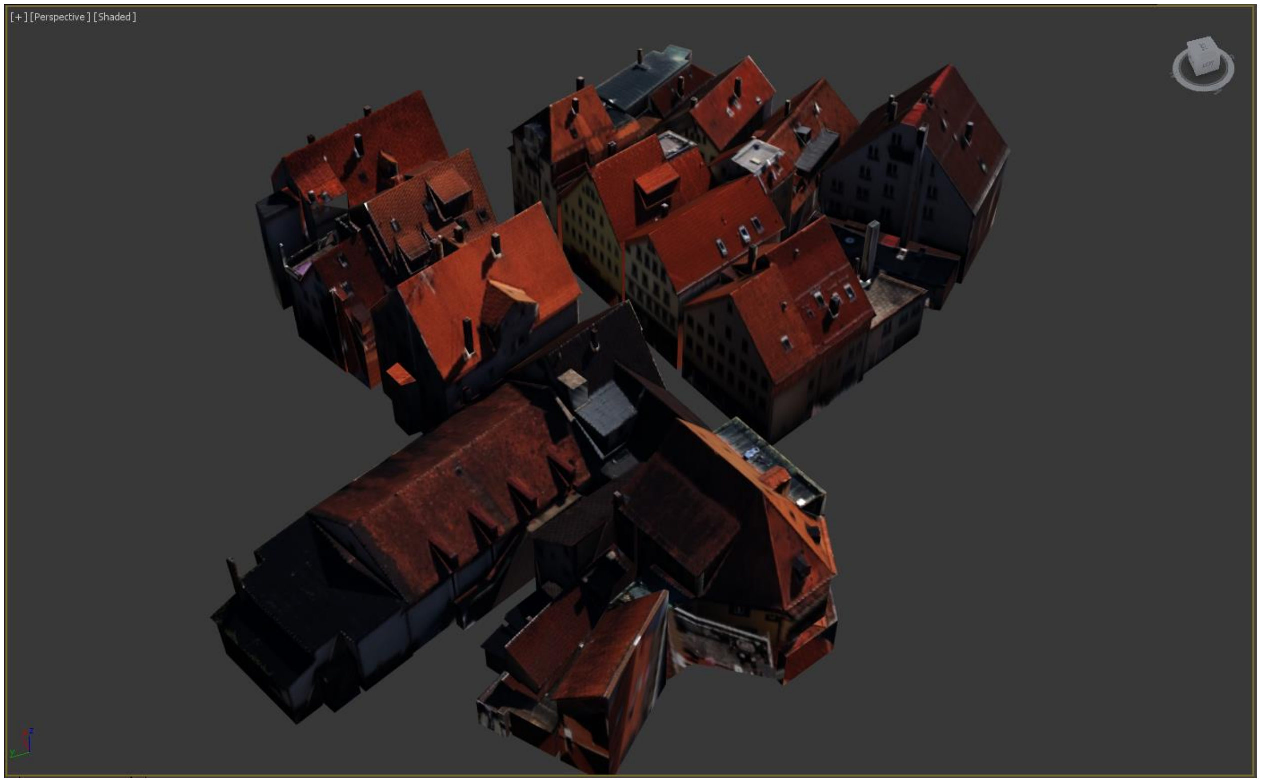


Figure 5.54: The modeled and texturized buildings in case study, visualization in Autodesk 3D Max

5.5 3D City Modeling with SketchUp

The general idea for the generation of a detail 3D model with SketchUp was combining the three datasets which are roof lines, footprints, and DTM. The roof surfaces are created by using roof lines, and the intersection of the roof surface with DTM gives the real height of the buildings. Additionally, the footprints information is utilized to extract the roof overhang.

The followed process with SkecthUp is shown in figure 5.55 and will be explained in detail in this chapter.

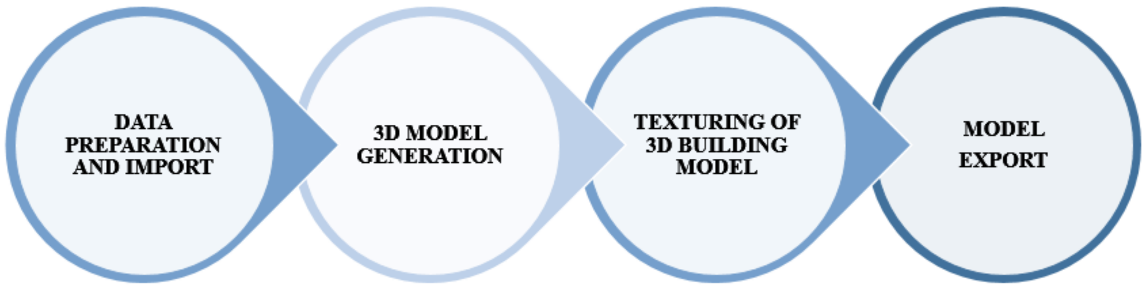


Figure 5.55: Implementation process with SketchUp

5.5.1 Data Preparation and Import

The first task of data preparation step is combining the datasets from different source and formats. The first challenge was converting the different formats into a format which SketchUp Pro enables to import. The roof line dataset is provided in *dwg*, which can be imported without a problem into SketchUp, yet DTM is in *xml* and footprints in *shp* format, which SketchUp is not able to import. Hence both datasets were converted into *dxf* format by the help of Autodesk 3ds Max software[10] (Figure 5.56, 5.57).The other approach would be using CityEdior, which is a commercial SketchUp plugin developed by3D Information Systems (http://www.3dis.de/loesungen/3d-stadtmodelle/cityeditor/, last access November 2016). CityEdior enables importing various formats as well as editing and exporting. As the plugin is not freely available, the formats have to be changed with Autodesk 3ds Max.

[10] Autodesk 3ds Max software offers free education license for 3 years.

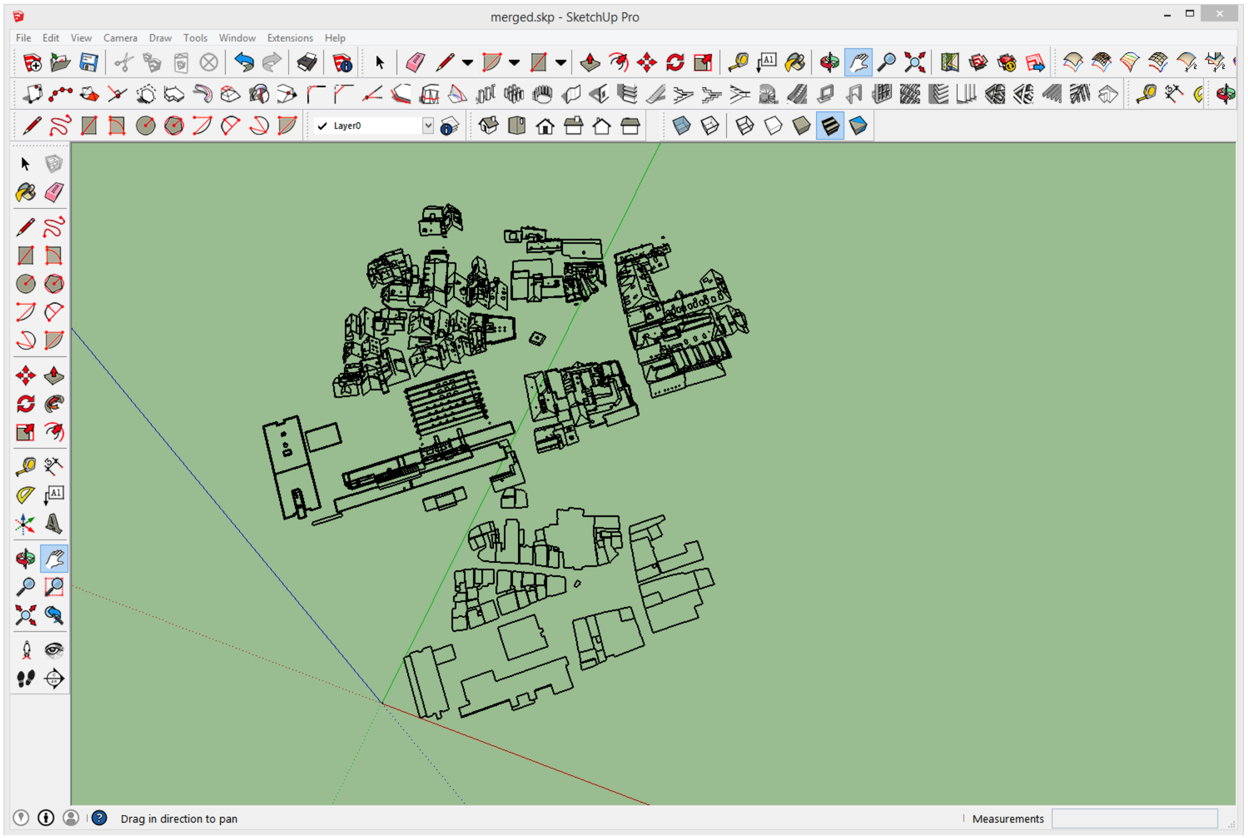

merged.skp - SketchUp Pro
File Edit View Camera Draw Tools Window Extensions Help
Layer0
Drag in direction to pan
Measurements

Figure 5.56: Roof lines and Footprint in SkecthUp after format conversion

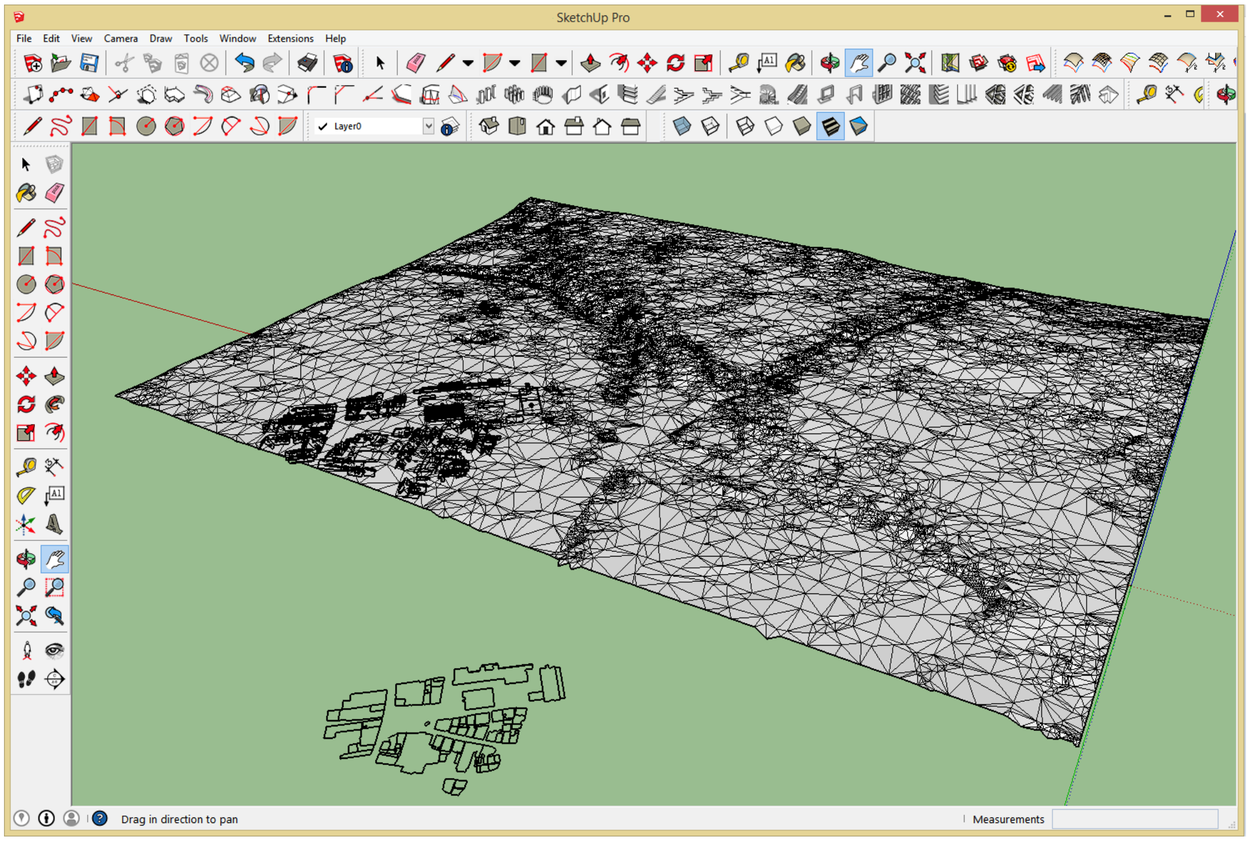
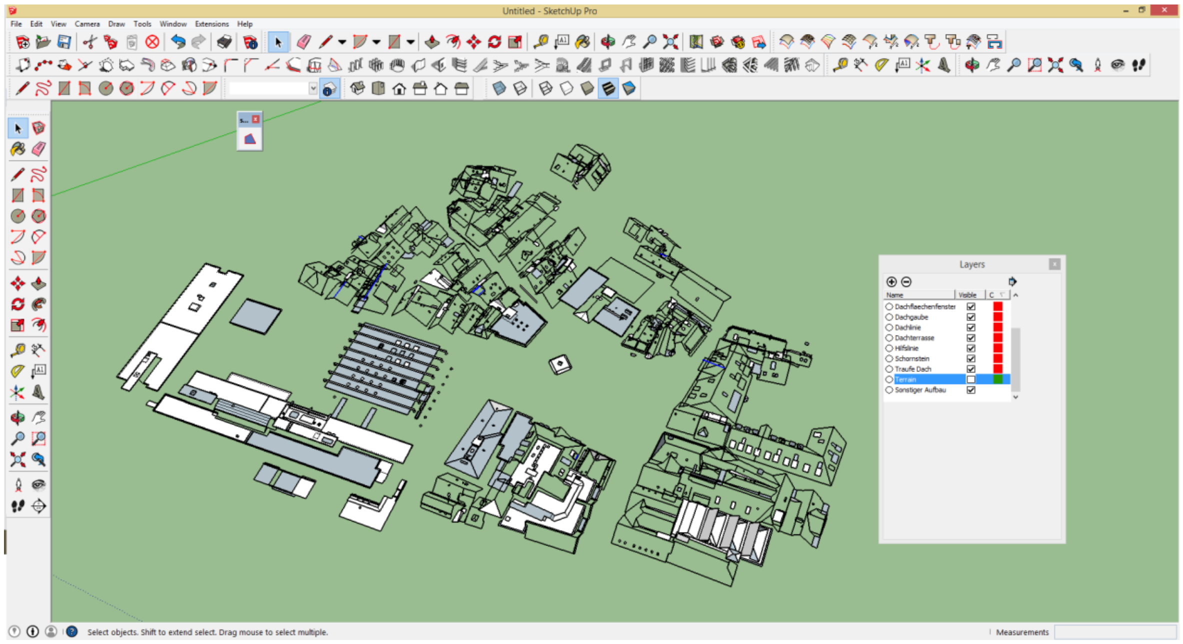


Figure 5.57: Roof lines, DTM and Footprint in SkecthUp after format conversion

5.5.2 Model Generation

The model generation begins by generating roof surface from roof lines. For this purpose, the basic tools which are explained in the third chapter can be used. However, as the intention is to make generation less time-consuming and more automatic, different plugins have been tried out. Firstly, s4u Make Face plugin, which generates automatically faces from coplanar line segments, was utilized. As it can be seen in Figure 5.58, most of the faces have been not created by the plugin. The main reason for that is non-coplanar lines.

