3B ŞEHİR MODELLERİNİN OTOMATİK ÜRETİMİ VE ETKİLİ GÖRSELLEŞTİRİLMESİ

AUTOMATIC RECONSTRUCTION AND EFFICIENT VISUALIZATION OF 3D CITY MODELS

MEHMET BÜYÜKDEMİRCİOĞLU

PROF. DR. SULTAN KOCAMAN GÖKÇEOĞLU

Supervisor

Submitted to

Graduate School of Science and Engineering of Hacettepe University

as a Partial Fulfillment to the Requirements

for the Award of the Degree of Doctor of Philosophy

in Geomatics Engineering

2023

ÖZET

3B ŞEHİR MODELLERİNİN OTOMATİK ÜRETİMİ VE ETKİLİ GÖRSELLEŞTİRİLMESİ

Mehmet BÜYÜKDEMİRCİOĞLU

Doktora, Geomatik Mühendisliği Bölümü

Tez Danışmanı: Prof. Dr. Sultan KOCAMAN GÖKÇEOĞLU

Mart 2023, 114 sayfa

Günümüzde kırsal bölgelerden göç, sürekli değişim ve şehirlerin yapısının giderek karmaşık hale gelmesi, şehir yönetimlerinde verimin artması için yeni yöntem arayışlarını beraberinde getirmektedir. 3 Boyutlu (3B) şehir modellerine olan talep giderek artmakta ve birçok ülkede çeşitli yönetim seviyelerinde farklı ölçeklerde aktif olarak kullanılmaktadır. 3B şehir modelleri sadece görsel modeller olmayıp, aynı zamanda sahip oldukları semantik veriler yardımıyla analizlere ve farklı görselleştirme uygulamalarına izin vermektedir. Bu modeller farklı detay seviyelerinde (LoD) üretilebilmekte ve seviyeleri arttıkça, bina ve çatıya ait modellenen nesnelerin/ayrıntıların miktarı da artmaktadır. Yüksek detay seviyesine sahip 3B modeller, genel olarak çok yüksek çözünürlüklü stereo hava fotoğrafları yardımıyla fotogrametri operatörleri tarafından manuel olarak çizilmektedir. Bu çok fazla insan gücü, zaman ve maliyet gerektiren bir süreçtir. Yüksek detaylı 3B şehir modellerini otomatik üretmek için literatürde farklı yaklaşımlar bulunmaktadır ancak bu konu tam olarak çözülememiş bir sorun olarak birçok araştırmacı tarafından çalışılmaya devam etmektedir. Üretilen modellerin verimli görselleştirilmesi ise, kullanılacak görselleştirme platformuna göre birçok optimizasyon ve farklı yaklaşım gerektiren ayrı bir sorundur. Bu tez kapsamında, çok yüksek çözünürlüklü optik görüntülerden otomatik olarak bina çatı tiplerinin otomatik olarak sınıflandırılması, bina ayak izlerinin çıkartılması, LoD2.2 seviyesinde çatı detaylarının çıkartılarak 3B bina modeli üretiminde kullanılmasına yönelik derin öğrenme tabanlı çözümler geliştirilmiştir. Çalışma sahaları Türkiye'nin farklı bölgelerinden seçilmiş ve kullanılan derin öğrenme yöntemine uygun şekilde öğrenme verisi hazırlanmıştır. Bu çalışmalarda elde edilen sonuçlar detaylı olarak incelenerek, potansiyel iyileştirmelere yönelik öneriler sunulmuştur. Ayrıca, LoD2 ve LoD3 şehir modellerinin görselleştirilmesine dair farklı çözümler geliştirilerek tartışılmıştır. Bu amaçla hem Web tabanlı Cesium kütüphanesi hem de sanal gerçeklik destekli Unity oyun motorunda görselleştirilme çalışmaları tamamlanmış ve farklı avantaj ve dezavantajları ortaya konulmuştur.

Anahtar Kelimeler: 3B Şehir modelleri, Derin Öğrenme, CityGML, Cesium, Sanal Gerçeklik

ABSTRACT

AUTOMATIC RECONSTRUCTION AND EFFICIENT VISUALIZATION OF 3D CITY MODELS

Mehmet BÜYÜKDEMİRCİOĞLU

Doctor of Philosophy, Department of Geomatics Engineering

Supervisor: Prof. Dr. Sultan KOCAMAN GÖKÇEOĞLU

March 2023, 114 pages

Today, migration from rural areas, continuous change in cities and the increasing complexity of their structure yielded the need of new methods to increase efficiency in their management. The demand for 3 dimensional (3D) city models is increasing and they are actively used by countries and municipalities at different scales. 3D city models are not only visual models, but also allow analysis and different visualization applications with the help of their semantic data. These models can be produced in different levels of detail (LoD), and as the levels increase, the amount of modeled objects/details of the building and roof also increases. 3D models with a high level of detail are produced manually by photogrammetry operators, usually with the help of very high resolution stereo aerial photographs. However, this process is costly in terms of labor and time. There exist different approaches in the literature to automatically generate highly detailed 3D city models, but the topic is still an active research area being investigated by several researchers. On the other hand, efficient visualization of the produced models also involves optimization issues and depending on the platform, and different approaches exist. Within the scope of this thesis, deep learning-based solutions have been developed for automatic classification of building roof types from very high resolution optical imagery, automatic extraction of building footprints, automatic extraction of roof details at LoD2.2 level and their use in the production of 3D building models. The study sites were selected from different regions of Türkiye and the training data were prepared in accordance with the requirements of the deep learning methods. The results are presented and suggestions for potential improvements are discussed. In addition, different solutions for the visualization of LoD2 and LoD3 city models are developed and discussed. For this purpose, web-based visualization with Cesium library and virtual reality supported Unity game engine were employed to reveal various advantages and disadvantages of both approaches.

Keywords: 3D City models, Deep Learning, CityGML, Cesium, Virtual Reality

ACKNOWLEDGMENTS

I didn’t know what future may bring when I started my PhD. In the end, I am greatful for my choice and have completed this journey with a lot of good memories. This thesis wouldn’t be completed without the help and support of many people.

My biggest thank goes to my supervisor Prof. Dr. Sultan Kocaman Gökçeoğlu. If I didn’t have the opportunity to work with her, I probably would never have pursued a PhD. I learned a lot from her, both academically and about life. She supported my every decision, and helped me become the person I am today. I will always be honored to be her student.

I am also very thankful to my doctoral committee members Dr. Nusret Demir and Dr. Murat Durmaz for their valuable comments on the thesis progress, and also Dr. Ali Özgün Ok and Dr. Ümit Işıkdağ for taking their time and contributing to the thesis as jury members.

I also would like to thank my best friend Gökhan and my colleagues and close friends Ilyas, Gizem and Recep at the Geomatics Engineering department at Hacettepe University. We have always helped and motivated each other whenever needed. Thank you for the great memories and continuous support during this journey.

Another big thanks goes to Prof. Dr. Martin Kada for accepting me as visiting researcher at TU Berlin. I have spent one amazing and very productive year at Berlin. He provided me an office with a nice view and a quiet environment to study. His supervision and his courses extended my knowledge about deep learning. I will always be grateful for his support.

I want to acknowledge financial support of the Scientific and Technological Research Council of Turkey (TÜBITAK) 2214-A International Doctoral Research Fellowship Programme during my one year visit at TU Berlin and 2224-A Grant Program for Participation in Scientific Meetings Abroad for attending ISPRS Congress 2022 in Nice, France. I also acknowledge financial support of Turkish Council of Higher Education (YÖK) for 100/2000 Ph.D. scholarship for 3 years. I also would like to thank General Directorate of Land Registry and Cadastre of Turkey for providing the data used in this thesis.

CONTENTS

LIST OF TABLES

LIST OF FIGURES

SYMBOLS AND ABBREVIATIONS

Abbreviations

BIM	Building Information Modeling
CAD	Computer Aided Design
CNN	Convolutional Neural Network
CSV	Comma-seperate Value
DL	Deep Learning
DSM	Digital Surface Model
DTM	The Digital Terrain Model
EOP	Exterior Orientation Parameters
FME	Feature Manipulation Engine
GAN	Generative Adversarial Network
GDLRC	The General Directorate of Land Registry and Cadastre of Turkiye
IOP	Interior Orientation Parameters
LoD	Level of Detail
LiDAR	Light Detection and Ranging
MVS	Multi-view Stereo
MOEU	Ministry of Urbanization and Environment
OGC	Open Geospatial Consortium
TMS	Tile Map Service
UAV	Unmanned Air Vehicle
UTM	Universal Transverse Mercator
WMTS	Web Map Tile Service

1. INTRODUCTION

Buildings are among the important objects in a city. As the population of cities continues to increase every day, management of cities become more complex. The production and use of virtual 3D city models and digital twins of cities have been increasing in parallel for efficient management and planning with simulations using the generated models. 3D city models can be generated with different levels of detail (LoD) [1] depending on the application requirements. As a result of higher LoD and rich semantic information, the number, variety, and quality of the analyses that can be performed on 3D city models also increase.

3D city models can be reconstructed manually, semi-automatically, or fully automatically. The process of manually reconstructing a 3D model of a city is commonly carried out by photogrammetry operators through manual digitization of roofs or 3D building models from stereo aerial imagery. However, it is very time-consuming and costly to manually digitize large numbers of buildings in cities. With the semi-automatic and automatic approaches, building geometries can be reconstructed from point clouds, mostly from LiDAR (Light Detection and Ranging) sensors. Point clouds obtained from optical images through stereo processing methods are preferred less due to sparse point distribution in low-texture areas. Thus, roof corners or roof details may not always be generated with optical (often called as photogrammetric) point clouds. Although LiDAR point clouds may present object shapes better than optical photogrammetric point clouds depending on image resolution and texture, the LiDAR data are often not available due to their high cost.

Automatic and semi-automatic 3D building reconstruction methods usually require building footprints as input along with a Digital Surface Model (DSM). The main reason for this is that these methods usually reconstruct each building by clipping the respective part from the DSM based on footprint area. Open data sources for building footprints such as OpenStreetMap [2], and Microsoft buildings [3] can also be used for this purpose. The main issue lies in the fact that these datasets are typically produced either manually or automatically with artificial intelligence (particularly machine learning) methods from satellite images, which usually have lower spatial resolution than aerial photos or LiDAR data. Based on the image spatial resolution and the method used to generate the data of the building footprints, openly available datasets are often not fully compatible with higher-resolution datasets. Due to this fact, building footprints are usually delineated manually using mono or stereo plotting methods, which is costly for time and labor. Automatically extracting building footprints will enable tremendous cost and time savings, as well as a reduction in the complexity of the process.

Novel deep learning (DL) methods can be more effective than the traditional methods of computer vision and photogrammetry [4]. The use of the DL technologies has transformed many tasks that were previously performed manually or with low accuracy into automated tasks with high accuracy. Automatic reconstruction of 3D city models with high detail levels (LoD2+) is still an unsolved problem, yet field of research in DL-based 3D building reconstruction is still in its early stages and there is still a great deal to be done. Through the use of the DL methods, roof structure extraction and 3D building reconstruction could be performed more accurately by overcoming some of the limitations of conventional methods.