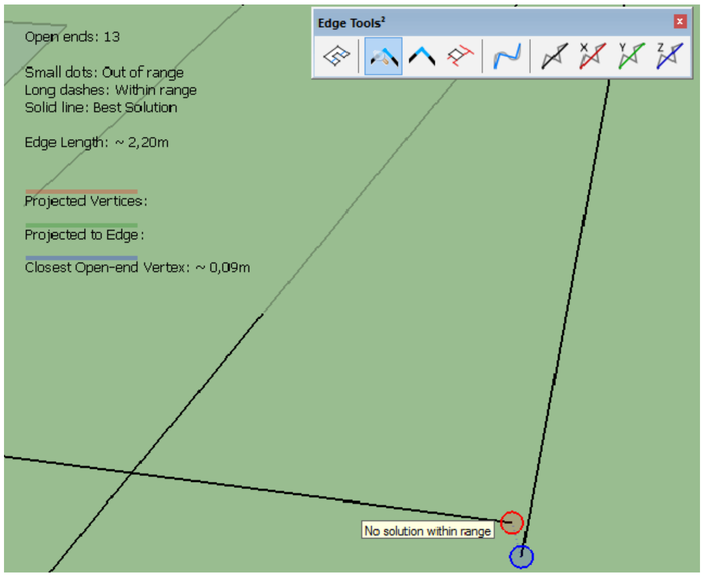


Name	Visible	C
Dachflaechenfenster	✓	🟥
Dachgaube	✓	🟥
Dachlinie	✓	🟥
Dachterrasse	✓	🟥
Hilfslinie	✓	🟥
Schornstein	✓	🟥
Traufe Dach	✓	🟥
Terrain		🟩
Sonstiger Aufbau	✓	

Select objects. Shift to extend select. Drag mouse to select multiple.

Figure 5.58: The result of the Make Face Plugin

Another reason for the failure of the plugin is open ends of roof lines (Figure 5.59). By mean of Edge Tools, not connected edges have been found out. Those lines have been manually corrected, and the faces have been created by s4u Make Face plugin.



Open ends: 13

Small dots: Out of range
Long dashes: Within range
Solid line: Best Solution

Edge Length: ~ 2,20m

Projected Vertices:

Projected to Edge:

Closest Open-end Vertex: ~ 0,09m

No solution within range

Figure 5.59: Example of open ends

The faces of non-coplanar lines have been generated by Sandbox Tools plugin (Figure 5.610). The plugin transforms roof lines into TINs. The tool requires selecting the edges of the object to be modeled. Otherwise, it triangulates through all objects. Therefore, the lines of each roof must be chosen separately in order to create independent faces.

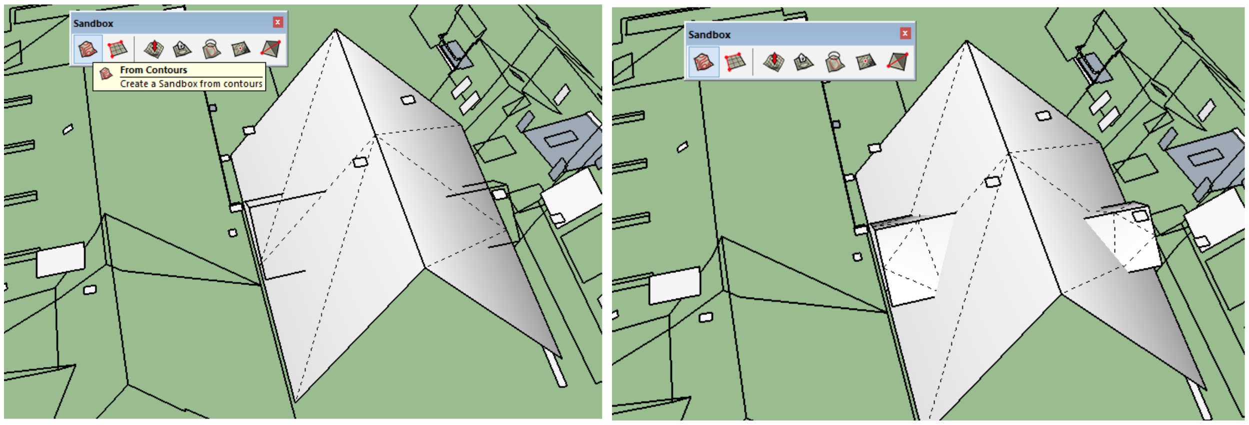


Figure 5.60: Creating the roof and dormer face with Sandbox Tools

The results after face generation can be seen in Figure 5.61.

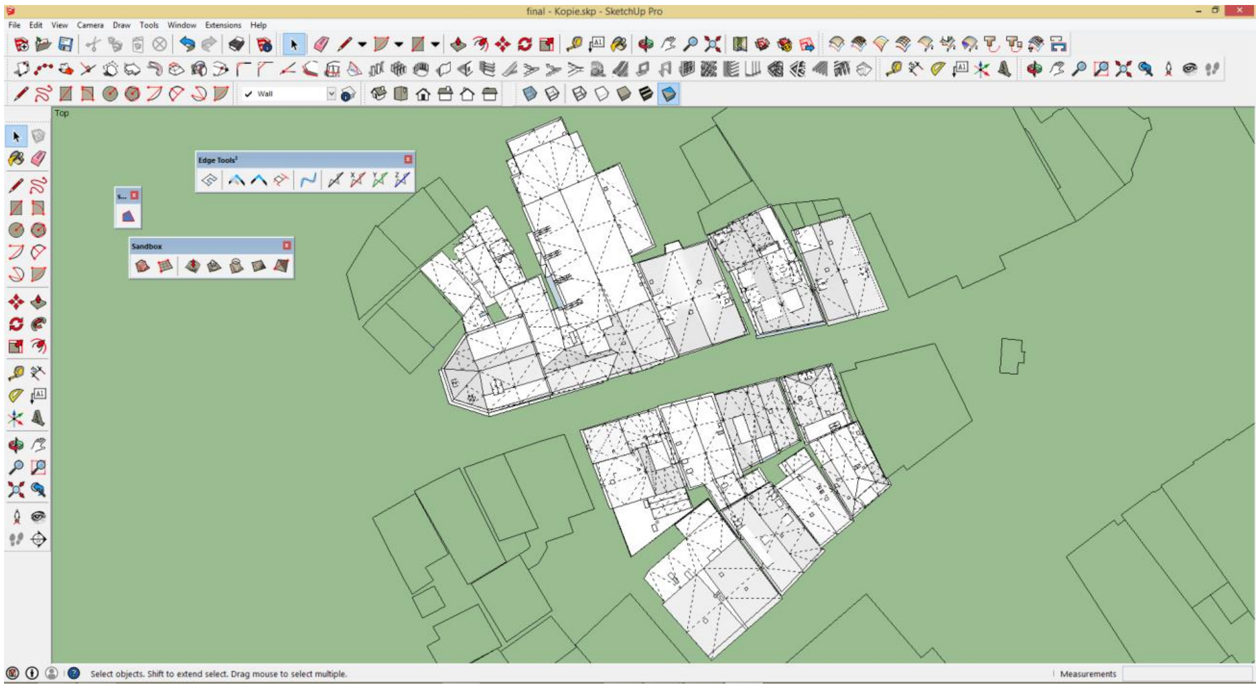

final - Kopie.skp - SketchUp Pro
File Edit View Camera Draw Tools Window Extensions Help
Wall
Top
Edge Tools²
Sandbox
Measurements
Select objects. Shift to extend select. Drag mouse to select multiple.

Figure 5.61: Roof surfaces

For the creation of the vertical walls, Extrusion Tools have been used (Figure 5.62). Building walls have been modeled by creating vertical walls by the plugin from footprint to the associated roof surface. With this method, both vertical walls and roof overhang have been modeled. Extrusion Tools also requires manual selection of each individual object, from where the vertical wall is to be created. As it can be seen in Figure 5.63, the edges of a footprint have been selected in order to create vertical walls. This process has been repeated for each and every building footprint that is to be modeled.

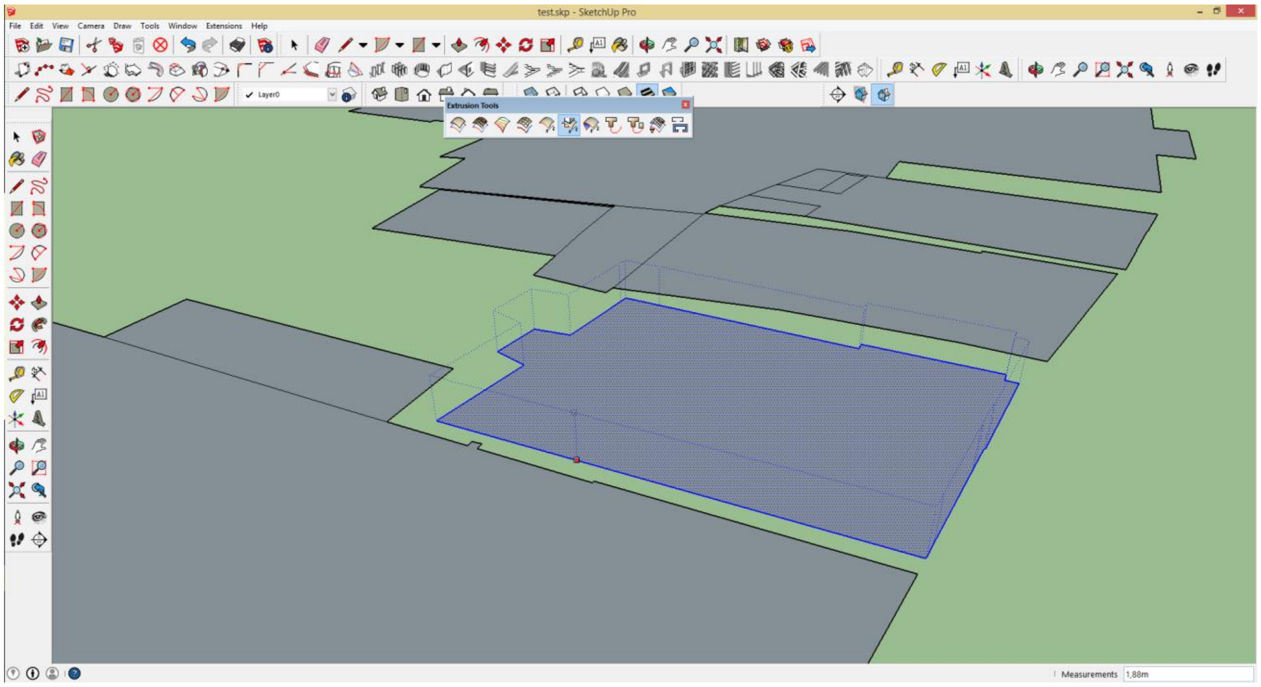


Figure 5.62: Creating Vertical Walls from building footprint with Extrusion Tools

After vertical walls are modeled, the buildings are intersected with DTM in order to obtain the real height of the buildings.

As a final point, the vertical walls of roof details such as dormer, windows, and chimney have been generated by Extrusion Tools. The vertical walls of roof details have been created from the roof detail to associated roof surface. The Extrusion Tools cut the vertical walls where the vertical meet with any surface. The results can be seen in Figure 5.63 and 5.64.

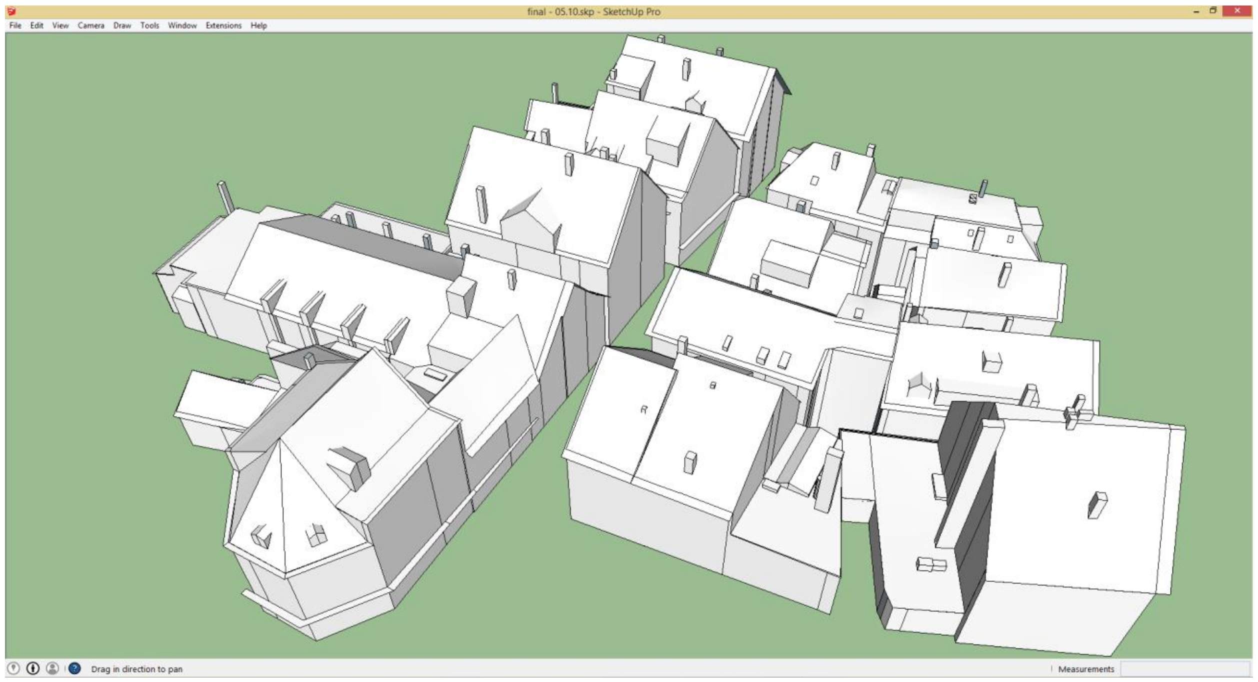

final - 05.10.skp - SketchUp Pro
File Edit View Camera Draw Tools Window Extensions Help
Drag in direction to pan
Measurements

Figure 5.63: The model after roof overhang creation

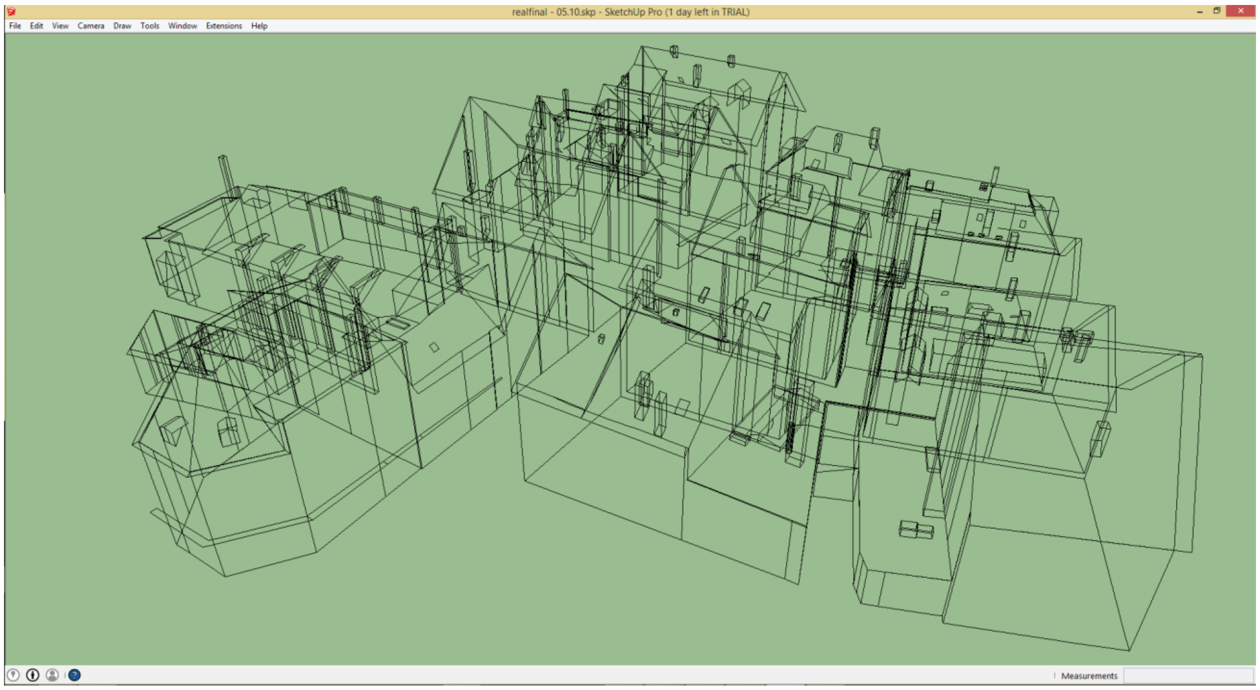

realfinal - 05.10.skp - SketchUp Pro (1 day left in TRIAL)
File Edit View Camera Draw Tools Window Extensions Help
Measurements

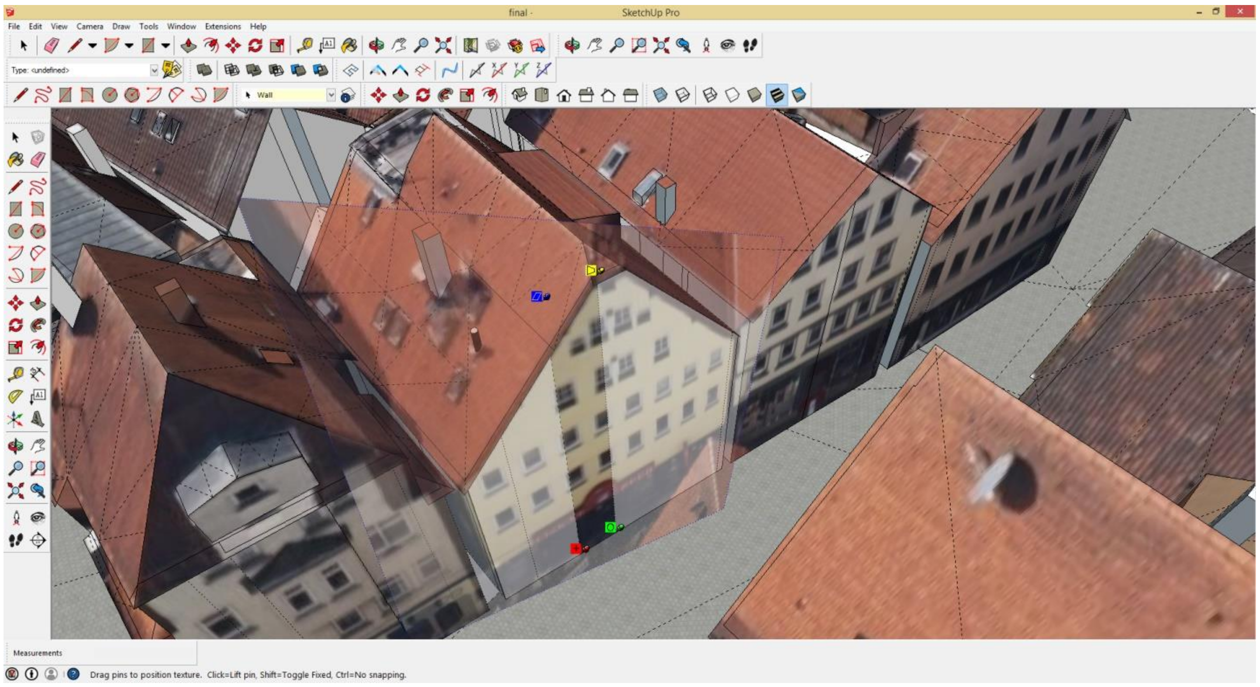
Figure 5.64: Wireframe of the model

5.5.3 Texture Generation

SketchUp enables texturing the existing model. The first step for that, the images will be used for the texturing must be added into SketchUp Material Libraries. The images with extension of .jpg, .png, .psd, .tif, .tga and .bmp can be used for the texturing. In the Material Libraries, each picture is considered as a material palette.

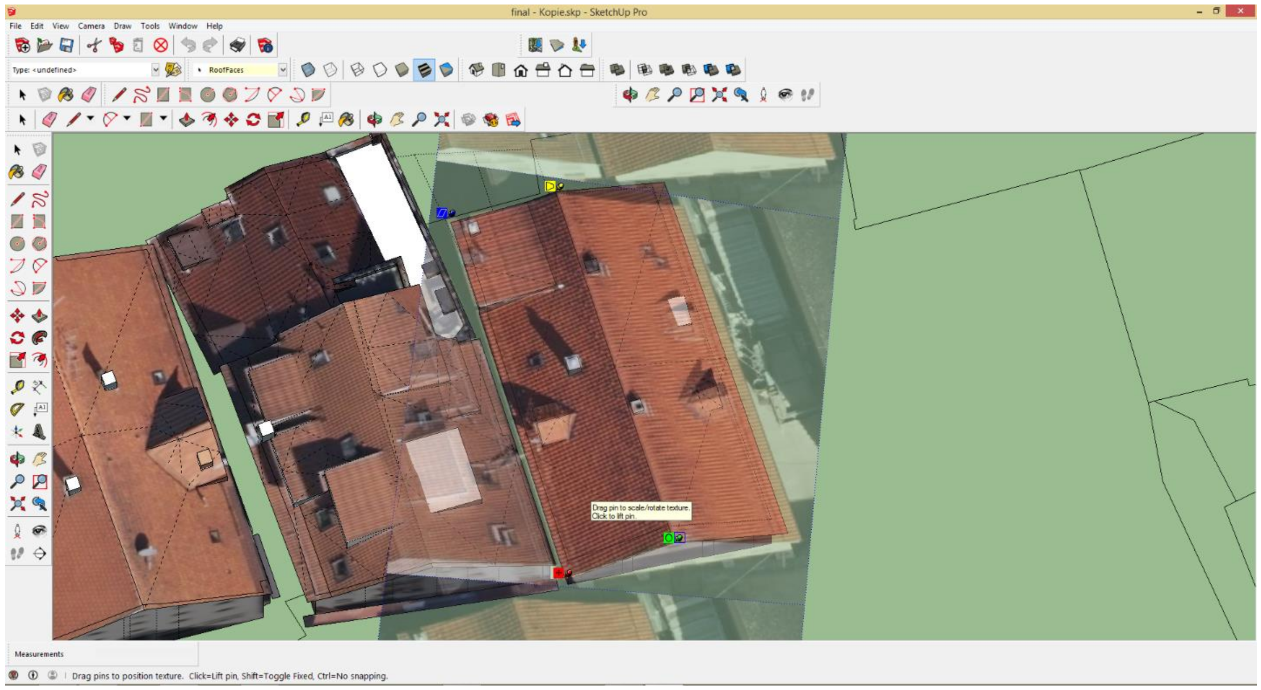
With Paint Bucket tool, the image can directly be projected into chosen face. The position of the texture image must be determined manually. Therefore four pins help the user to match the picture with the appropriate angle of incidence and perspective (Figure 5.65 and 5.66). If a face is divided into more than one neighbor faces, the positioning of one image is sufficient as the texture of the other neighbor face can be projected from the first face with Sample Paint tool.

Texturing models with SketchUp is a completely manual process. There is no automatic matching option. Therefore user has to decide which picture, the model needs to be textured. In the case study, the images have been cropped building by building as the aerial images were rather large, which makes texturing the images with pins harder and slower.



Drag pins to position texture. Click=Lift pin, Shift=Toggle Fixed, Ctrl=No snapping.

Figure 5.65: Texturing the building facade with SkecthUp


final - Kopie.skp - SketchUp Pro
File Edit View Camera Draw Tools Window Help
Type: <undefined>
RoofFaces
Measurements

Drag pin to scale/rotate texture.
Click to lift pin.

Drag pins to position texture. Click=Lift pin, Shift=Toggle Fixed, Ctrl=No snapping.

Figure 5.66: Texturing the roof with SkecthUp

5.5.4 Model Export

SketchUp allows export of the model to 3DS, AutoCAD DWF, AutoCAD DXF, COLLADA, FBX, IFC, Google Earth File, OBJ, VRML and XSI file. The model can also be exported as 2D Graphic as JPEG Image, PDF File or Tagged Image File. In addition, the Plugin CityEditor, which has been mentioned before, allows an export of SketchUp models to CityGML and 3D-PDF documents.

5.5.5 Results

The following sequence of figures (Figure 5.67 to 5.74) shows visualizations of the modelling results from SkecthUp.