This thesis aimed to develop a framework for DL-based reconstruction and visualization of 3D city models towards full-automatization and high efficiency. Although not all steps could be fully-automatically realized, several investigations were carried out to realize the automatization and reveal the major issues. Towards achieving this goal, a number of DL-based methods has been investigated and proposed using very high-resolution stereo aerial imagery for roof type classification, building footprint extraction, and LoD2.2 building roof structure extraction for 3D building reconstruction. For this purpose, diverse datasets from different geographical regions of Türkiye were utilized to train various DL methods. Each study was conducted in a different study area. As the only dataset available at the time, the Çeşme dataset was used in the roof type classification study (Chapter 4). Later, the data provided by TKGM was used in other studies (Chapter 3 and 5). The results were validated both quantitatively and qualitatively, and discussed accordingly.

In addition, various solutions for the visualization have been investigated and their advantages and disadvantages could be analyzed. For this purpose, different visualization platforms such as web globes and game engines were implemented by integrating digital terrain models (DTMs), true orthophoto basemaps, 3D city plans, as well as building models with different levels of detail. Aerial photogrammetric images were also visualized in a virtual globe using data from both drones and airplane to assess the image acquisition process. Thus, the full processing chain involved in the 3D city model production, from data collection to visualization, could be assessed with a holistic approach.

1.1. Motivation

The process of creating 3D city models involves several stages, such as data collection (mostly for high-resolution optical images with stereo capability), pre-processing (photogrammetric triangulation, DSM and DTM generation, orthophoto production), feature extraction, roof and building modeling, and presentation of the models in suitable environments. The main stages (data collection, modelling, visualization) and their sub-processes are depicted in Figure 1.1. Red color in Figure 1.1 indicates the steps contributed in this thesis. This thesis aimed at improving various parts of this process such as the automatization of building model generation, investigating and increasing the efficiency of their visualization and presentation, and quality control through automated methods and visual inspection platforms.

Building models with a high LoD are often generated manually by photogrammetry operators using stereo aerial imagery. This process requires manual digitization by an experienced operator and is a costly and time-consuming process in most cases. Several commercial software reconstruct 3D city models semi-automatically, but they have certain limitations. Preparing the input data require excessive file format conversions, coordinate system transformations, and further data preprocessing. Building roofs obstructed by trees or shadows are usually reconstructed incorrectly. Software based on roof libraries are limited to widely used roof types, thus complex roofs or small objects on them such as chimneys or small windows cannot be reconstructed. Therefore, further research on the development of fully automatic algorithms for 3D reconstruction is needed.

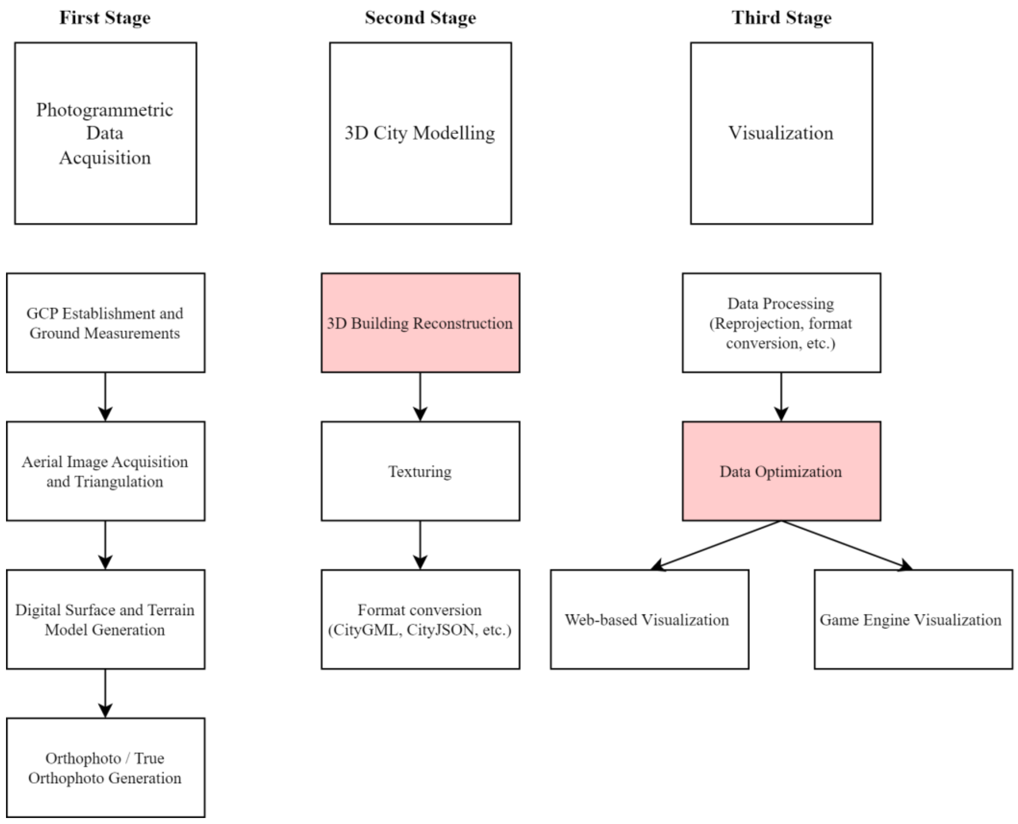

First Stage
Second Stage
Third Stage
Photogrammetric Data Acquisition
3D City Modelling
Visualization
GCP Establishment and Ground Measurements
3D Building Reconstruction
Data Processing (Reprojection, format conversion, etc.)
Aerial Image Acquisition and Triangulation
Texturing
Data Optimization
Digital Surface and Terrain Model Generation
Format conversion (CityGML, CityJSON, etc.)
Web-based Visualization
Game Engine Visualization
Orthophoto / True Orthophoto Generation

Figure 1.1 The main stages of 3D city model generation.

The DL methods have the capability of improving the performance and quality of roof segment detection and 3D building reconstruction. As a part of the thesis, a DL-based framework using line segment detection networks for extracting and vectorizing LoD 2.2 roof details at the city scale is presented. This thesis presents the first study that utilizes DL methods for extracting roof geometry in the form of vector lines at the LoD2.2. This method differs from the other studies in the literature in that it does not require building footprints and uses a single RGB image as an input while most existing methods require building footprint vectors together with RBG images for clipping images directly from building boundaries based on building footprints so that each tile contains a single building. Predicted roof structures are then vectorized, reprojected, merged, and exported at the city scale using custom scripts developed in Python. As a final step, the transition between tiles and vector geometries is corrected and redundant junctions are removed with post-processing. After the post-processing, 3D building geometries based on detected roof structures are reconstructed.

1.2. Objectives and Research Questions

This thesis has the following main objectives:

- To investigate the potential and efficiency of the DL methods for automatic 3D building reconstruction and to develop a methodology for accurate reconstruction.
- To analyze visualization methods for 3D city models with various LoDs and semantic data using different technologies such as virtual globes or game engines and provide recommendations.

Based on these goals, this thesis addresses the following questions:

- Do the DL methods provide improved results compared to conventional methods in terms of building information extraction from VHR aerial imagery?
- In what ways can the DL methods be utilized in the automatic production of 3D city models, and which advantages could they provide?
- How accurately can the DL methods classify the different roof types and with which LoDs? Is it possible to improve results by fine-tuning existing Convolutional Neural Networks (CNNs) with pre-trained weights?
- How robust are the DL methods against problems such as trees blocking roofs or shadows causing false reconstructions?
- Could the DL be used to detect complex roof structures other than the widely used gable, hip, pyramid, etc roof types?
- How can 3D city models be efficiently visualized on the web? What shall be the motivation for visualizing the models on different platforms (web, game engine) what are the requirements for obtaining high performance?

1.3. Contributions

Contributions of this thesis can be listed as following:

- A dataset for roof type classification with deep learning containing 10,000 unique roof images and their class labels was generated from very high resolution orthophotos.
- The first study on automatic vectorization of LoD2.2 roof structures with deep learning at city scale is presented.
- Python scripts developed for automatic training data generation for line segment detection networks using existing city models and VHR true orthophotos.
- The first LoD2.2 roof structure training dataset with more than 2.2 million lines and 139k buildings for line segment detection networks is presented.
- The LoD3 city model of Bizimsehir, Türkiye's first smart city project, was visualized on the web and in the Unity game engine with VR support prior to construction.
- A methodology proposed for web-based visualization of photogrammetric image acquisition flights with UAV and aircraft.

1.4. Thesis Structure

This thesis consists of the following sections:

Chapter 1 describes the motivation behind the thesis and the problems it aimed to investigate.

Chapter 2 provides a literature review on the conventional and the DL methods for building information extraction from VHR optical imagery, 3D building reconstruction, and 3D city model visualization.

Chapter 3 explores the effect of combining height information (nDSM) with RGB data for building footprint extraction, and conversion from raster to vector format.

Chapter 4 explores the potential of the DL methods for classifying roof types, and compares the results of developed shallow CNN with fine-tuned popular CNNs using pre-trained weights on the ImageNet dataset.

Chapter 5 introduces the first study for extracting and reconstructing LoD2.2 roof structures using line segment detection networks.

Chapter 6 investigates the visualization and exploration of 3D city models on the web using CesiumJS library, and also in the Unity game engine for assessing the use of Virtual Reality.

Chapter 7 provides an overview of the issues experienced throughout the thesis, analyses the results, and presents the recommendations for future work.

2. RELATED WORK

A literature review of building information extraction and 3D building reconstruction methods is presented here together with visualization approaches. The first sub-section presents the DL methods (classification, segmentation, and roof segment detection). The second sub-section gives a literature review on 3D building reconstruction. An overview of the literature on visualization methods is provided in the third chapter.

2.1. Conventional Methods

Conventional image segmentation methods are based on traditional computer vision techniques and algorithms. These methods aim to partition an image into multiple regions or segments based on certain characteristics or features of the image, such as color, texture, or intensity. One popular method is the thresholding technique [5], where an image is segmented into foreground and background based on a certain threshold value. Another common method is edge detection [6], which identifies and extracts the boundaries or edges of objects in an image. Other methods include region growing [7], where adjacent pixels with similar characteristics are grouped together, and clustering techniques, which group pixels based on their similarity in a feature space. These conventional segmentation methods are still widely used in various applications, such as medical imaging, remote sensing, and surveillance, and can often provide accurate results in simple scenarios. However, in more complex or challenging situations, such as images with low contrast or high noise, machine learning-based segmentation methods are often preferred due to their ability to learn and adapt to various image characteristics and features.