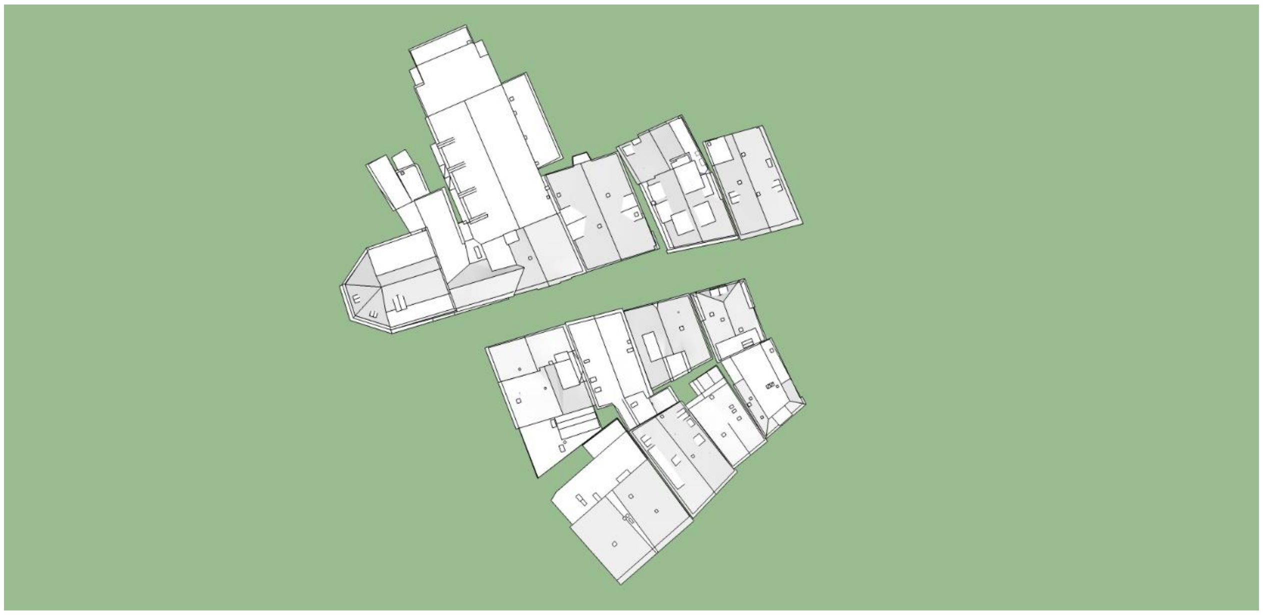


Figure 5.67: The modeled buildings in case study, visualization in SkecthUp

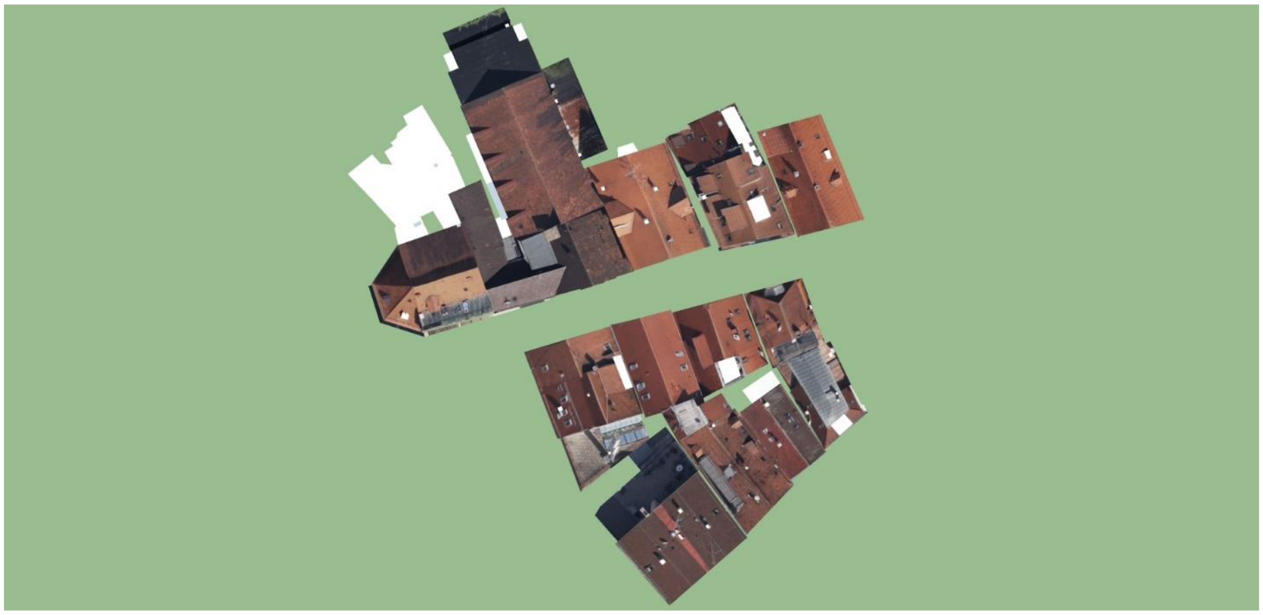


Figure 5.68: The modeled and texturized buildings in case study, visualization in SkecthUp

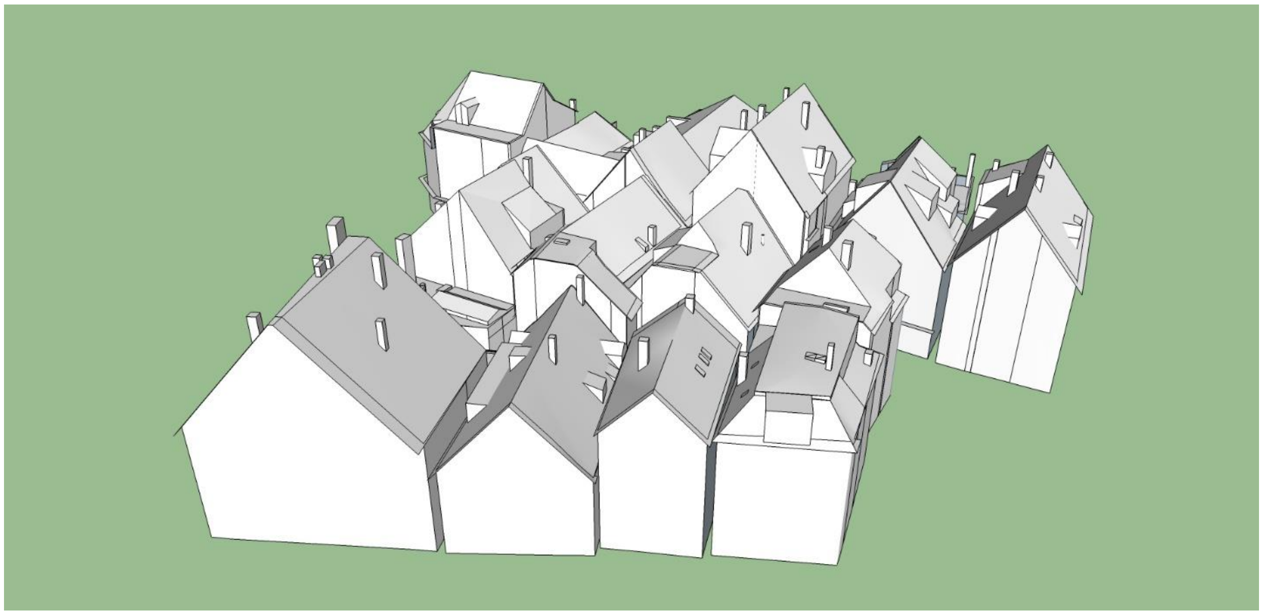


Figure 5.69: The modeled buildings in case study, visualization in SkecthUp

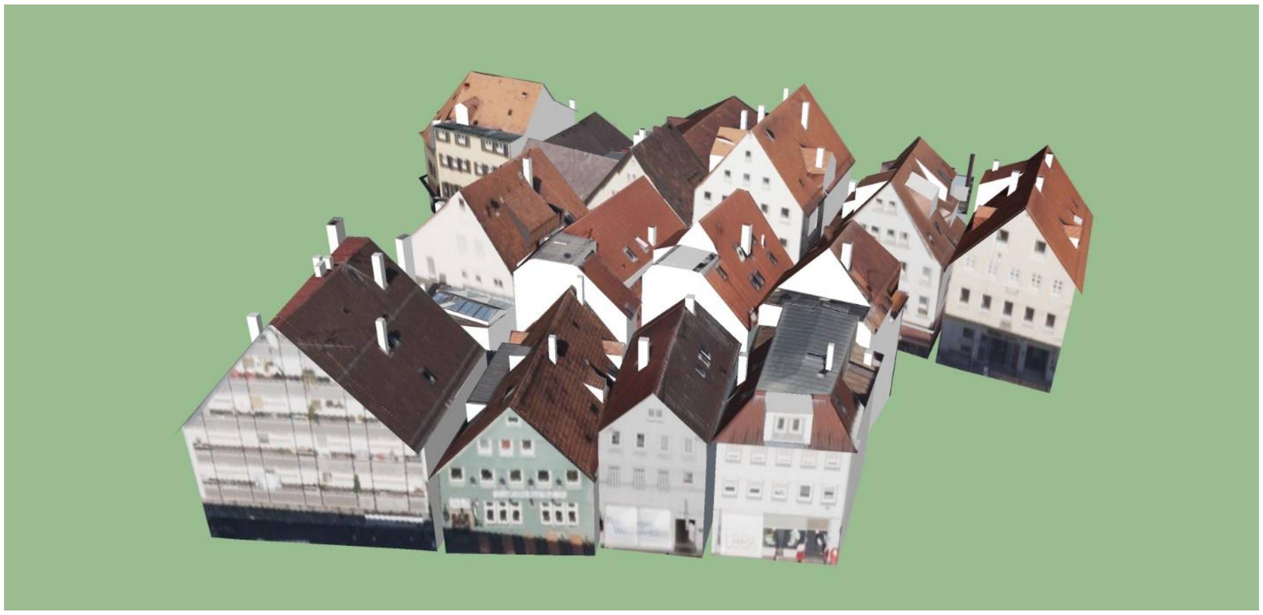


Figure 5.70: The modeled and texturized buildings in case study, visualization in SkecthUp

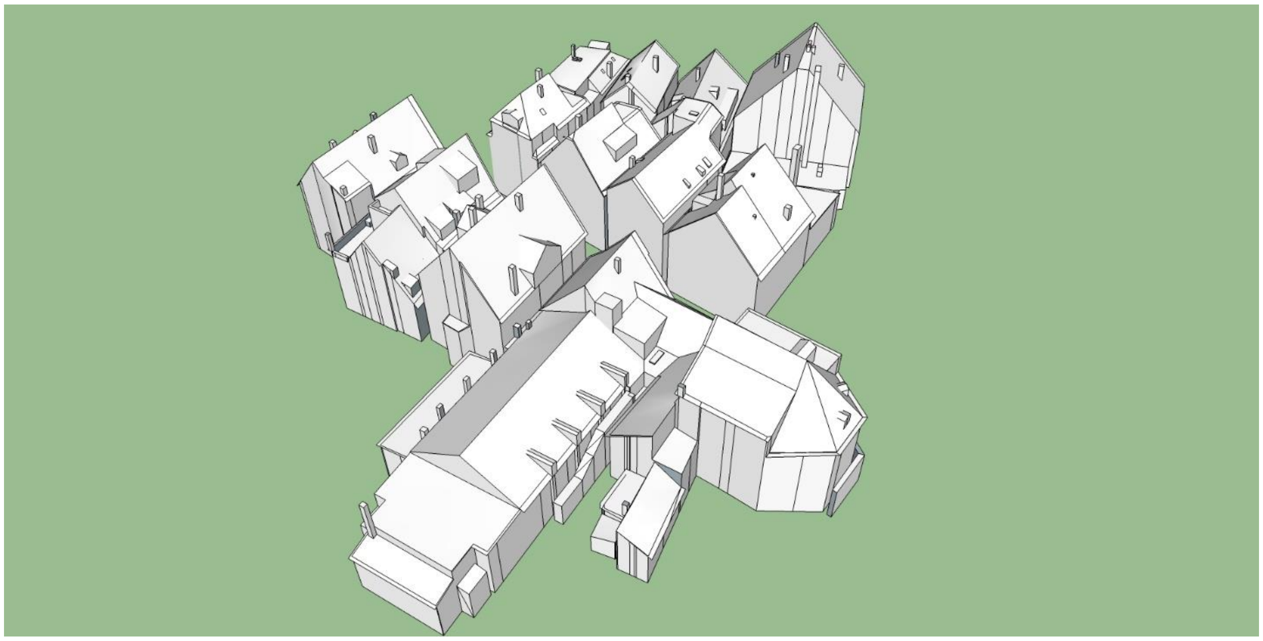


Figure 5.71: The modeled and texturized buildings in case study, visualization in SkecthUp

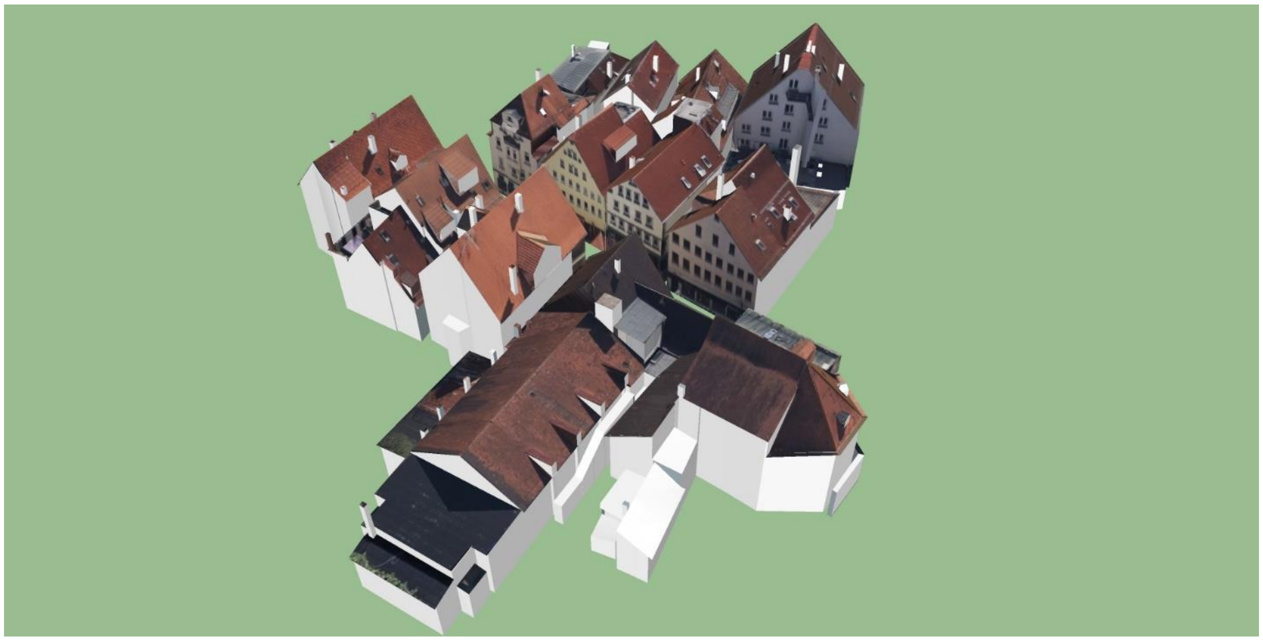
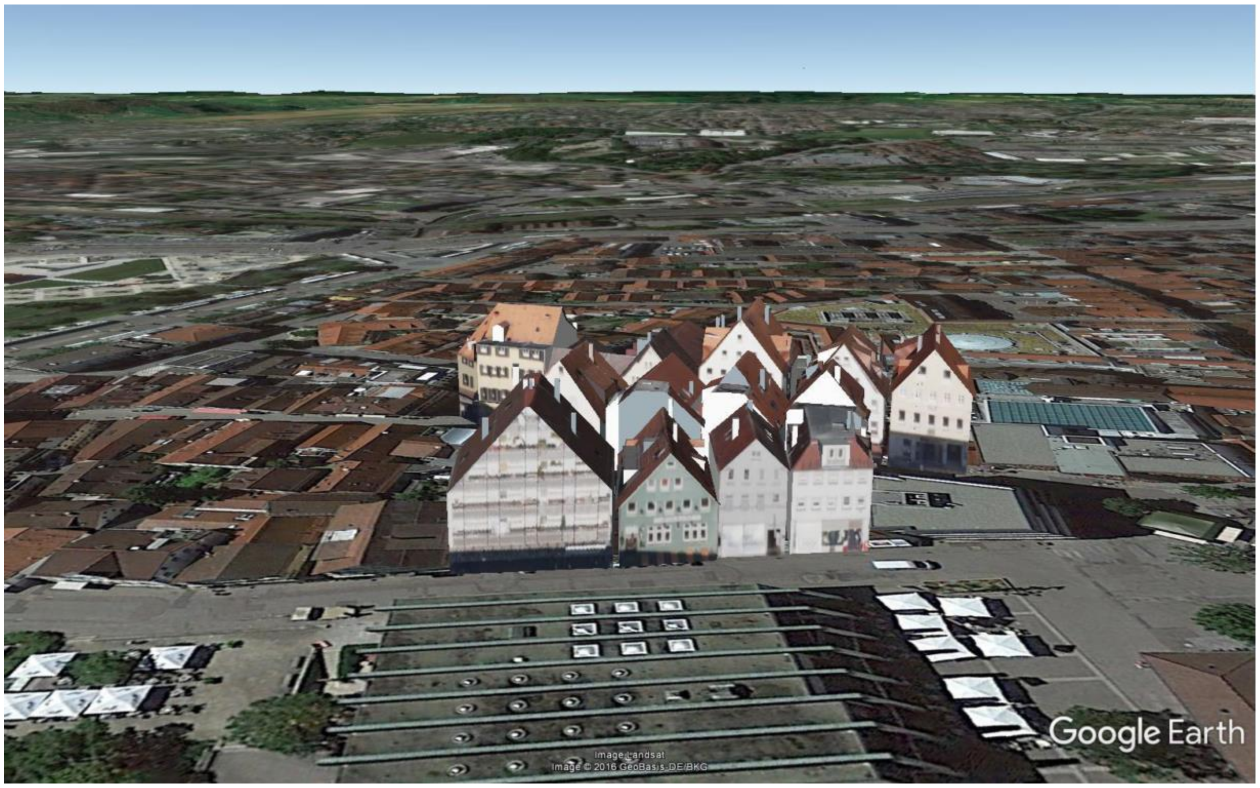


Figure 5.72: The modeled and texturized buildings in case study, visualization in SkecthUp



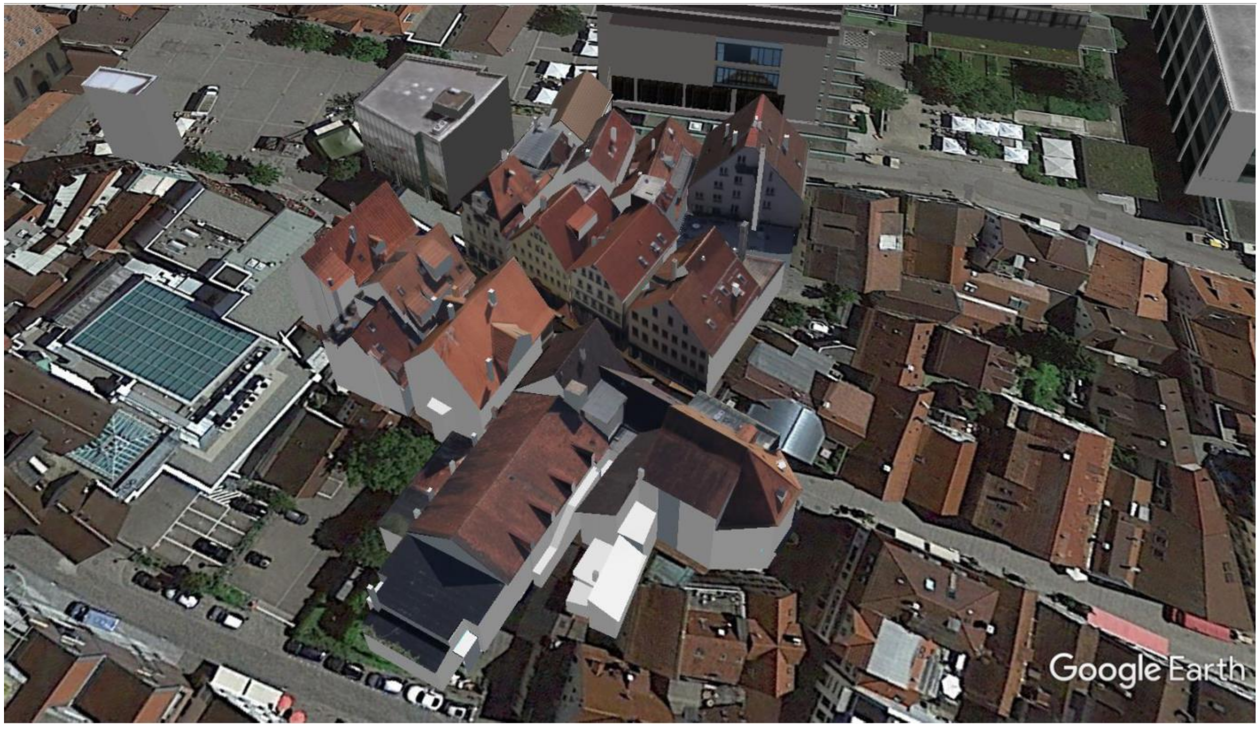


Figure 5.73: The modeled and texturized buildings in case study, visualization in Google Earth

Figure 5.74: The modeled and texturized buildings in case study, visualization in SkecthUp

CHAPTER 6

The comparative assessment (using some 20 buildings) of the selected software is based on mainly the results of the exemplary 3D city model, as well as its process of generating and texturing. Additionally, other criteria have been taken into consideration such as import and export options, easiness, required effort and ability to update. By doing so, the requirement and the process to generate an entire city model with selected software packages could be estimated. In this section, the results of the case study are discussed and compared to each other.

6 RESULTS AND DISCUSSION

The needs for each 3D city modeling project vary depending on the aim of the model, size, and composition of the city, the budget of the project as well as at what detail the model will be. Same sort of 3D city model may not be feasible for different applications. For instance, an LOD2 city model for the purpose of noise protection is not feasible for the use of a real estate property project, which requires LOD. Because of this reason, this criterion has also been taken into account during the assessment of the suitability of the software for a 3D city modeling project. Figures of the models by each software can be seen in Chapter 5 as well as in Figure 6.1, 6.2 and 6.3.

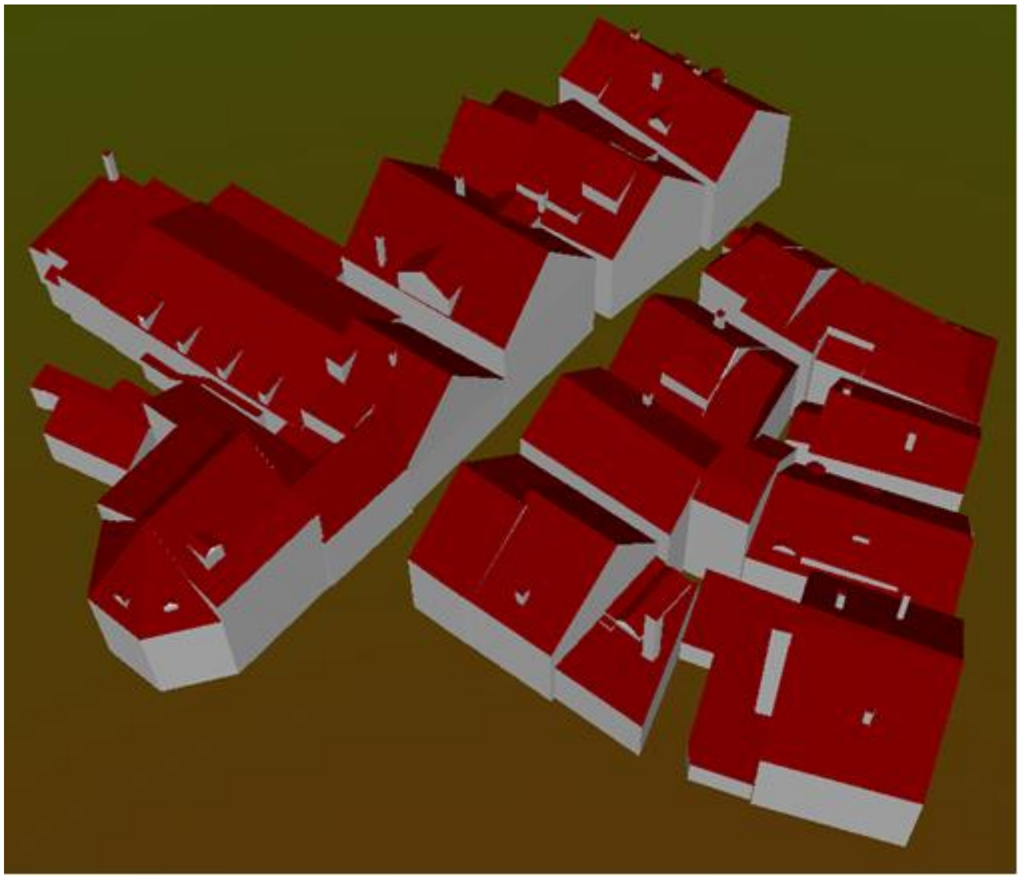


Figure 6.1: Model 1 is generated by 3D Editor

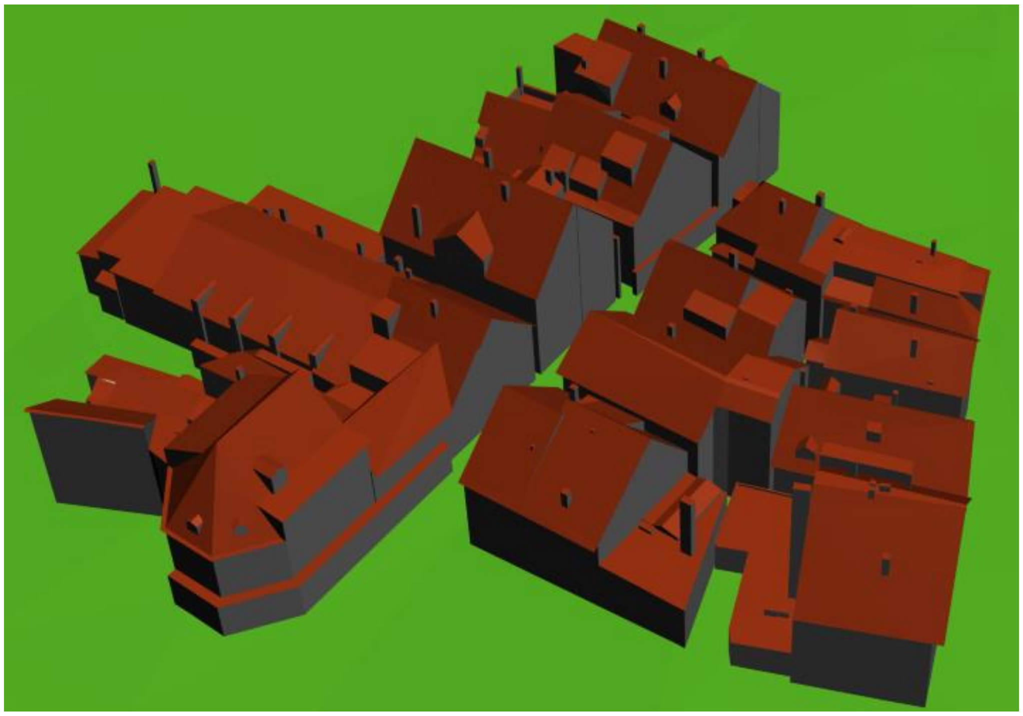


Figure 6.2: Model 2 is generated by CityGRID Modeler

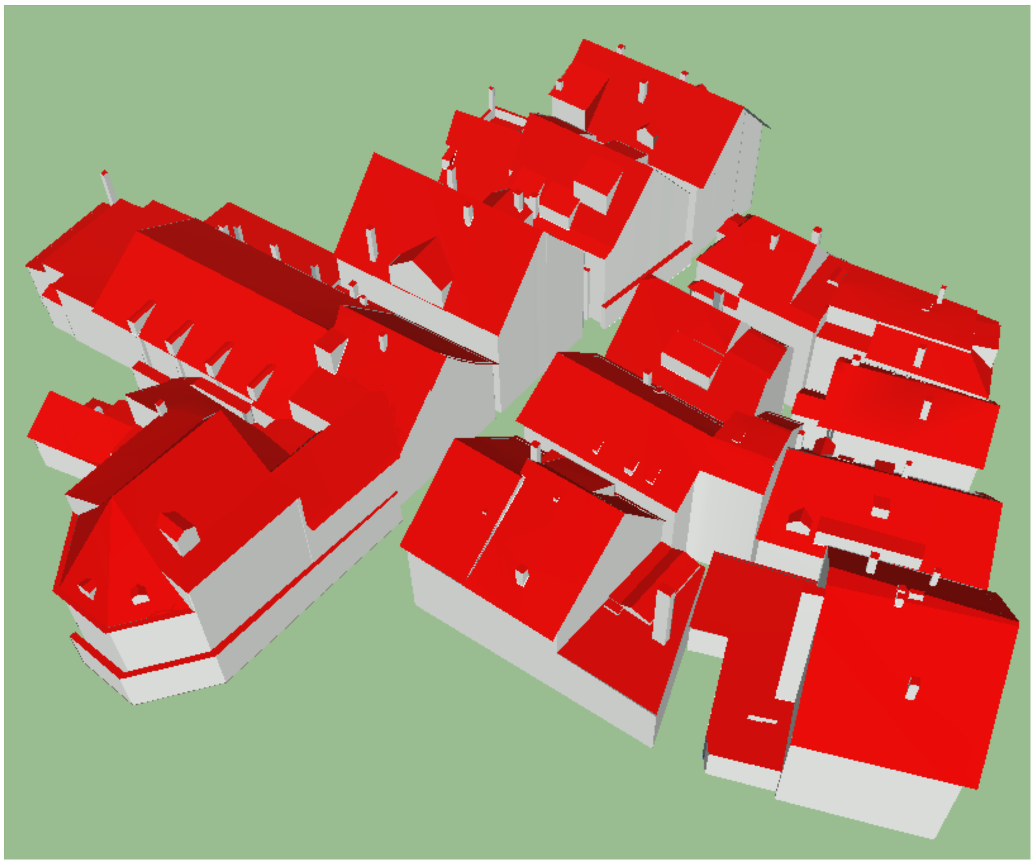


Figure 6.3: Model 3 is generated by SketchUp

6.3 Comparison of Basic Working Principle

Each software package assessed in this thesis allows the user to interactive edit and model 3D building with different methods.

3D Editor works mainly on the principle of photogrammetric interpretation. Therefore, the program requires aerial images for the modeling. Without data, the modeling is not possible. 3D Editor enables modeling the buildings based on predefined building primitives. These predefined building primitives are the main limitation of the. In cases with many irregular buildings, as they can be found in historic cities or in small villages, the adaptability very soon reaches its definite limit

To model a building with **CityGRID Modeler**, input data requires either as XML or available in the database. The program has a line-oriented approach. Hence, input data must have a line structure which is mostly extracted from photogrammetric restitution.