Conventional line detection methods are used to identify lines or edges in an image. These methods are based on various techniques, such as gradient-based methods, Hough transform [8], and template matching [9]. Gradient-based methods detect edges in an image by calculating the gradient magnitude and direction of each pixel and thresholding the resulting image to identify edges. The Hough transform is a popular technique that transforms an image space into a parameter space to detect lines or other shapes. Template matching involves comparing a portion of an image with a predefined template to detect lines or other shapes that match the template. Other techniques include line fitting, which fits a line to a set of points, and edge linking, which connects edges or segments to form longer lines. These conventional line detection methods have been widely used in various applications, such as computer vision, robotics, and remote sensing. However, like other conventional computer vision techniques, they have limitations in handling noisy or complex images, and may require fine-tuning of parameters to achieve optimal results. With the recent advancements in deep learning and neural networks, machine learning-based line detection methods have shown promising results and are becoming increasingly popular in various applications.

2.2. Building Information Extraction with Deep Learning

The DL studies used for the building information modelling in the literature can be categorized for classifying roof types, and extraction of building footprints and roof structures as presented in the following.

2.2.1. Roof Type Classification with CNNs

A city's structure is primarily comprised of buildings, which play an important role in many aspects. Over the past few years, simulation of 3D city models has been used across many applications [10]. Building roof types can be used for model-driven 3D reconstructions of buildings and to reduce the reliance on digital surface models (DSMs) [11].

A great deal of progress achieved in both photogrammetry and remote sensing through the use of DL [4]. Several approaches are presented in literature that relies on DL techniques to classify roof types. It is possible to classify roof types using deep CNNs as well as to estimate roof heights [12]. In their study, Partovi et al. [11] used pan-sharpened WorldView-2 imagery 50 cm Ground Sampling Distance (GSD) to classify roof types with a Visual Geometry Group (VGG-Net) model. A model-based approach has been developed by Alidoost and Arefi [13] using a combination of different data sources to enhance roof type detection using CNNs. Mohajeri et al. [14] classified six roof types for solar energy applications and achieved an overall accuracy of 66% using LiDAR data. A study by Qin et al. [15] evaluated Deep Convolutional Neutral Networks (DCNN) in building segmentation and achieved 94.67% accuracy using Gaofen-2 satellite imagery. Additionally, they stated that DCNNs have the potential to improve building mapping in dense urban areas with a wide variety of roof patterns using very high-resolution imagery.

Ölçer et al. [16] classified different roof types using a few training examples with a Siamese neural network and achieved 66% accuracy. Bittner et al. [17] investigated the use of Conditional Generative Adversarial Network (cGAN) for roof type classification using DSM derived from Worldview-1 satellite imagery. LiDAR and satellite imagery can be combined for labeling and classifying roof types using different machine learning-based methods [18]. An average accuracy of 67% was obtained for classifying rooftops using LiDAR and a random forest method by Assouline et al. [19]. ISPRS benchmark dataset for building reconstruction and classification is a widely used dataset for similar tasks that were developed by Rottensteiner et al. [20]. As part of the dataset, high-resolution aerial imagery with an 8 cm resolution and laser scanning data from an airborne laser scanner (6 points/m^2) are used to detect and reconstruct buildings, trees, and 3D models. They also provided an overview of current methods along with a discussion of the common problems of the benchmark results [21].

2.2.2. Building Footprint Extraction with Deep Learning

The CNNs are considered more effective than conventional semantic image segmentation methods in remote sensing imagery and image analysis in general [22]. As well as classifying pixels and determining the content of those pixels, these networks have also been used to predict the spatial structures of objects. CNNs are capable of detecting, segmenting, and categorizing round objects of varying sizes and shapes. Additionally, CNNs can predict the spatial extent of features including buildings, types of roofs, and other objects [23].

Usually, image segmentation is performed using two-dimensional (2D) images, as that is the most commonly used approach. The depth information, however, can also be used as an additional band on top of RGB images to provide additional information. A semantic image segmentation process that incorporates height information frequently produces better results compared to RGB-only results [24]. The majority of algorithms used for the extraction of buildings use RGB imagery as the only source of input [25]. Integrating height information with RGB imagery can help to solve weaknesses of aerial images, such as shadows, poor lighting, clouds, and many other obstructions.

A study by Marmanis et al. [22] shows that the CNNs can utilize high-resolution aerial images to segment them and explicitly represent the boundaries between different classes. Using the ISPRS Vaihingen benchmark dataset as the benchmark, their DCNN model was able to achieve a 95.2% F1-score for building segmentation. The DeepResU-Net was developed by Yi et al. [26] for Very High Resolution (VHR) imagery to be utilized for pixel-based building extraction. As compared to the U-Net, network performance and overall accuracy increased significantly.

A study by Kada and Kuramin [27] utilized the PointNet++ and KPConv algorithms to classify building roofs from LiDAR data and scored an IoU of 94.8%. Jiwani et al. [28] used a modified DeepLabV3+ for extracting building footprints from satellite images. It was demonstrated in the test with the help of three building extraction benchmark datasets that their method achieved state-of-the-art results regardless of image resolution and building density. The different CNNs can efficiently be combined to extract building footprints based on VHR aerial imagery, as shown by Li et al. [29]. Pixel-based segmentation accuracy of the model is measured by comparing each overlapping pixel, and the precision, recall, and confidence of the model are 92.6%, 91.4%, and 85.1%, respectively, for the WHU building dataset. WHU dataset consists of aerial imagery dataset and satellite imagery dataset with varying resolutions from 0.075m to 2.5m with manually delineated building footprint vectors, and used as benchmark dataset in many building extraction studies.

Building extraction can also be achieved by combining data from multiple sources and using CNNs. Additional data sources can be added as a fourth band to RGB images without making any modifications to the CNN model. Sun et al. [30] used a frame field learning model for building footprint extraction from orthophotos and nDSM with 0.25m GSD. It was observed that incorporating the nDSM has made an improvement of 12% to the intersection over union(IoU) value, as compared to the 58% gained from only using RGB images.

Bittner et al. [31], used fully convolutional networks (FCN) for building segmentation using spectral and height data collected by multiple sensors and achieved 85.5% accuracy. Zhao et al. [32] developed a methodology using CNNs and recurrent neural networks (RNNs) for generating regularized building outlines in vector format, where a CNN was used to extract image features, and RNN was used to extract building polygons to generate regularized building outlines. Following the work of PolyMapper [25], the researchers have made several improvements to the backbone, as well as improved the detection, and recurrence modules. It has also been shown that deep learning can be combined with guided filtering for estimating district boundaries by Xu et al. [33].

2.2.3. Roof Structure Extraction with Deep Learning

Building outlines and roof line structures can be extracted from remote sensing imagery with DL methods. Conventional methods (non-DL) also can be used to extract line segments from aerial imagery in urban areas [34]. The neural networks are not only capable of detecting edges in images, but also of assembling them into graphs. Several studies found in the literature use conventional and DL-based methods to detect roof lines and reconstruct 3D models. An extensive review of DL-based 3D building reconstruction methods are presented by Buyukdemircioglu et al. [35].

Using very high-resolution orthophotos of the city of Detmold, Hensel et al. [36] vectorized building roof structures using Point-Pair Graph Network (PPGNet). The F1 scores for junction detection and edge detection for LoD2.0 roof structures were 0.93 and 0.87, respectively.

Conv-MPN [37] was proposed as a DL architecture for reconstructing roof structures as a planar graph from remote sensing imagery. Conv-MPN is a two-stage method that requires corner detection as the first stage, then network training as the second stage. As well as being computationally expensive, multi-stage approaches are inefficient for both training and inference.

Roof Structure Graph Neural Network (RSGNN) [38] is another one-stage graph neural network for extracting LoD2.0 roof structures from satellite and aerial imagery. Their method achieved state-of-the-art results for extracting roof structures from VHR images. Additionally, they introduced using Hough transform modules to improve line feature detection using geometric priors. The Deep Roof Refiner [39] is another deep learning method for extracting roof structures. In quantitative and qualitative experiments, they achieved an optimal F1-score of 60.89% and 63.48%, respectively.

Gui and Qin (2021) proposed a DL-based LoD2 building reconstruction using MVS satellite images. Using a "decomposition-optimization-fitting" paradigm, they reconstructed LoD2.0 building models based on a model-driven approach. Since the roof models are reconstructed based on a roof type library, it may be challenging to obtain reliable predictions for complex roof structures with the proposed method.

In a study by Alidoost et al. [40], 3D roof structures were extracted from aerial imagery using a Y-shaped convolutional neural network. This framework consists of a Y-shaped CNN with two encoders and one decoder. The proposed CNN computes predicted heights and rooflines for three classes of eaves, ridges, and hips in LoD2.0 from RGB imagery. Kenzhebay [41] proposed a method for roof structure extraction from aerial imagery and

DSMs and FCNs. Muftah et al. [42] used CNNs for extracting LoD2 roof structures from aerial imagery.

2.3. 3D Building Reconstruction

Here, related work on building reconstruction methods is presented based on conventional and novel machine learning methods. Special emphasis was also given to the DL methods and the combinations of different approaches due to their popularity.

2.3.1. Conventional Methods

3D building reconstruction and 3D city modeling are mostly performed using conventional methods. Detailed reviews of conventional 3D building reconstruction methods as well as their applications are available in the literature [43-45]. The subsurface growing method is an example of a conventional technique that can be used for reconstructing 3D buildings [46]. Polyfit [47] is a data-driven software that reconstructs lightweight polygonal surfaces using point clouds. Photogrammetric point clouds are also used for reconstructing 3D building models by combining RANSAC and contextual knowledge [48]. Digital surface models (DSM) and 2D footprints can be combined to automatically create LoD1 building models [49]. It has also been demonstrated that model-driven reconstruction methods can be used to semi-automatically reconstruct 3D city models from large-format aerial imagery as illustrated in Figure 2.1 [50].

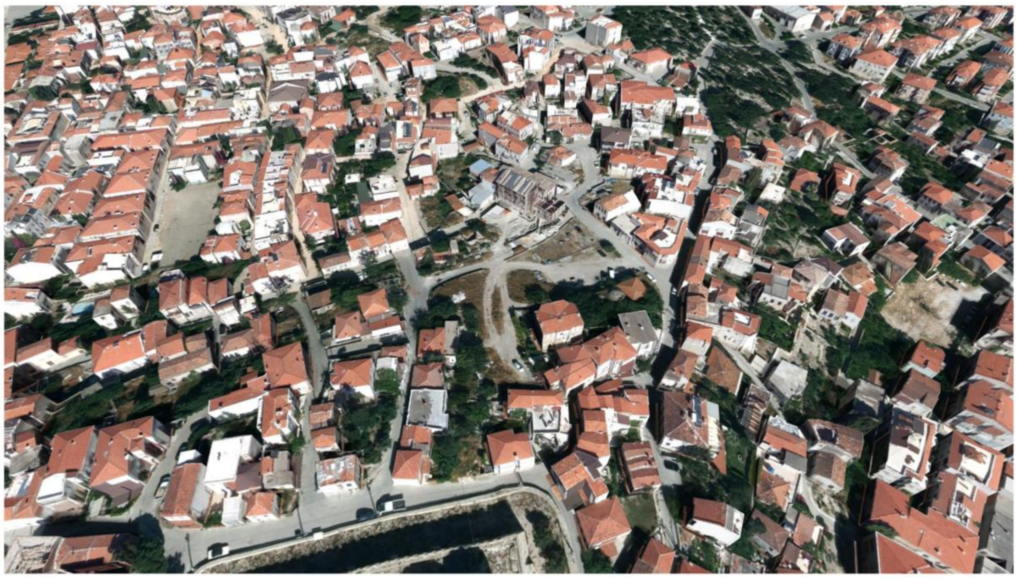


Figure 2.1 A view of the semi-automatically reconstructed 3D city model of Cesme, Türkiye [50].