SketchUp works mainly on the principles of sketch-based modeling. There is no special tool for the modeling of a 3D building but through the combination of a number of tools. In contrast to 3D Editor and City Modeler, SketchUp does not require data for modeling. In another word, SketchUp can create both existing building and proposed buildings.

CityGRID Modeler and 3D Editor are more specific for a modeling of ensembles of 3D building whereas SketchUp is a 3D modeling software for a wide range of applications with a focus more on individual objects rather than huge arrangements.

The interface of 3D Editor and CityGRID Modeler allows the user to see the model both in 2D and 3D. However, the model in SketchUp cannot be simultaneously seen in 2D. The user interface enables only 3D view.

6.4 Comparison of Import & Export Capacity

3D Editor´s initial file format is *op3d/tdc*, and it also allows importing file in *dxf* and *shp* file. However, one can work with *dxf* only if it fulfills the requirement of the program (See Chapter 5.1). 3D Editor cannot export the model as *CityGML*. However, the Export software ofby Hexagon enables conversion of *op3d/tdc* into *CityGML*.

As already mentioned, **CityGRID Model** requires an input file either in XML format or an available data set in the database. If the input data is a CAD format such as dwg, it must be converted to xml format through CityGRID Converter. Working directly with *CityGML* format is not possible. A *CityGML* file can be converted through FME into an xml file, and then the program can work with it.

In contrast to other assed software, **SketchUp** provides larger import and export capacity (Table 6.1). Through various purchasable and costless plug-ins, import and export capacity of SkecthUp increases. For instance, City Editor[11] plug-in, which is a charged plug-in, allows importing, editing and exporting CityGML file.

In summary, apart from 3D Editor, both assessed software packages requires either a plug-in or additional software to work with CityGML. The only difference between them is the cost factor. SketchUp plug-ins are generally more convenient compared to that for FME software.

	Import	**Export**
3D Editor	*Vector file*: dxf, shp	dxf, vrml *via Export software:* CityGML,Collada, EsriGDB, kml, kmz, myVR, obj, shape
	DEM file: asc, dtm	
	Image file: vit, tiff, raw, tld, bmp	
CityGRID	*Vector file*: xml, from database (Oracle und MSSQL)	vrml, dxf, xml, kml,CityGML
	DEM file: xml, from database (Oracle und MSSQL)	
	Image file: bmp, gif, jpg, png, tga, tif	
SketchUp	*Vector data*: *:* dwg, dxf, 3ds, dae, kmz, fbx	dae, kmz, 3ds, dwg, dxf, fbx, obj, wrl, xsi
	DEM file: dem, ddf	
	Image file: jpg, png, psd, tif, tga, bmp, eps, pdf	

Table 6.1: Comparison of Import and Export Capacity

6.5 Comparison of Modeling Process

By the help of each assessed software packages in this thesis, a 3D building model can be generated, an existing 3D building can be edited, and the model can be enriched by adding roof details such as dormers, chimneys, and other superstructures. The main difference between software is the process of the modeling.

As a digital photogrammetric workstation, 3D Editor can generate 3D building models with predefined building primitives. This process requires manual afford for each and every building to be modeled. Depending on the complexity of the building and roof type, the time required for modeling of one building changes from 2 up to 10 minutes. It can be longer if one wants to add more details on the roof. In the case study, an LOD2 city model has been automatically created through **CityModeller** in order to avoid manual generation of 3D buildings with **3D Editor**. By using 3D Editor, buildings have been edited and enriched with roof details. 3D Editor has no special tool for the generation of the roof detail. Therefore, roof details are also generated manually by using buildings primitives. There exist no additional building primitives that are special for roof or facade details.

[11] CityEditor (see https://extensions.sketchup.com/de/content/cityeditor-2; last access December 2016)

CityGRID Modeler generates faces of 3D buildings by the help of triangulation algorithm. Among the assessed software packages, only the Modeler provides special tools for the creation of roof details (i.e. dormer, chimney, roof window). It does not require any manual effort and enables to generate so many roof details at once. For instance, the roof details of 18 buildings have been generated all together by using "Detach new complexes" tool. CityGRID Modeler enables to detach tunnels and building passage to the buildings.

Each building block in **SkecthUp** is composed of edges and faces. A face is a closed coplanar spatial polygon of any number of lines (minimum is three). However, in the case study, all lines that belong to one roof, except the triangular ones, were not always coplanar. In non-coplanar cases, a face needs to be composed of multiple faces (Figure 6.4).

SketchUp and 3D Editor model 3D buildings individually. Contrary to SketchUp and 3D Editor, simultaneous modeling of several buildings is possible only with CityGRID Modeler.

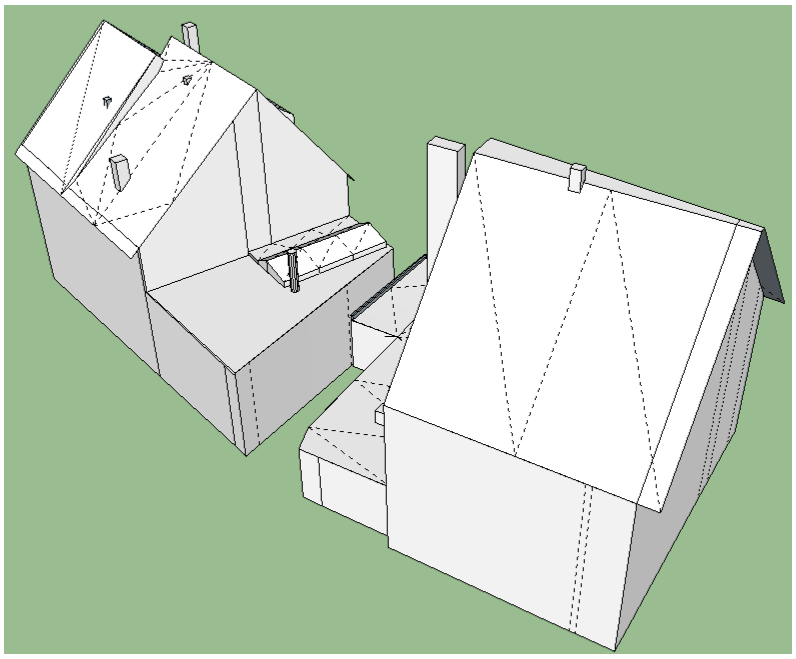


Figure 6.4: The multiple faces of roofs in SkecthUp

Each assessed software packages in this thesis can generate a 3D building seamlessly with a standard roof type such as gable roof, hip roof or gambrel roof (Figure 6.5).

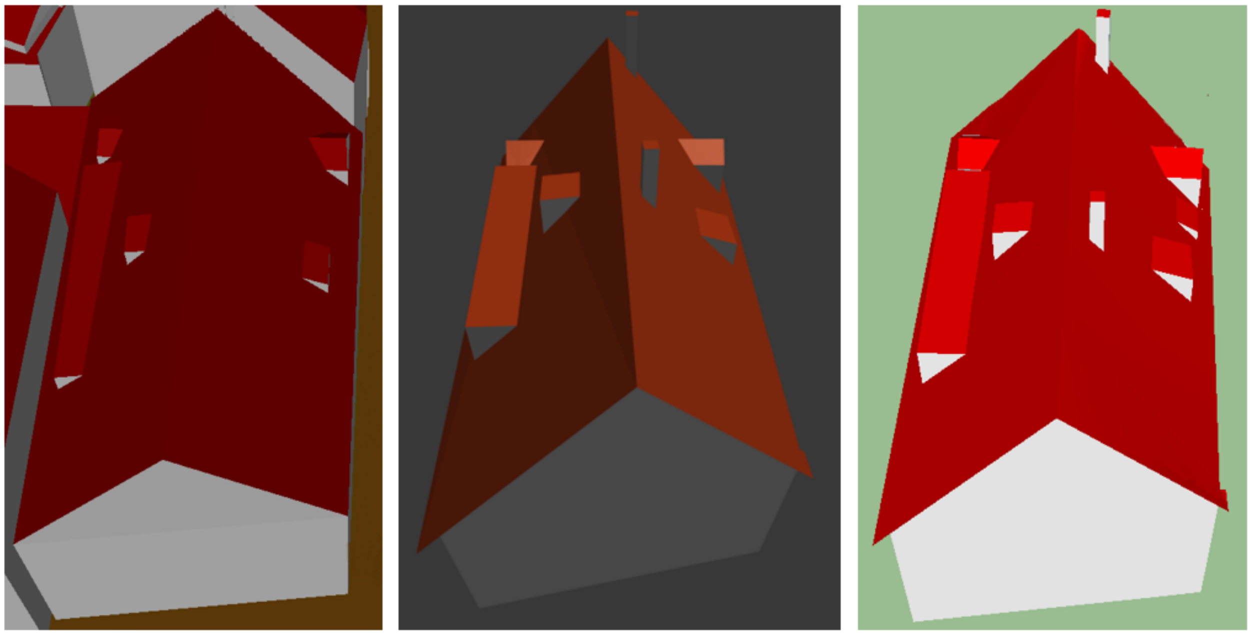


Figure 6.5: A 3D Building model with gable roof by 3D Editor (left), by CityGRID Modeler (middle), by SketchUp (right)

The generation of a 3D building with a standard roof requires manual afford for both 3D Editor and SketchUp, whereas CityGRID Modeler accomplishes it automatically by triangulation algorithm. As long as the input data has no geometrical error, the user does not need to interfere with CityGRID Modeler while modeling.

Roof details can be generated manually by mean of 3D Editor and SketchUp as well. With CityGRID Modeler, this task can be completed automatically by using a special tool. As it can be seen in Figure 6.5, the building model with CityGRID Modeler and SketchUp are fairly similar, while the roof detail of the building model with a 3D editor has smaller roof details.

In Figure 6.6, another 3D building model, which has been correctly generated by each software package, can be seen. The building originally has two-pieces gable roof type, a terrace and a winter garden on the terrace. 3D Editor does not have such a building primitives. Therefore, the building had to be split into three parties, where the roof type changes and each part has been modeled individually. The roof type of the winter garden on the terrace cannot be generated by any building primitives in 3D Editor. Hence, it has also been by generated by a combination of gable and flat roof. Combined building primitives approach helps the user to model also irregular building. However, it requires more time to model such buildings more accurately. For CityGRID, the same automatic process has been pursued as for the previous example in Figure 6.2. With SketchUp, the building and roof details have been mainly modeled manually.

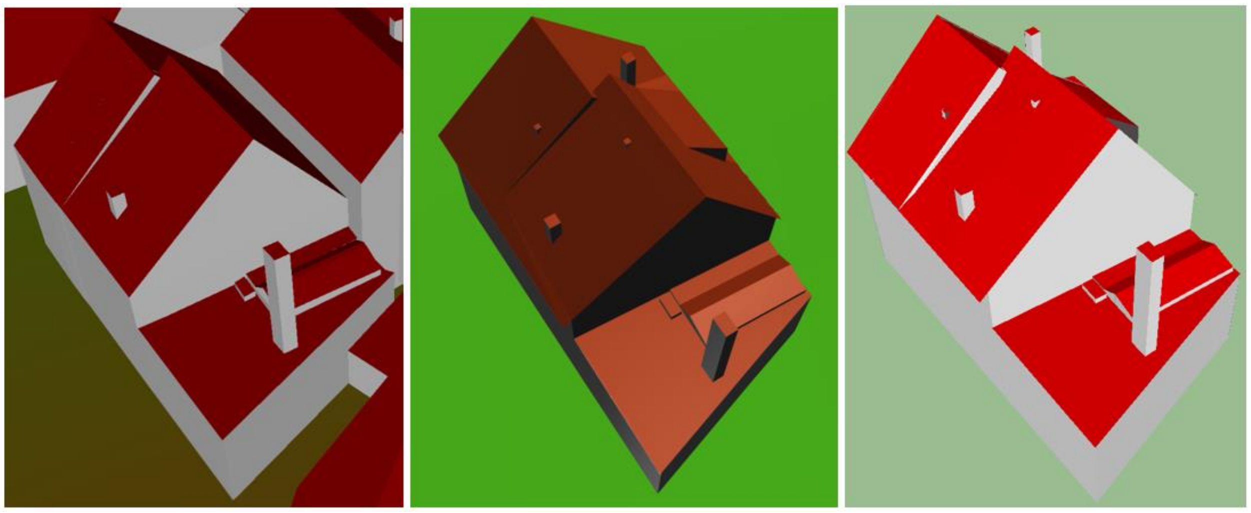


Figure 6.6: A 3D Building model with terrace and winter garden by 3D Editor (left), by CityGRID Modeler (middle), by SketchUp (right)

The building model in Figure 6.7 has a less common roof type and building structure. With 3D Editor, it has been modeled by combination gambrel roof and tent roof. The model with 3D Editor is less realistic than the models with CityGRID Modeler and SkethUp. The results of with CityGRID Modeler and SkethUp are quite similar. However, SketchUp necessitates more time than CityGRID Modeler.

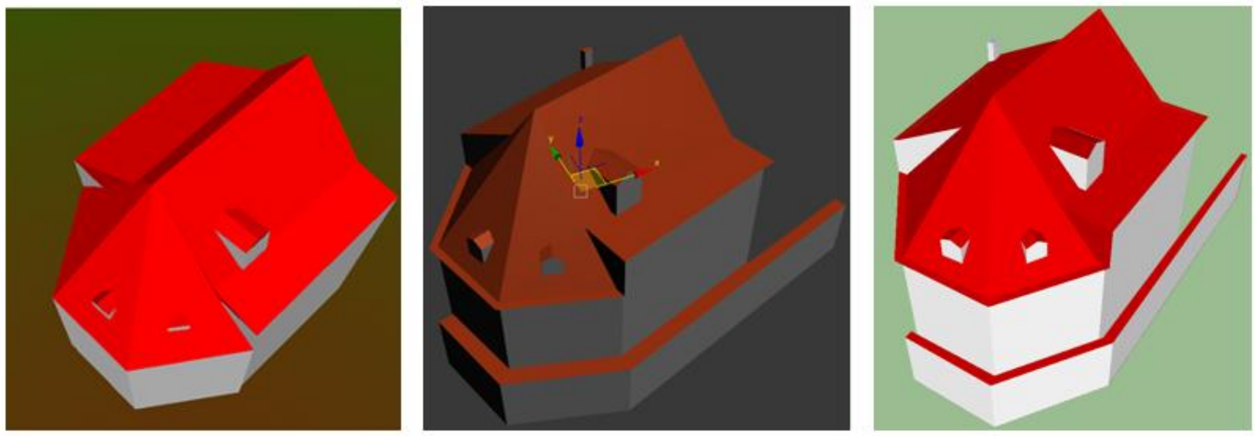


Figure 6.7: A 3D Building model with irregular roof type by 3D Editor (left), by CityGRID Modeler (middle), by SketchUp (right)

The building with terrace in Figure 6.8 can be successfully modeled only by CityGRID Modeler and SketchUp. The part of the roof, where the balcony is located in the building, has been modeled as the flat roof in 3D Editor as predefined building primitives of 3D Modeler do not contain a balcony or terrace type. However, if it´s required, 3D Editor can also accomplish modelling the terrace. For that, first one flat roof must be generated, and on the top of it, side terrace parapets can be modeled on the flat roof by creating a very tiny flat roof. Side terrace parapets must be located in a way that the outline of the flat roof under terrace parapets and the outlines of the terrace parapets must overlap. Because of that, it´s rather tedious work and requires high attention.

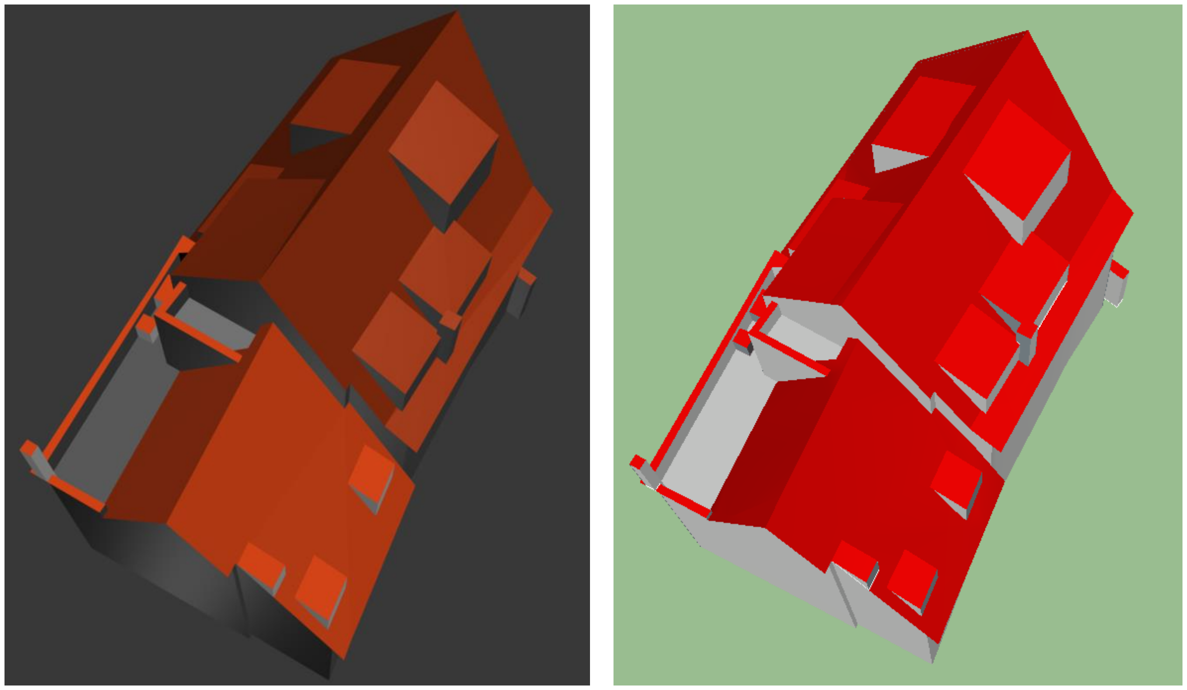


Figure 6.8: A 3D Building model with terrace by CityGRID Modeler (left) and by SketchUp (right)

3D Editor does not allow the generation of the roof overhang. It can only be accomplished by CityGML Export tool of Hexagon for single buildings while exporting the model. The roof overhang generation is possible for both CityGRID Modeler and SketchUp. The comparison of the software packages regarding other building components can be seen in Table 6.2.

Software	Roof Details	Roof Overhang	Facade Details	Balcony, Terrasse	Building passage, Tunnel	Tower	Interior features
3D Editor	yes	no	no	no	no	yes	no
CityGRID Modeler	yes	yes	yes	yes	yes	yes	no
SketchUp	yes	yes	yes	yes	yes	yes	yes

Table 6.2: Comparison of the modeling capability of the software

In terms of level of detail, the resulting model with SketchUp and CityGRID Modeler are richer than the model with 3D Editor. The model with SketchUp has same content regarding modeled building, roof details and roof overhang as the model with CityGRID Modeler.

SketchUp enables also modeling the interior features of the building. Therefore, SketchUp is one of the assets software packages that it is capable of generating LOD4 building models (Table 6.3).

Software	Level of Detail Capacity
3D Editor	LOD2

CityGRID Modeler	LOD2, LOD3
SketchUp	LOD1, LOD2, LOD3, LOD4

Table 6.3: Comparison of Level of Detail Capacity

6.6 Comparison of Required Time

The time spent on generation of a 3D city model is an important factor that needs to be taken into consideration when deciding which 3D city modeling software package to use.

Provided one wants to generate a highly detailed model the most amount of time is required with 3D Editor and SketchUp, as the most of the work has been done manually throughout the case study.

There is little manual work associated with creating the model with CityGRID Modeler. It does not require active interference of an operator and batch mode while modeling is possible. Therefore, the computer can be left over to work on generation the model. However, one should not forget that CityGRID Modeler requires input data that must have a line structure which is mostly extracted from photogrammetric restitution. Extracting of line structure of roof and roof details is the most time-consuming step during photogrammetric restitution. According to 3D City Modeling Project for Konya Metropolitan Municipality, an experienced photogrammetric interpretation operator has digitized an average of 60-80 building roofs per day for the Konya project. But this amount can change according to the complexity of the roof structure (Özerbil, 2015). Therefore, CityGRID Modeler requires long and expensive pre-processing before the modeling can start.

6.7 Comparison of Easiness of software

Modeling with **3D Editor** is not easy and requires an understanding of photogrammetry as well as a lot of practices. **SketchUp** is understandable and easy to use. Therefore, the users can learn very quickly on their own. CityGRID Modeler is much harder to understand than SketchUp. Training is, therefore, mandatory to create 3D city models from **CityGRID Modeler**. Compared to CityGRID Modeler and 3D Editor, SketchUp provides a short learning curve for 3D modeling.

6.8 Comparison of Texturing Process

Texturing highly enhances the visual impression of 3D city model. For texturized building modeling, each building facade requires a realistic texture. Therefore, mapping texture onto roofs and facades of the building model is very time-consuming, and it may take days depending on the size of the city. Texturing can be based on different principles such as by color or pattern libraries or photographs.

3D Editor itself does not offer texturing functionality. TextureMapper software from Hexagon, which has been used in the case study, takes over the texturing task from 3D Editor. CityGRID Modeler enables texturing directly through CityGRID Modeler both semi-automatically and manually (see 5.2.4). However, CityGRID Administrator has been used for texturing purposes in the case study in order to accomplish the task completely automatically.

The precondition for automatic texturing with TextureMapper and CityGRID Administrator are geo-referenced aerial images with high overlap. TextureMapper does not allow editing texture of the building. As it can be seen in Figure 6.9, the results of the texture processing are quite good. The correct texture images have been assigned to correct surfaces with both software as long as proper images are available.

According to the log file of TextureMapper, the texturing duration is 2 hours and 21 minutes. TextureMapper offers the possibility to work in batch process when series of the model need to be texturized.

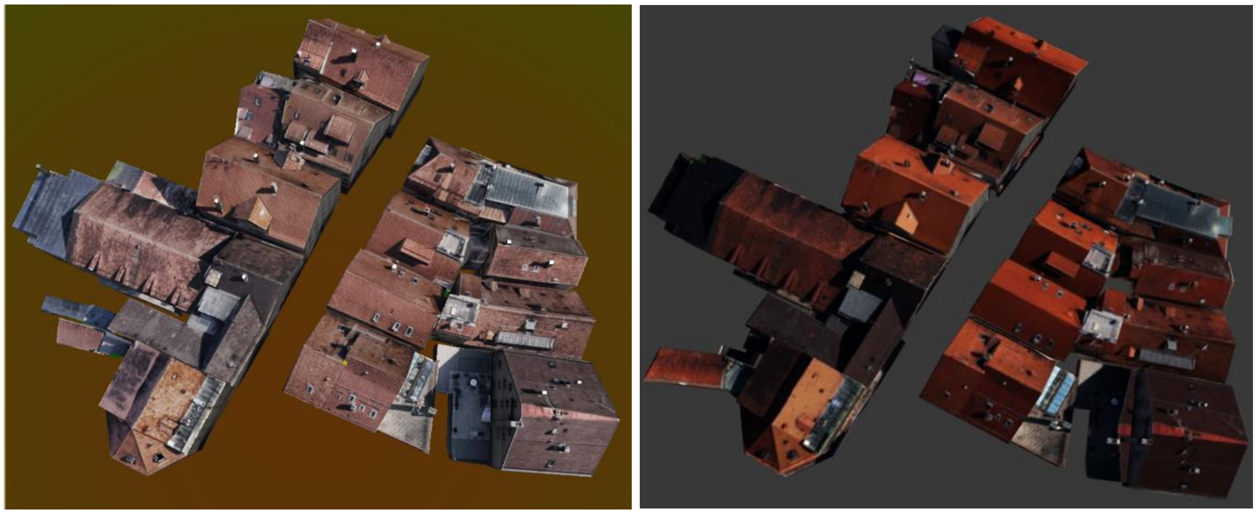


Figure 6.9: Output of case study by TextureMapper Modeler (left) and by CityGRID Administer (right)

With SketchUp, models cannot be texturized automatically. It does not require geo-referenced aerial images. As the texturizing is possible only with manual effort, the quality of the texture is not advanced. Unlike the other software packages, SketchUp allows the user to texture the model with the artificial material (e.g. color or material library). Additionally, it can obtain texture images directly from Google Earth or Google Street View.

Software	**Automatic Texturing of Roof**	**Automatic Texturing of facade**	**Texturing with non- oriented images**	**Texturing with other material**
3D Editor	yes	yes	no	no
CityGRID Modeler	yes	yes	yes	no
SketchUp	no	no	yes	yes

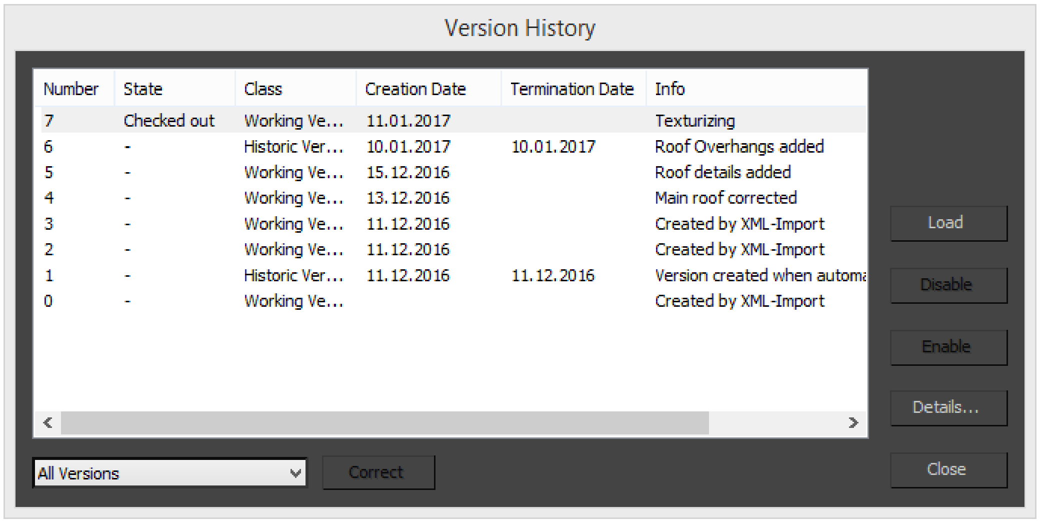
Table 6.4: Comparison of Texturing Capacity

6.9 Comparison of Updating Process

Nowadays, the cities grow rapidly. Consequently, 3D city model must be periodically maintained and updated with new data.

3D Editor and SketchUp do not a special tool for an update process and require manual work for updating. Neither software offers any possibility to manage the different versions of the models.

CityGRID Modeler allows managing the complete history of building models. Several versions of the model are accessible through the database. Older version can be loaded and compared to the actual version. Modeler also enables the model to load and export at created historic version. Figure 6.10 shows an example of a version history of a model. The version is listed with respect to creation date. There are two different classes of versions: Working versions and Historical versions. Working versions may be created for backup purposes while modeling. Version 2 to 5 in figure 6.10 comprise modeling step and hence are assigned as working versions. A historical version is often created when a modeling comes to an end. Creation and termination date are assigned to this version. This gives the possibility to question the model at a specific date. For instance, querying the database for the data 10.01.2017 returns the version 6 of the model to the user. A historical version is technically a working version and can be further edited if it requires.



Number	State	Class	Creation Date	Termination Date	Info
7	Checked out	Working Ve...	11.01.2017		Texturizing
6	-	Historic Ver...	10.01.2017	10.01.2017	Roof Overhangs added
5	-	Working Ve...	15.12.2016		Roof details added
4	-	Working Ve...	13.12.2016		Main roof corrected
3	-	Working Ve...	11.12.2016		Created by XML-Import
2	-	Working Ve...	11.12.2016		Created by XML-Import
1	-	Historic Ver...	11.12.2016	11.12.2016	Version created when automa
0	-	Working Ve...			Created by XML-Import

Figure 6.10: Version History window of CityGRID Modeler

CHAPTER 7

The thoughts and conclusion regarding the practical experience and the result of case study have been presented in this chapter.

7 CONCLUSIONS

The thesis aimed to explore, asses and compare the potential and performance of selected 3D city modeling software packages, which have different approaches and methods for 3D city modeling. In order to achieve the goal, a case study has been carried out on a small area in the city Reutlingen, Germany, which has been considered as a prototype of a large city.