Reconstruction of 3D buildings can also be performed with rule-based methods and photogrammetric point clouds [51]. A common and fast technique for 3D building reconstruction models is building footprint extrusion. It is also possible to perform LoD2 building reconstruction using the half-spaces method [52]. Drešček et al. [53] presented a methodology based on a method known as Extract, Transform, Load (ETL) for 3D building reconstruction from photogrammetric point clouds. Murtiyoso et al. [54] developed a data-driven framework for LoD2 building reconstruction using photogrammetric point clouds.

Using RANSAC constraints and topological-relation constraints, Li and Wu [55] reconstructed 3D complex buildings using incomplete point clouds. Li and Shan [56] proposed a two-step RANSAC-based method 3D building reconstruction method using both LiDAR and photogrammetric point clouds. Using 2D building footprints and Airborne Laser Scanning (ALS) point clouds, an automatic algorithm was developed to reconstruct ten million LoD2 buildings in the Netherlands [57]. There is still a challenge for researchers in the area of automatic reconstruction of LoD3 building models. It has been shown that LoD3 building models are usually digitized manually and that they can be merged and visualized together with existing 3D city models [58].

2.3.2. Machine Learning Methods

Apart from the neural networks, other machine learning approaches can also be used to reconstruct 3D buildings and 3D city models. Using context-free and weighted attributes, Dehbi et al. [59] developed a method for reconstructing 3D buildings based on context-free grammar rules using Markov Logic Networks. Reconstruction of 3D building models without elevation data is also possible with the help of machine learning [60]. According to the proposed method, a building's height is predicted by analyzing the footprints and attributes associated with the building, and then the footprints are extruded into 3D models using the predicted height. It is possible to produce LoD2 models using datasets with lower LoDs by predicting the roof types using machine learning methods [61]. As a result of using multiclass classification, they predicted the type of roof with an accuracy of 85% and predict whether the roof was flat with an accuracy of 92%. Park and Guldmann [62] reconstructed a 3D city model using LiDAR point clouds using a Random Forest-based point cloud classification methodology. Multi-temporal (4D) city models can also be reconstructed by combining machine learning methods with historical information [63].

2.3.3. DL-Based Reconstruction

Conventional 3D building reconstruction methods mainly involve two major problems [64]. First, due to the number of manual designs involved in them, they are prone to errors. Additionally, they are incapable of learning semantic features associated with 3D shapes. Also, a large part of the effectiveness of this method relies on image quality and camera calibration. By leveraging deep neural networks to automatically learn 3D shapes from earth observation data, DL methods can resolve these deficiencies.

Several DL-based methods can be found in the literature for 3D building reconstruction. The DL models are extremely powerful in many computer vision tasks by using images to learn features [65]. It is also possible to reconstruct 3D buildings from EO data using these methods. To perform a parametric 3D building reconstruction using satellite imagery, Wang and Frahm [66] proposed a DL-based solution on the parametrization of buildings as 3D cuboids. It has also been shown that CNNs can be used to reconstruct buildings with procedural modeling. By inferring shape grammar rules from sequences of 3D points, CAD-quality models were generated with Neural Procedural Reconstruction [67]. Nishida et al. [68] developed a CNN-based tool for automatically generating 3D building models from remote sensing imagery. Alidoost et al. [10] used CNNs for building detection and reconstruction using aerial imagery. Based on the proposed method, they achieved root mean square errors (RMSEs) of 3.43 m for 3D building reconstruction and 1.13 m for nDSM. Multiple CNNs with encoder-decoder architecture were used by Agoub et al. [69] to create 3D city models with depth maps. Figure 2.2 provides an overview of 3D city model reconstruction based on their approach.

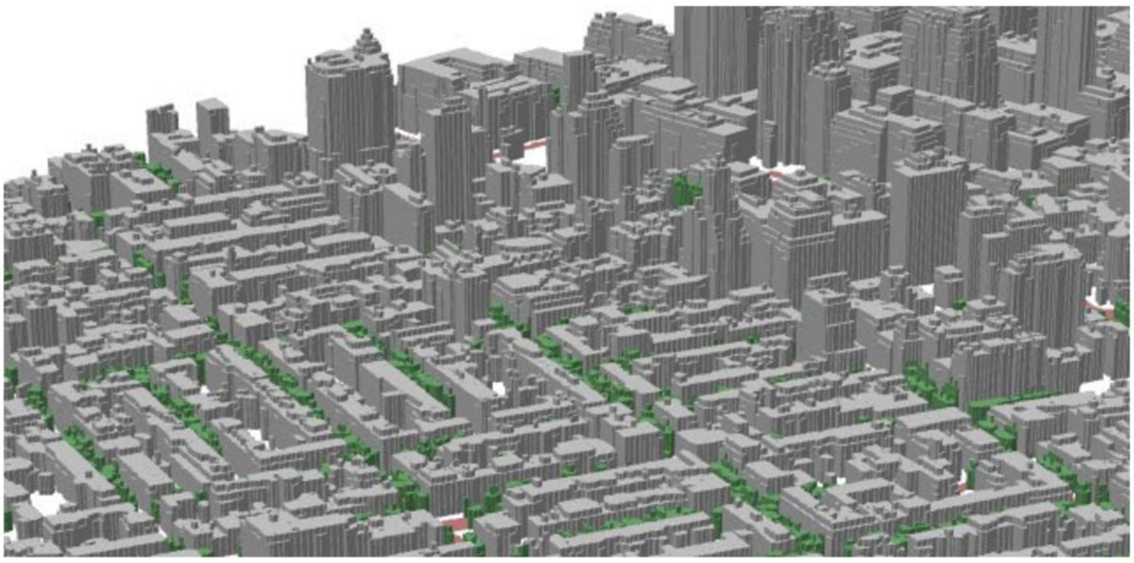


Figure 2.2 CNN-based 3D city model reconstruction of the Manhattan area [69]

Knyaz et al. (2020) presented another example of the use of CNN in a grid structure. It has been demonstrated that CNN is an effective method of automatically segmenting wire structures based on semantics, which overcomes the limitations inherent in photogrammetric processing when applied to reconstructed complex grid structures in three dimensions.

Generative Adversarial Networks (GANs) [70] also are capable of generating 3D building models. There are two main parts of a GAN, a generator, and a discriminator. To produce photorealistic images and fool the discriminator, generators learn the distribution of real images. Discriminators judge whether generated images are real or fake. GAN-based methods also can be used to identify 3D shapes with obscured or missing portions [64].

3D buildings can be reconstructed from noisy DSMs using Conditional GAN (cGAN) [71]. A 3D surface model of LoD2 was created by Bittner et al. [72] using stereo satellite imagery with 50 cm GSD. Bittner et al. [17] also produced digital surface models (DSMs) that provided high levels of detail similar to the LoD2 building forms but also assigned additional object class labels on every pixel. It is also possible to use GANs to automatically reconstruct buildings in LoD1 [73]. FrankenGAN [74] is another network for reconstructing and enriching 3D city models with geometric details and building textures. Roof-GAN [75] uses a combination of primitive roofs for generating 3D roof geometries. Figure 2.3 illustrates a view of 3D roofs with a different number of primitives reconstructed by Roof-GAN.

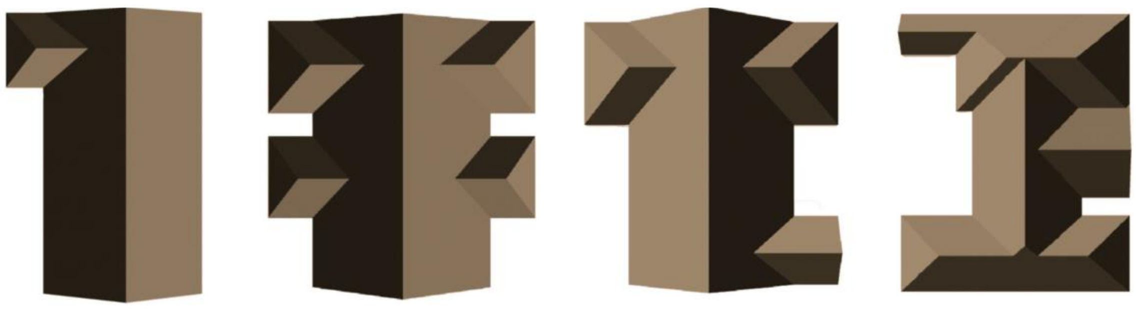


Figure 2.3 A view of roofs reconstructed by Roof-GAN [75]

2.3.4. Combination of Deep Learning Based and Conventional Methods

The DL methods can also be used to classify and reconstruct buildings from LiDAR point clouds [76, 77]. A DL-based segmentation was used for 3D city modelling using satellite imagery as mesh 3D models with textured surfaces by Leoatta et al. [78]. DL-based methods can be used with point clouds for the automatic estimation of building roof shapes in complex and noisy scenes [79].

A DL-based 3D reconstruction framework was introduced by Yu et al. [80] that automatically creates LoD1 building models using stereo aerial imagery. It has been demonstrated by Gui and Qin [81] that deep learning can be used to reconstruct the LoD2 building model's MVS aerial imagery. There are several steps in the developed workflow, such as detecting building segments on the instance level, extracting initial building polygons, decomposing and refining the building polygons, fitting the basic model, and merging the models. DL-based methods are also capable of reconstructing historical 3D city models [82]. Partovi et al. [83] developed a DL-based workflow for automating 3D building reconstruction. Their method consists of building footprint extraction, building decomposition, roof type classification, and the calculation of 3D roof structure parameters.

Teo [84] used FCNs for detecting building regions from laser scanning data and 3D prismatic building model reconstruction. An automatic reconstruction method for building models is developed by Kippers et al. [85] using the combination of CityJSON and building footprints. In their paper, Yu et al. [86] proposed a DL-based method for automatically reconstructing LoD1 building models. Their method includes three steps: DSM generation, building boundary detection, and 3D building reconstruction. Zhang et al. [87] developed a framework for 3D building reconstruction using PointNet++ and a holistic primitive fitting method. Chen et al. [88] developed a three-step method, which makes use of embedded implicit fields and point clouds for 3D building reconstruction. Moreover, DL-based methods can be combined with geographic information systems (GIS) and satellite imagery to reconstruct 3D city models [89].

2.4. Visualization of 3D City Models

In this section, 3D city model visualization methods were discussed under the web- and game engine-based technologies. It is also possible to use other desktop software including those from the geographic information system (GIS) software vendors, which are not considered here.

2.4.1. Web-based Visualization

Although various platforms exist for visualizing 3D city models, the most common use is web-based visualization. The main reason for this is that it allows quick use via the web browser without requiring any extra software installation. 3D city models of different cities and countries are presented online and can be accessed by users. Web-based visualization of 3D city models can be used in different applications with the help of semantic data. Urban building energy modeling [90], Building Information Modeling (BIM) [91], Heating Demand Prediction [92], air quality information [93], flood simulation [94], smart city applications [95], cultural heritage [96] can be given as example to different usages of 3D city models.

Stakeholders and citizens can use virtual 3D city models as part of collaborative processes with their cities to help improve their quality of life [97]. 3D city models also allow different analyses with the help of their semantic data. Visibility analysis is one of the most common analyzes used on these models [98]. Although the main element of 3D city models is buildings, other city objects are also of great importance in visualization and analysis. Visualizing city furniture, bridges, tunnels, vegetation, etc. are also can be visualized with building models [99]. Building models can be integrated with architectural plans and cadastral data for more detailed analysis [100].

Virtual globes have become very popular and widely used in many applications. WebGL technology made it possible to visualize and explore maps in 3D. Using WebGL requires no additional plugins or extensions and enables cross-platform flexibility. Even with very large datasets, it provides high performance with the help of GPU and WebGL technology. CesiumJS [101] is a 3D geospatial data visualization library for both web and game engines. As part of the streaming performance enhancements provided by CesiumJS, the datasets are rendered using WebGL (Web Graphics Library). Different types of geospatial data are supported by CesiumJS, including 3D city models, terrain, imagery, and point clouds.

CesiumJS uses 3D Tiles [102], an Open Geospatial Consortium (OGC) standard format for rendering and streaming 2D/3D geospatial datasets including 3D city models, photogrammetric models, Building Infrmation Modeling(BIM)/Computer Aided Design(CAD) models, and point clouds. Loading large volumes of geospatial data or 3D city models on the Cesium virtual globe as a single tile usually is not recommended for performance reasons. Thus, tiling large volumes of data is the best solution for this problem. By using Adaptive Quadtree Tiling, 3D Tiles loads huge datasets as smaller parts and renders them by dividing them into tiles, efficiently and effectively. By tiling, stream performance can be improved and the browser's hardware requirements can be reduced. In 3D Tiles, performance can be created for many zoom levels in the same view using a geometric error to select detail levels and an adjustable pixel defect. In 3DTiles, 3D geometries and models are stored in glTF [103] format, which is widely used across a variety of applications that deal with 3D geometries and models. It is possible to store, stream, and optimize geographical data using Cesium ION [104], which is a cloud-based platform that optimizes, tiles, and serves 3D geodata such as images, terrains, buildings, point clouds, BIM/CAD, photogrammetry, and many other types of geospatial data based on CesiumJS.

2.4.2. Game Engines, Virtual Reality, Augmented Reality and Mixed Reality

The large size of the geometries and textures within 3D city models requires performance optimization for visualization. It is possible to integrate 3D city models into game engines to visualize them more realistically. By supporting features such as high-detailed photogrammetry models, terrain models, basemaps, and 3D buildings, game engines can visualize high-detailed 3D geospatial datasets.

The popularity of virtual reality (VR) can be attributed to its use in many fields, but it is most famous for its use in computer games. Game engines with VR technology offer many benefits, such as the ability to explore 3D city models at the street level or to better visualize future cities by combining them with existing 3D city models. Virtual reality is an effective tool for evaluating the impact of future cities on the environment and infrastructure. VR can enhance the planning and design process by allowing stakeholders from different disciplines to participate in the process.

3D virtual representations of landforms can be created and visualized with the Unity game engine [105]. The Unreal engine is capable of visualizing large-scale photogrammetric models [106]. Game engines are also used for visualizing and disseminating cultural heritage [107]. Modeling and texturing procedures are provided within their pipeline for converting point clouds into textured models to import into game engines.

Virtual reality technology is also capable of allowing users to explore museums in a new way [108]. Virtual reality is more than just a static virtual environment, it can also be used as a powerful tool for combining various data types in a single scene [109]. Kim and Kim [110] have used eye-tracking experiments to study perception and cognitive processes in VR simulations. As part of their research, Broucke and Deligiannis [111] evaluated perceived workloads and the parameters associated with data immersion by analyzing tasks of data exploration on different web interfaces and the proposed VR application. Historical cities can be explored interactively with VR technology [109]. Game engines can also be used for the visualization and monitoring of smart cities in a virtual reality environment [112].

Geospatial data can be visualized and interacted with using a variety of platforms and technologies, including VR [113], AR [114], and mixed reality [115]. Several areas in which AR technologies are being used may be able to improve citizen-authority engagement, such as urban decision-making and stakeholder participation [116]. Liu et al. [117] conducted an outdoor case study with an AR system to detect thermal targets in façade inspection tasks. Based on VR/AR environments, Santana et al. [118] developed a mobile visualization application to display simulation and modeling results at the building level.

3. DEEP LEARNING BASED BUILDING FOOTPRINT EXTRACTION WITH FUSION OF TRUE ORTHOPHOTOS AND ELEVATION INFORMATION

At this stage of the study, instead of classifying roof types as explained in the previous chapter, building roofprints were aimed to reconstruct. Therefore, a framework for the building footprint extraction using CNNs was implemented and experimented using VHR true orthophotos and nDSM of Selcuk town in Izmir Province, Turkey. Unet and LinkNet networks with different backbones were used on two different datasets, i.e. RGB (image only) and RGB-Z (image + elevation information), and their quantitative and qualitative results are discussed. The results presented in this chapter were largely published in [119].

3.1. Motivation for Building Footprint Extraction

Pixel-based classification and semantic segmentation of remotely sensed imagery can be used to obtain information for several tasks, such as mapping and analysis of land cover or the object detection. A major challenge in semantic image segmentation is the continuous increase in resolutions of remotely sensed imagery. The amount of detail contained in very high-resolution aerial images is making DL-based approaches for extracting buildings more challenging. Higher image resolution results in wider class imbalances and increased levels of difference for all classes, even though the VHR is capable of collecting small details. A variety of conventional image segmentation methods are still in use today, including thresholding, region-growing, and edge-based methods, but they have some limitations. They are sensitive to noise and these methods may not adapt well to changes in the image data or to different imaging modalities.

There have been significant improvements in the performance of CNNs over conventional methods in the last decade. As a result, DL-based segmentation and classification methods are becoming more popular and widely used by many researchers.

This chapter examines the performance of two CNN models for building segmentation, namely U-Net [120] and LinkNet [121]. The results of these two CNNs are compared using a set of different backbones. The dataset includes true orthophotos and nDSM generated from VHR stereo aerial imagery. A comparative evaluation of the implemented methods was conducted using first RGB data only, and then RGB + nDSM data. It was found that fusing height information with RGB data improved model accuracy, thereby improving their performance. Following the building extraction, the segmented buildings were converted into vector geometry from pixels using GDAL [122], which was then simplified to improve their appearance by smoothing with Douglas-Peucker [123] simplification method. Vectorization result measures are presented and discussed in detail in the last sub-section.

3.2. Study Area and Dataset

The production and development of 3D city models are common in countries around the world, but they are also an important and active topic in Türkiye as well. General Directorate of Land Registry and Cadastre of Türkiye (GDLRC) is started "Production of 3D City Models and Creation of 3D Cadastre Bases" project in 2018. The project is to include all provinces and districts of Türkiye's settlement and development regions, it is planned to produce 3D models of about 11 million buildings in these regions. This project has been carried out in cooperation with GDLRC and the private companies. These companies are responsible for digitizing building geometries and roof models in CityGML LoD2.3. This study was conducted using data produced and provided by the GDLRC within the scope of this project.

Experiments were conducted in a field of approximately 4.12 km^2 with 13,269 buildings in Selcuk, Izmir, Turkey (Figure 3.1). A total of four types of data were included in the dataset, involving true orthophotos (RGB), raster DSMs, and DTMs with 0.1 m GSD, along with vector building footprints. A ground filter or similar method can be used to remove man-made objects from DSM to generate DTM if one has not already been generated for the study area. A number of different ratios were used for the number of training, validation, and test images, and the ratio that yielded the best results was selected. 80% of the dataset is used for training deep learning models, 10% for validation, and 10% to test the performance of the models, respectively. In total, 2,185 buildings were included in the test area, including buildings with different roof types and structures. A random sample of validation data was selected from the study area. The buildings in the test area were excluded and not used as part of the training deep learning models.