In chapter 2, the fundamentals of 3D modeling have been surveyed. The necessity and importance of 3D model generation of real objects have been described, and the representative methods for the generation of 3D city model have been presented. Furthermore, the mainly used 3D city model formats have been presented.

In chapter 3, the software packages, which are aimed to be used in this thesis have been introduced. The requirements for each and every software explained. Besides this, their strengths and weaknesses in 3D city modeling have been pointed out.

The subsequent steps for the generation of exemplary 3D city model for each and every software packages have been described in chapter 5. Data requirements of the software packages have been explained in order to accomplish the predefined tasks. Furthermore, the difficulties during the modeling process have been pointed out.

In chapter 6, the results of the case study have been discussed, and the models have been compared. The results show that all three software packages are suitable for the generation of 3D city model. CityGRID Modeler and SketchUp make possible 3D city model with LOD3, whereas 3D Editor can generate a 3D model with LOD2 including roof details. The model with **CityGRID Modeler** has the highest potential and details, which requires least manual effort. CityGRID Modeler provides the best solution when a large city with a high level of details is planned to be modeled. The drawback of the software, it requires well-defined roof structures as an input, which makes the pre-processing longer than SketchUp and 3D Editor.

The model with **3D Editor** has a lower potential in terms of building details compared to other models. The main reason for that is the model-driven approach of 3D Editor is, which makes model dependent on the building primitives that are in available in the building library. However, the benefit of the software is the straight forward production. It gives a good result for not complex buildings. Particularly, an LOD2 city model can be modeled very fast by using a combination of City Modeller and 3D Editor with minimum manual effort, and the results can be easily exported to many formats. Hence, the model is suitable for applications, which do not require a high level of detail and fulfills the intention of the software developers. In addition to this, the lack of details may then be compensated by the 3D impression of the texture.

TextureMapper enables texturing the generated LOD2 models highly automatically and thus provides results which are best suited for medium scale visualization.

The model with **SketchUp** looks as good as the model CityGRID Modeler. The benefit of the use of SketchUp is an easy tool and very flexible in terms of the level of detail. However, it is only useful for very small areas, preferably for individual buildings as all work requires manual effort including texturing process. The cost of SketchUp is significantly lower than CityGRID Modeler and 3D Editor. Therefore, if 3D city modeling project has a limited budget, SketchUp may be the optimal choice. However, it should be taken consideration that SketchUp may not be preferable for GIS as well as surveying applications, as the coordinate reference system may not always be preserved while importing and exporting the models.

In summary, the case study has indicated that each and every software package has advantages as well as drawbacks dependent on the point view from which expectations have been defined. If one clearly sticks to the intended sort of application, the severeness of disadvantages may decrease significantly, while the advantages rise. A software package may give sufficient results for a specific project, and the same product may not be financially achievable for other 3D city modelling projects. For the determination of an optimal software package for a certain 3D city model project, at first the requirements of the project and the composition of the city must be defined, the feasibility seriously analyzed by taking into consideration the composition and complexity of the city. By keeping in mind those general conditions, this study does not dare to categorize the software packages into winners and losers.

Eventually, there are two issues which have to be mentioned. Firstly, the assessment has been carried out by an operator who was not expert in any of the three software packages. A skilled operator might have come to one or the other different conclusion. The more complicated the tools are, the more expertise is needed to optimally and efficiently obtain a high quality result. Secondly, it should be emphasized that approaches to 3D city modelling are still in the focus of ongoing development and research. The selection of software concentrated on available commercial tools rather than pure research activities. Since companies provide more or less frequent updates to their products when new theoretical ideas, new algorithms, new data sources, new user requirements, or more convenient user interfaces are available, the assessments carried out in this work may just be seen as a kind of snapshot of the time when this evaluation commenced. Still, the selection of software packages has also been a selection of basic approaches and, therefore, the assessment is able to show the shortcomings and advantages of the different approaches in a more general way, thus providing longer lasting validity

8. REFERENCES

1. *3DCon GmbH (2000 – 2013) Manual tridicon® 3D Editor*
2. *Bahu, J. M., Koch, A., Kremers, E., & Murshed, S. M. (2013). Towards a 3D spatial urban energy modelling approach. ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences, (1), 33-41.*
3. *Baltsavias, E. P., & Gruen, A. (2003). Resolution convergence: a comparison of aerial photos, LIDAR and IKONOS for monitoring cities. Remotely sensed cities, 47-82.*
4. *Biljecki, F. (2013). The concept of level detail in 3D city models: PhD Research Proposal.*
5. *Biljecki, F., Ledoux, H., Stoter, J., & Zhao, J. (2014). Formalisation of the level of detail in 3D city modelling. Computers, Environment and Urban Systems, 48, 1-15.*
6. *Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., & Çöltekin, A. (2015). Applications of 3D city models: State of the art review. ISPRS International Journal of Geo-Information, 4(4), 2842-2889.*
7. *Bratfisch, Sylvia, Ing Robert Roschlaub, and Dipl-Ing Josef Dorsch (2014). LoD3-Gebäudemodelle in Bayern –alternative technische Lösungsansätze. Paris Lodron University of Salzburg.*
8. *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.*
9. *Döllner, J., Baumann, K., & Buchholz, H. (2006). Virtual 3D city models as foundation of complex urban information spaces. Competence Center of Urban and Regional Planning (pp. 107-112).*
10. *Eran, S., Sayed, J., & Rosdi, M. (2005). The Design and Development of a Virtual 3D City Model. Department of Surveying Science and Geomatics Faculty of Architecture, Planning and Surveying UiTM.*
11. *Förstner, W. (1999). 3D-city models: automatic and semiautomatic acquisition methods. Photogrammetric Week '99.*
12. *Franic, S., Bacic-Deprato, I., & Novakovic, I. (2009). 3D model and a scale model of the City of Zagreb. In ISPRS WG II/2, II/3, II/4 Workshop on quality, scale and analysis aspects of city models, Lund, Sweden (pp. 1-7).*
13. *Frueh, C., Sammon, R., & Zakhor, A. (2004, September). Automated texture mapping of 3D city models with oblique aerial imagery. In 3D Data Processing, Visualization and Transmission, 2004. 3DPVT 2004. Proceedings. 2nd International Symposium on (pp. 396-403). IEEE.*
14. *Grün, A., Baltsavias, E., Henricsson, O. (Eds.), 1997. Automatic Extraction of ManMade Objects from Aerial and Space Images (II). Birkhäuser, Basel.*
15. *Grün, A., Kübler, O., Agouris, P. (Eds.), 1995. Automatic Extraction of Man-Made Objects from Aerial and Space Images. Birkhäuser, Basel*
16. *Gröger, G., Kolbe, T. H., Nagel, C., & Häfele, K. H. (2012). OGC city geography markup language (CityGML) encoding standard, Version 2.0, OGC doc no. 12-019. Open Geospatial Consortium.*

17. *Heritage, G. L., & Large, A. R. (2009). Principles of 3D laser scanning. Laser scanning for the environmental sciences, 21-34.*
18. *Hexagon AB (2009 – 2015) Manual tridicon® CityModeller*
19. *Hexagon AB (2006 – 2015) Manual tridicon® TextureMapper*
20. *Hexagon AB (2005 – 2015) Manual tridicon® Export*
21. *Huang, J. (2008). 3D MODEL GENERATION USING MULTIPLE VIEW IMAGES.*
22. *Kerschner, M. (2004). Generating and managing virtual city models with the CITYGRID system. In 1º Congreso Internacional Ciudad y Territorio Virtual, Barcelona, 3, 14 y 15 Septiembre 2004 (pp. 47-53). Centre de Política de Sòl i Valoracions.*
23. *Kolbe T. H. (11/2005). CityGML – Status and Roadmap. OGC TC Meeting Bonn.*
24. *Kurakula, V., Skidmore, A. K., Kluijver, H., Stoter, J., Dabrowska-Zielinska, K., & Kuffer, M. (2007). A GIS-based approach for 3D noise modelling using 3D city models. MSc proposal, University of Southampton, UK.*
25. *Löwner, M. O., Casper, E., Becker, T., Benner, J., Gröger, G., Gruber, U., ... & Schlüter, S. (2013). CityGML 2.0–ein internationaler Standard für 3D-Stadtmodelle, Teil 2: CityGML in der Praxis. Zeitschrift für Geodäsie, Geoinformation und Landmanagement, 2(2013), 131-143.*
26. *Maas, H. G. (1999). Closed solutions for the determination of parametric building models from invariant moments of airborne laserscanner data. transformation, 2, 20.*
27. *Mao, B. (2011). Visualisation and generalisation of 3D City Models (Doctoral dissertation, KTH Royal Institute of Technology).*
28. *Marre, F. (2011). Photogrammetry or LIDAR. GeoConnextion Geospatial. GIS, Spatial Technologies Magazine.*
29. *Morgan, M., & Habib, A. (2002, April). Interpolation of lidar data and automatic building extraction. In ACSM-ASPRS Annual conference proceedings (pp. 432-441).*
30. *Moskal, L. Monika (2008). "Lidar fundamentals.", Workshop on Site-scale Applications of LiDAR on Forest Lands in Washington*
31. *Navratil, G., Bulbul, R., & Frank, A. U. (2010). Maintainable 3D models of cities.*
32. *Nex, F., & Remondino, F. (2012). Automatic roof outlines reconstruction from photogrammetric DSM. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 1(3), 257-262.*
33. *Ozerbıl, T., Gokten, E., Onder, M., Selcuk, O., Sarılar, N. C., Tekgul, Tutuneken, A. (2015). Oblique Aerial Image Acquisition, 3D City Modeling, 3D City Guide Project for Konya Metropolitan Municipality. International Journal of 3-D Information Modeling (IJ3DIM), 4(2), 34-47.*
34. *Qiming, Z., & Wenjiang, Z. (2004). A preliminary review on three-dimensional city model. Geo-Spatial Information Science, 7(2), 79-88.*
35. *Previtali, M., Barazzetti, L., Brumana, R., Cuca, B., Oreni, D., Roncoroni, F., & Scaioni, M. (2014). Automatic façade modelling using point cloud data for energy-efficient retrofitting. Applied Geomatics, 6(2), 95-113.*
36. *Rothe, D. I. R., & Janne, D. I. F. C. (2009). Vergleich des tridicon™ CityDiscoverer und Autodesk® LandXplorer zur Nutzung von 3D Stadtmodellen.*

37. *Rottensteiner, F., Kager, H., Briese, C., & Kraus, K. (2002, October). LIDAR activities at the Viennese Institute of Photogrammetry and Remote Sensing. In Proceedings of the 3rd Int. LIDAR Workshop, Columbus.*
38. *Sadek, E., Sadek, S. B., Ali, S. J., & Kadzim, M. R. B. (2002). The Design and Development of a Virtual 3D City Model. Retrieved December, 18, 2015.*
39. *Schuckman, Karen (2014). "Characteristics of Lidar Data" Geography, College of Earth and Mineral Sciences, The Pennsylvania State University.*

40. *Schwalbe, E., Maas, H. G., & Seidel, F. (2005). 3D building model generation from airborne laser scanner data using 2D GIS data and orthogonal point cloud projections. Proceedings of ISPRS WG III/3, III/4, 3, 12-14.*
41. *Singh, S. P., Jain, K., & Mandla, V. R. (2014). Image based 3D city modeling: Comparative study. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 40(5), 537.*
42. *Singh, S. P., Jain, K., & Mandla, V. R. (2013). Virtual 3D city modeling: techniques and applications. ISPRS-International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, 1(2), 73-91.*
43. *Stadler, A., & Kolbe, T. H. (2007, June). Spatio-semantic coherence in the integration of 3D city models. In Proceedings of the 5th International Symposium on Spatial Data Quality, Enschede.*
44. *Tarsha-Kurdi, F., Landes, T., Grussenmeyer, P., & Koehl, M. (2007). Model-driven and data-driven approaches using LIDAR data: Analysis and comparison. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, 36, 3-W49A.*
45. *Uggla, G. (2015). 3D City Models-A Comparative Study of Methods and Datasets. Master Thesis in Geoinformatics, KTH Stockholm.*
46. *UVM Systems GmbH (2013) Manual CityGRID® Grundlagen*
47. *UVM Systems GmbH (2013) Manual CityGRID® Modeler*
48. *UVM Systems GmbH (2013) Manual CityGRID® Administrator*
49. *Verma, V., Kumar, R., & Hsu, S. (2006). 3d building detection and modeling from aerial lidar data. In Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on (Vol. 2, pp. 2213-2220). IEEE.*
50. *Vosselman, G. (2002). Fusion of laser scanning data, maps, and aerial photographs for building reconstruction. In Geoscience and Remote Sensing Symposium, 2002. IGARSS'02. 2002 IEEE International (Vol. 1, pp. 85-88). IEEE.*
51. *Wolf, P. R., & Dewitt, B. A. (2000). Elements of Photogrammetry (with Applications in GIS).*
52. *Zhou, G., Song, C., Simmers, J., & Cheng, P. (2004). Urban 3D GIS from LiDAR and digital aerial images. Computers & Geosciences, 30(4), 345-353.*

APPENDIX A: Supported Object Type by Tridicon® 3D Editor

The following object types can be supported at present[12]:

- Point
- Line (one start and one end point)
- Polyline (consists of at least three points, maximum number is variable)
- Polygon (always closed, maximum number is variable)
- Circle (construction with center/radius or with three points)
- Text
- Symbols (for insertion of symbols:

The following objects are only available as 3D objects

- Flat roof with quadrangular ground plane
- Flat roof with polygon as ground plane (polyflat roof)
- Lean-to roof
- Tent roof with quadrangular ground plane
- Tent roof with polygon as ground plane (polytent roof)
- Gable roof
- Hip roof
- Hipped gable roof
- Combination Gable/Hip roof
- Combination Gable/Hipped gable roof
- Combination Hip/Hipped gable roof
- Cut gable roof
- Cut hip roof
- Combination Cut gable / Cut hip roof
- Gambrel roof
- Gambrel hip roof
- Combination Gambrel hip / Gambrel roof
- Barrel roof
- Tower (circular ground and cover plane; different diameters possible)
- Tower roof (consist of a cylindric tower with conic roof)
- Dome roof
- Wall
- Extruded Polyline
- Extruded Polygon

[12] Source: [CityModeler Manual, 2009 -2015, S. 44-45].