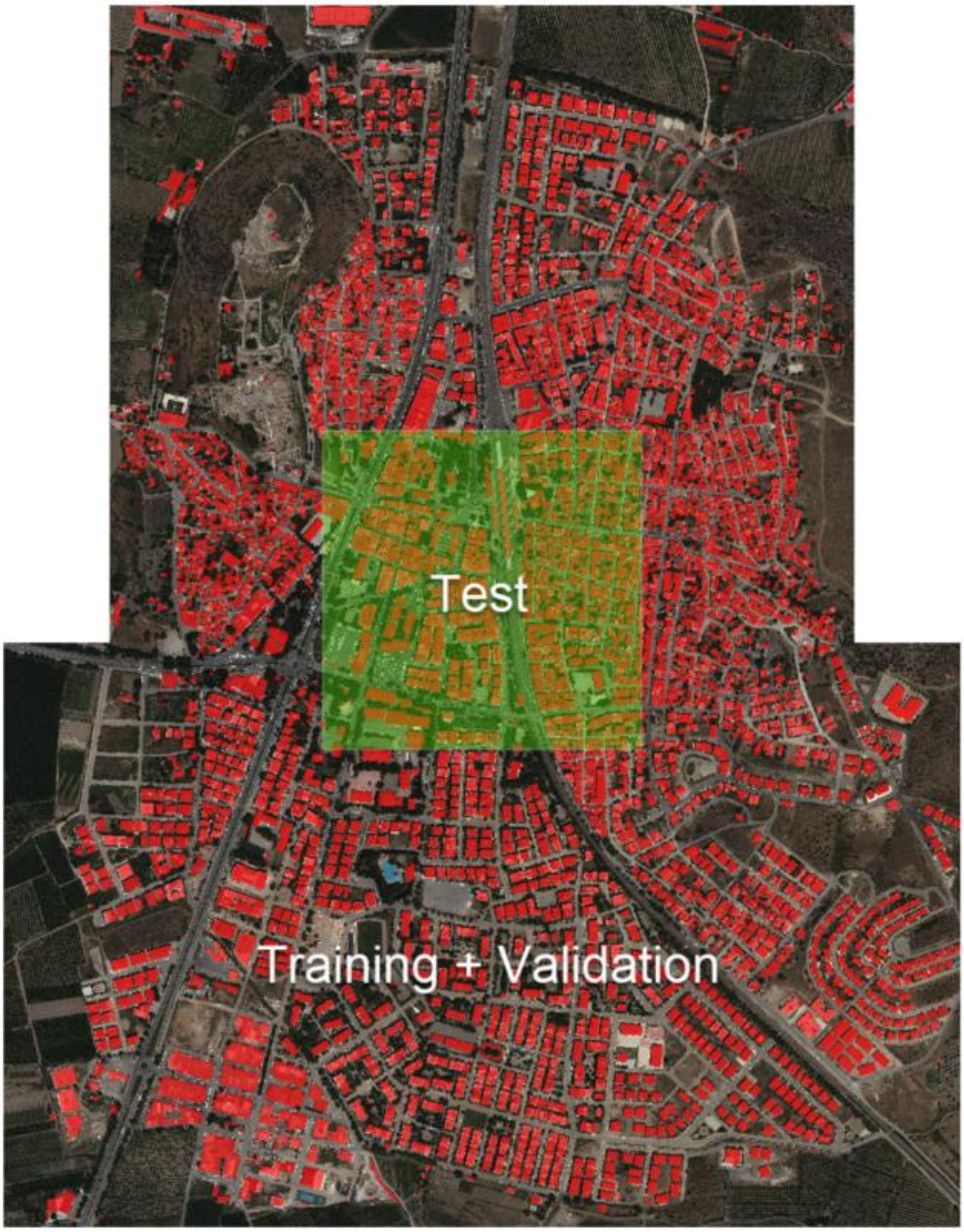


Figure 3.1 An overview of the study area and building footprints

Several pre-processing operations are performed for generating input image tiles for CNNs. Raster image of the building footprints was generated using building footprint vectors, then the ground truth masks are generated by assigning pixels within the building given the value of "1" and those outside given the value of "0". The study area was clipped into 256x256 grids of non-overlapping pixels from the raster data, and those grids were then used as inputs to the deep learning models. An example tile illustrating the RGB true orthophoto, vector building footprint, nDSM, and ground truth mask can be seen in Figure 3.2.

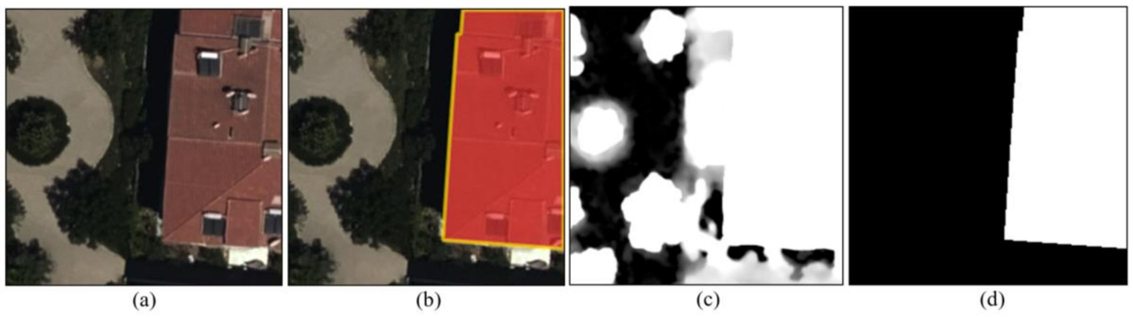


Figure 3.2 A sample tile from the study area: (a) RGB true orthophoto, (b) building footprint vector, (c), nDSM, and (d) building mask [119]

3.3. Model Training

Several network training processes were conducted using UNet and Link-Net with different backbones to obtain the most accurate possible results. To determine whether building height information contributes significantly to the results of networks, two separate inputs (RGB and RGB + nDSM) for networks were used. Several backbone networks (ResNet-18, ResNet-50, and SeResNet-18) were used with U-Net and LinkNet, both of which have a good reputation for their success when used for segmentation, to achieve different levels of success in segmentation. Each CNN was trained using the generated training dataset, i.e., a pre-trained weight or fine-tuning of the weights is not used during the training phase. The learning process includes tweaking a few parameters that are critical to the success of the process, such as the initial learning rate, batch size, number of epochs, loss function and optimization method. An overall view of the developed framework is given in Figure 3.3. An overview of model training parameters is given in Table 3.1.

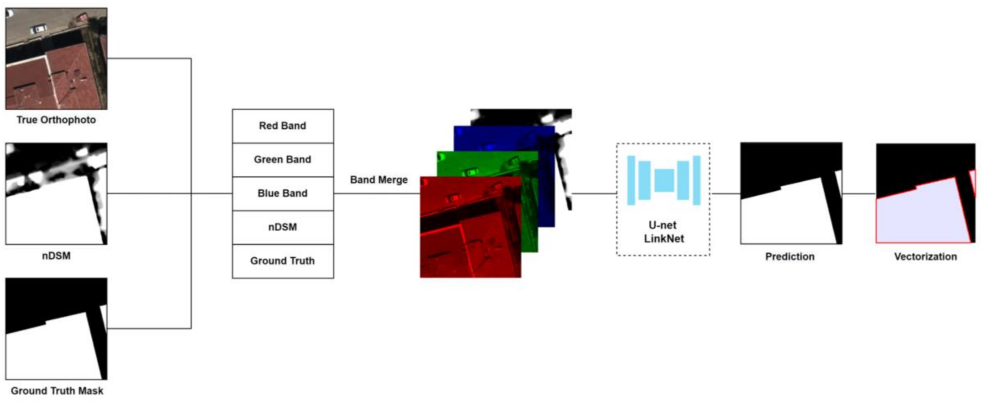
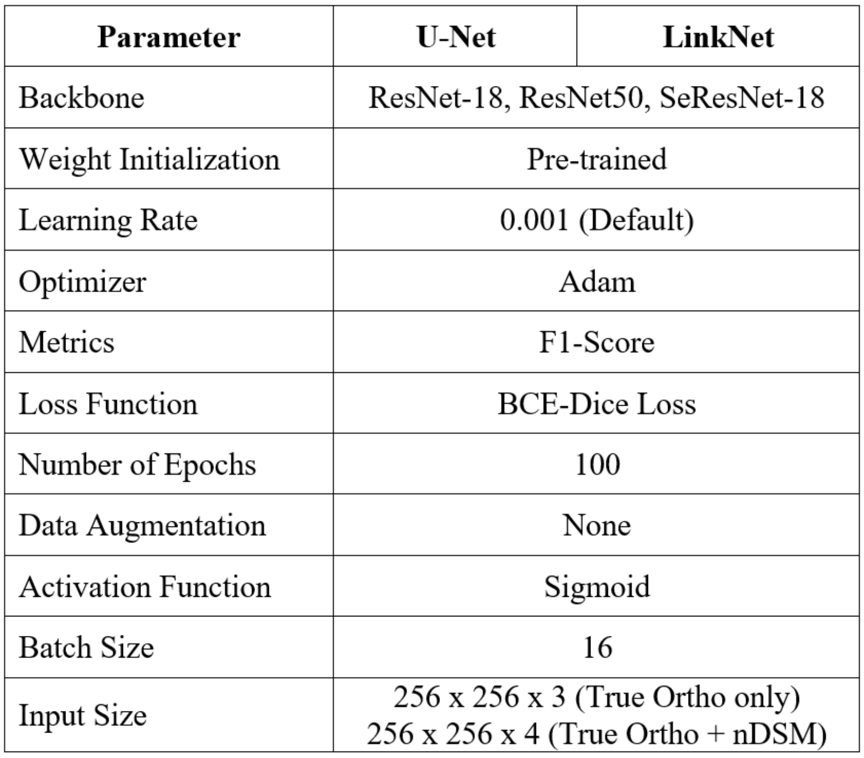


Figure 3.3 An overall view of the developed framework [119]



Parameter	U-Net	LinkNet
Backbone	ResNet-18, ResNet50, SeResNet-18	
Weight Initialization	Pre-trained	
Learning Rate	0.001 (Default)	
Optimizer	Adam	
Metrics	F1-Score	
Loss Function	BCE-Dice Loss	
Number of Epochs	100	
Data Augmentation	None	
Activation Function	Sigmoid	
Batch Size	16	
Input Size	256 x 256 x 3 (True Ortho only) 256 x 256 x 4 (True Ortho + nDSM)	

Table 3.1 Model training parameters [119]

Model training was performed using the Adam optimizer. The main difference between different optimization algorithms is the way the learning rate is implemented as well as the frequency with which these parameters (weights) are updated. The training was performed 0.001 learning rate and weight decay is not used during model training.

An epoch of training refers to a complete cycle through all of the datasets for training. Several model trainings were performed with different epochs as part of the model training process to decide the most suitable number of epochs for training without overfitting. Each CNN is trained for 100 epochs, as many models' highest accuracy is achieved up to 100 epochs, and training with more epochs did not result in further improvements in the results. Each time the model was run through an epoch, the validation data were used to evaluate the model for that epoch.

Batch size is also an important parameter that determines how many images will be used for training in each epoch. Usually larger batch sizes lead to better results. Taking this into consideration, a batch size of 16 is used for each model training. Models performance is assessed using the F1-score, and Binary cross entropy (BCE)-dice is used as the loss function. There are several ways to measure the error rate. One is by using the F1-score, a harmonic mean that gives an estimate of how many incorrect classifications were made. Segmentation is typically carried out using the BCE-Dice loss function. Both approaches are useful for various reasons, such as being able to maintain the stability of BCE while still allowing for some diversity in the loss.

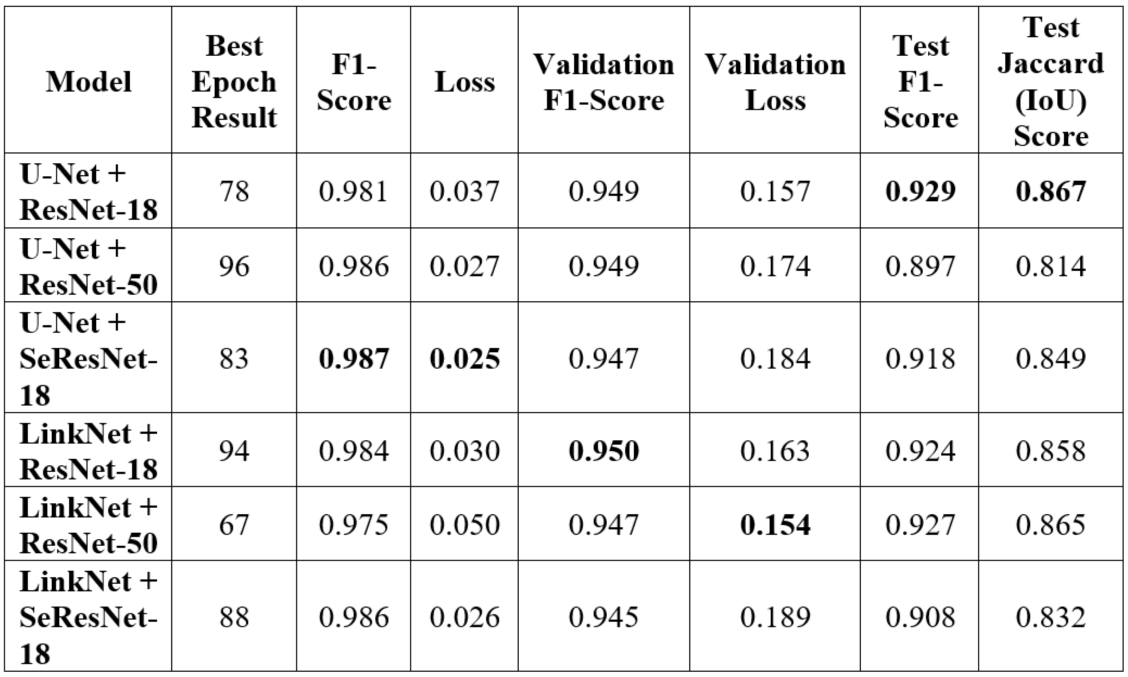
Data augmentation techniques can also be applied to prevent models from overfitting. Images can be augmented by performing geometric operations, such as flipping, scaling, and rotating. Since no overfitting occurred during the training phase, this study did not use any data augmentation techniques. It is also common to employ pre-trained models trained on large datasets by fine-tuning pre-trained weights. This experiment does not use pre-trained weights or fine-tuned networks but instead trains all networks from scratch using the generated training dataset.

3.4. Results

In this section, RGB-only results and RGB-Z results are discussed. Results and comparisons of the vectorization process are also presented.

3.4.1. True Orthophoto (RGB) Results

The results of the model based on the use of only true orthophotos are shown in Table 3.2. U-Net with ResNet-18 backbone achieved the highest F-1 Score and IoU score on the test data with 92.8% and 86.7%, respectively. LinkNet with ResNet-50 backbone, which achieved the second best performance with an F-1 score of 92.7% and IoU score of 86.5% resulted as second. Results show that U-Net and LinkNet are of performed similar in terms of visual analysis. There are, however, some differences between the segmented output of U-Net and LinkNet, with LinkNet usually containing unorganized predictions that are less homogeneous. In addition to this, the U-Net has shown to be an effective means of separating structures with small areas, even though it is more challenging. Nearly all models failed to predict correctly when the rooftops were covered with trees. The quality of segmentation was primarily affected by shadows for all models. A shaded area belonging to the building class is frequently misclassified as a non-building area. Using pre-trained weights rather than fine-tuned CNNs has the potential to be a useful strategy to improve the segmentation performance. A comparison of the predictions of all the models is shown in Figure 3.4 on the same image tile.



Model	Best Epoch Result	F1-Score	Loss	Validation F1-Score	Validation Loss	Test F1-Score	Test Jaccard (IoU) Score
U-Net + ResNet-18	78	0.981	0.037	0.949	0.157	**0.929**	**0.867**
U-Net + ResNet-50	96	0.986	0.027	0.949	0.174	0.897	0.814
U-Net + SeResNet-18	83	**0.987**	**0.025**	0.947	0.184	0.918	0.849
LinkNet + ResNet-18	94	0.984	0.030	**0.950**	0.163	0.924	0.858
LinkNet + ResNet-50	67	0.975	0.050	0.947	**0.154**	0.927	0.865
LinkNet + SeResNet-18	88	0.986	0.026	0.945	0.189	0.908	0.832

Table 3.2 Quantitative results of RGB-only training [119]

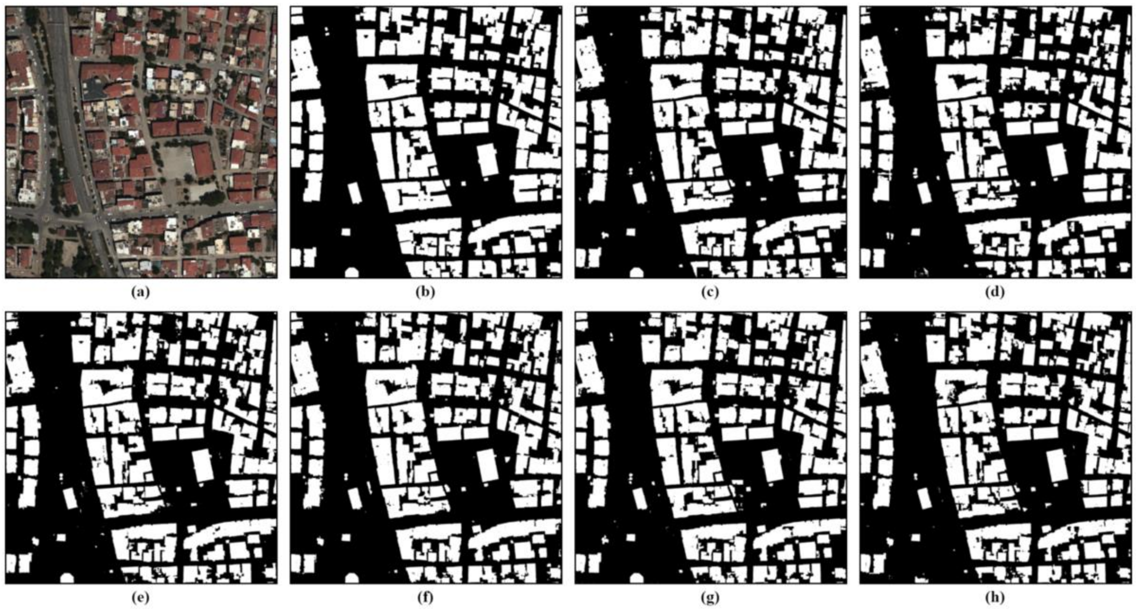


Figure 3.4 An overview of the RGB-only predictions for the test area: (a) True orthophoto, (b) Ground truth, (c) U-Net+ResNet-18, (d) U-Net+ResNet-50, (e) U-Net+SeResNet-18, (f) LinkNet+ResNet-18, (g) LinkNet+ResNet-50 and (h) LinkNet+SeResnet-18 [119].

3.4.2. True Orthophoto + nDSM (RGB-Z) Results

By incorporating height information into the training process, F1 and IoU scores significantly increased when compared to RGB-only results. LinkNet with ResNet-50 backbone achieved the best F1-Score and IoU score on test data with 96.1% and 92.6%, respectively. As compared to RGB-only results, using nDSM for the fourth band improved the F1-Score and IoU scores by 3.2% and 5.9%, respectively. The validation loss, which was previously 0.154, was reduced to 0.073 as a result of this improvement. Visual inspection of test predictions also indicates that incorporating height information increased the performance of all CNNs. (Figure 3.5). This can be seen in the fact that the outputs were smoother, homogeneous, and more structured. Furthermore, adding nDSM as height information also resulted in a less fuzzy presentation of boundaries along buildings as well. Moreover, it has been observed that the U-Net was able to provide a better segmentation quality for complex buildings, as well. A comparison of the performance measures of different CNNs for RGB +nDSM is presented in Table 3.3.

Model	Best Epoch Result	F1-Score	Loss	Validation F1-Score	Validation Loss	Test F1-Score	Test Jaccard (IoU) Score
U-Net + ResNet-18	93	0.982	0.034	0.972	**0.073**	0.958	0.919
U-Net + ResNet-50	95	**0.987**	**0.023**	**0.973**	0.086	0.960	0.924
U-Net + SeResNet-18	84	0.983	0.032	0.972	0.080	0.958	0.920
LinkNet + ResNet-18	93	0.985	0.028	0.972	0.086	0.958	0.919
LinkNet + ResNet-50	89	0.986	0.027	0.973	0.079	**0.961**	**0.926**
LinkNet + SeResNet-18	95	0.987	0.025	0.971	0.093	0.960	0.925

Table 3.3 Quantitative results of RGB + nDSM training [119]

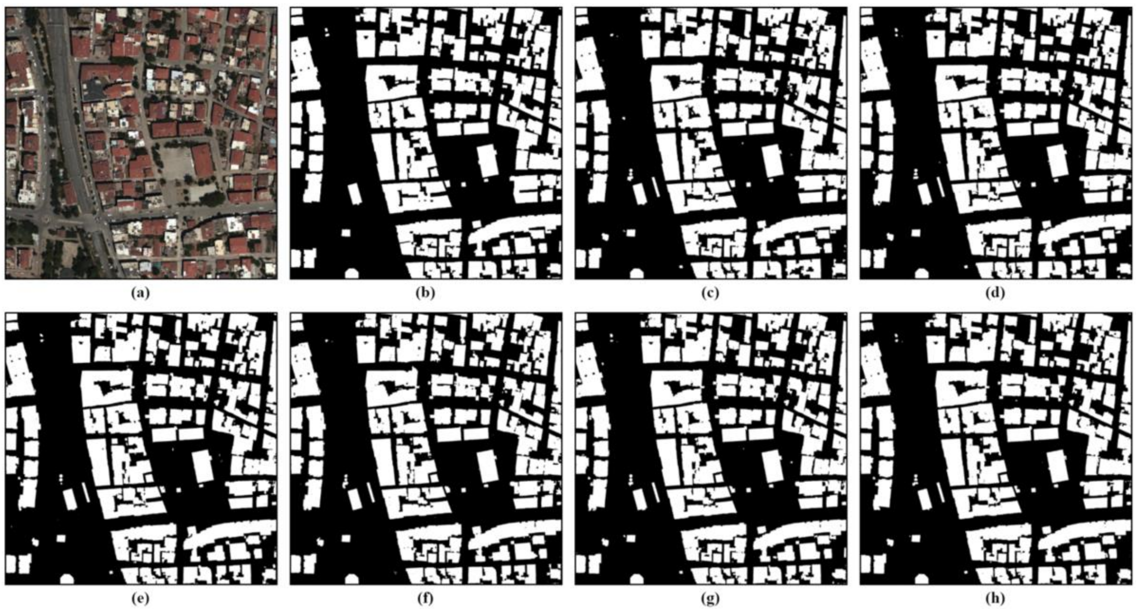


Figure 3.5 An overview of the RGB + nDSM predictions for the test area: (a) True orthophoto, (b) Ground truth, (c) U-Net+ResNet-18, (d) U-Net+ResNet-50, (e) U-Net+SeResNet-18, (f) LinkNet+ResNet-18, (g) LinkNet+ResNet-50 and (h) LinkNet+SeResnet-18 [119].

3.5. Vectorization

As part of the vectorization, predicted building boundaries were vectorized with the help of GDAL. The Douglas-Peucker line simplification algorithm was applied to vectorized building footprints to reduce the total number of lines and make lines smoother. In the Douglas-Peucker algorithm, a vector dataset is simplified by a tolerance value, so less than the given tolerance value will be straightened out and simplified. Several tolerance values are used to minimize the mean difference of vector geometries. Used tolerance values for different models and their mean differences are given in Table 3.4. A view of the generated vectorized building footprints from the test area can be seen in Figure 3.6.

Model	Data	Tolerance	Mean Difference
U-Net + ResNet-18	True Orthophoto	0.25	0.014 m²
U-Net + ResNet-50	True Orthophoto	0.20	0.006 m²
U-Net + SeResNet-18	True Orthophoto	0.30	0.004 m²
LinkNet + ResNet-18	True Orthophoto	0.25	0.008 m²
LinkNet + ResNet-50	True Orthophoto	0.30	0.006 m²
LinkNet + SeResNet-18	True Orthophoto	0.20	0.009 m²
U-Net + ResNet-18	True Ortho + nDSM	0.25	0.015 m²
U-Net + ResNet-50	True Ortho + nDSM	0.25	0.046 m²
U-Net + SeResNet-18	True Ortho + nDSM	0.20	0.009 m²
LinkNet + ResNet-18	True Ortho + nDSM	0.25	0.011 m²
LinkNet + ResNet-50	True Ortho + nDSM	0.25	0.015 m²
LinkNet + SeResNet-18	True Ortho + nDSM	0.15	0.025 m²

Table 3.4 Simplification tolerance values and their mean differences. [119]

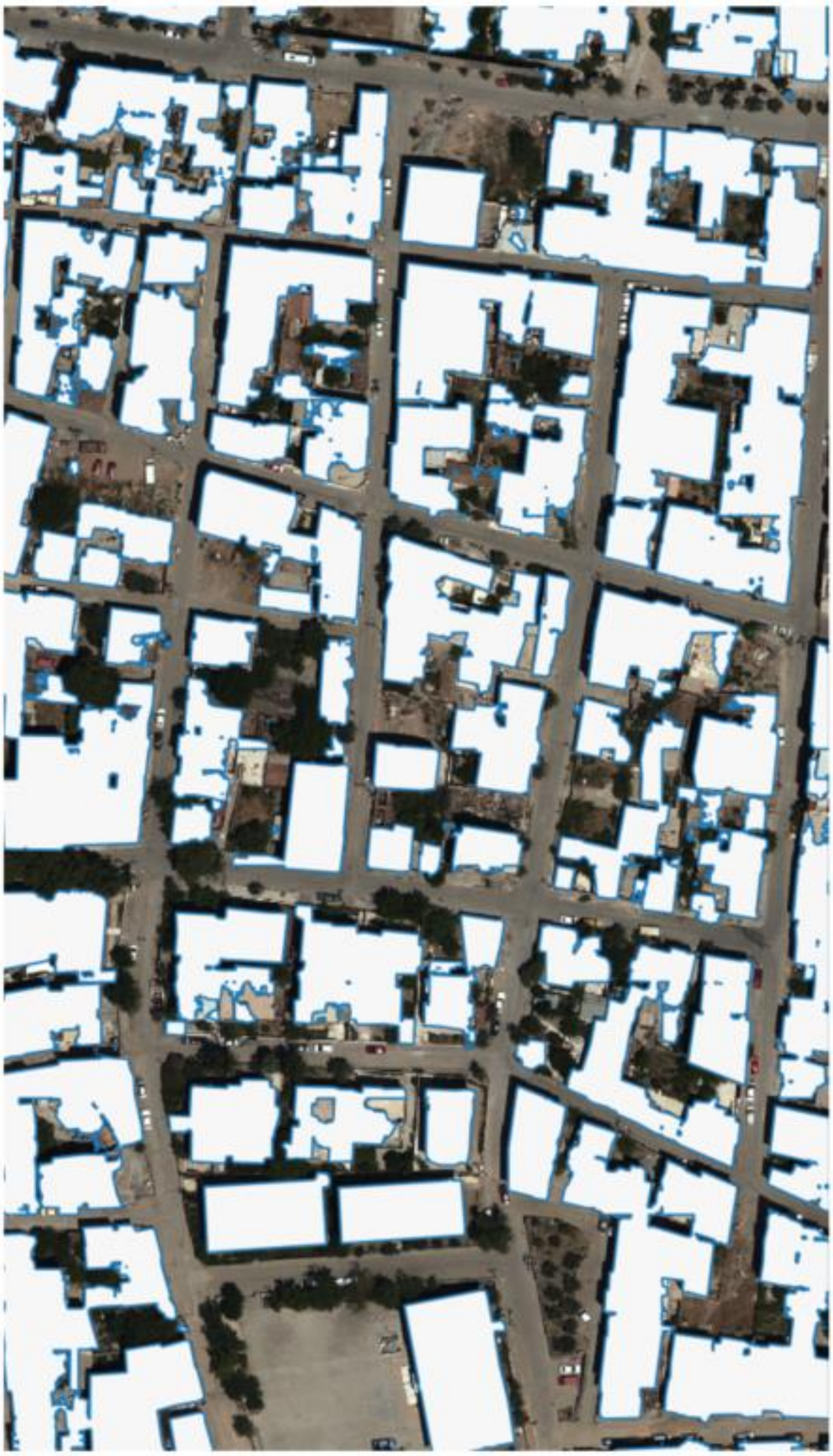


Figure 3.6 An overview of simplified building footprint vectors [119].

3.6. Discussions and Conclusions on Building Footprint Extraction

This section compares the performance of two different CNNs, namely the U-Net and LinkNet with different backbones, for extracting building footprints from VHR true orthophotos and nDSMs. Results are presented based on input data from two sources: RGB data only, and RGB data combined with height information (nDSM). Ground truth mask data is generated from building footprints provided by the The General Directorate of Land Registry and Cadastre of Turkiye (GDLRC) of Turkey.

As a result of the analysis, it can be concluded that fusing nDSM with RGB data significantly improved the performance of the model. Using RGB data, it was found that U-Net with ResNet-18 backbone model was able to achieve both highest F1-score and IoU (IoU) scores 92.9% and 86.7%, respectively. LinkNet with ResNet-50 backbone achieved F1-score of 96.1% and IoU scores of 92.6% as the best results for the data fusion approach (nDSM), respectively. Compared with the test data, it can be seen that F1-score has improved by 3.2% and IoU has improved by 5.9% in terms of accuracy.

The visual inspection revealed that false predictions appeared to be caused because of obscured roofs due to shadows or trees, as well as by the areas that were located between two buildings that were close together. To overcome these problems, it might be possible to train line segment detection networks with more data so that they can directly extract roof structures, instead of classifying them based on pixels. Alternatively, future studies can also provide a deeper understanding of hyperparameters by analyzing them in more detail. Building footprints are generated and updated with this approach, which reduces manual efforts performed by the mapping agencies and enables them to produce and keep up-to-date building data in an effective way.

When the results presented in this chapter are compared with the roof type classification study given in the previous section, it can be concluded that the DL methods generally performed better for building extraction task. Due to the fact that each roof patch is considered as a whole in the roof type classification, and pixel-based segmentation is used in DL-based building footprint extraction, there is expected to be a difference in accuracy between these two approaches. Considering the fact that results are generally dependent amount of the used training data and quality of the ground truth data, results can be improved in future studies by expanding the dataset used in roof type classification or by investigating different DL architectures.

DL-based object detection methods, such as YOLO [124] can also be used for this task. YOLO (You Only Look Once) is an object detection algorithm that uses a single CNN to predict the locations and classes of objects in an image. The YOLO network is composed of 24 convolutional layers followed by two fully connected layers. The network takes the entire image as input and divides it into a grid of cells. Each cell is responsible for predicting bounding boxes and class probabilities for objects that lie within the cell. Using YOLO together with SAHI [125] leads to more accurate results in object detection.

YOLO uses anchor boxes to allow the network to predict objects of different shapes and sizes. YOLO detect objects in an image and draw bounding boxes around them. It works by dividing the image into a grid and predicting bounding boxes and class probabilities for each cell in the grid. YOLO has a single forward pass and predicts all the objects in the image in one go, which makes it very fast and efficient. However, YOLO has difficulty detecting small objects and objects that are close together.

YOLO and segmentation networks, such as U-Net have very different network architectures due to the nature of the tasks they are designed for. YOLO is designed for object detection, where the focus is on predicting the location and class of objects within an image. U-Net is designed for image segmentation, where the focus is on segmenting the image into different regions based on their semantic meaning. However, both architectures use convolutional neural networks and have shown excellent performance on their respective tasks. YOLO is an object detection algorithm that is fast and efficient but may struggle with small or closely-packed objects, while U-Net is a segmentation network that is precise and well-suited for building segmentation but may be slower and require more computational resources.

Building footprints can be used as an important input for 3D building reconstruction. One approach to using building footprints for 3D building reconstruction is to extrude the footprints to generate a basic 3D model (LoD1) of the building. This involves assigning a height value to the footprint and extruding it vertically to create a simple box-like shape representing the building's height. While this method can generate a coarse 3D model of the building, it may not accurately capture the building's shape, style, or architectural details.

Another approach is to use machine learning techniques to learn from the building footprints and generate more detailed 3D models. This involves training a machine learning model on a large dataset of building footprints and their corresponding 3D models, enabling the model to learn the relationships between building footprints and 3D models. Once trained, the model can generate more accurate and detailed 3D models of buildings from building footprints alone.

In summary, building footprints can be used as an important input for 3D building reconstruction, enabling the generation of basic 3D models of buildings and serving as a key input for more advanced techniques, such as machine learning and data fusion.

4. ROOF TYPE CLASSIFICATION USING CONVOLUTIONAL NEURAL NETWORKS

In this chapter, the DL approach implemented for the roof type classification for six categories is presented. A dataset with 10,000 unique roof images was generated based on VHR orthophotos with 10 cm GSD of Cesme, Türkiye is used for the classification task. A detailed comparison of the classification results obtained from different CNNs is also presented in this section. The results given here are largely published in [23].

4.1. Motivation for Roof Type Classification

Despite recent advancements in computer vision and photogrammetry, it remains a challenge for researchers to automate the extraction of building information in urban environments. Roof is a key element in buildings, and it is generally required by applications that work with three-dimensional geographic information systems (3D GIS) in order to know what type of roof it is, and what its geometry is, to perform other analyses on generated models. Model-driven and data-driven methods are two widely used conventional ways for 3D building reconstruction. A model-driven method involves predicting the roof model from the input image or DSM and then matching the roof model against the roof library to reconstruct a 3D building model [126]. Therefore, roofs are essential parts for model-driven reconstruction of 3D building models.

Conventional methods are still frequently used for image segmentation and classification. The main limitation of these techniques can be attributed to the fact that they generally use object features such as corners, edges, and lines. Various methods of artificial intelligence (AI) have been developed over the last few decades for the extraction of features from images. Deep learning, especially CNNs, presents a powerful tool for image classification problems. However, it must be noted that a large volume of training data is necessary to achieve accurate results with the DL-based methods. In this part of the study, the primary goal was to transform very high-resolution orthophotos (10 cm) of

Cesme, Türkiye, into a roof type dataset that contains 10,000 roof images of 6 commonly used roof types. A shallow CNN model was developed in this study for further classification and the results were compared with different fine-tuned CNNs using ImageNet weights. The results of shallow CNN and fine-tuned CNNs using ImageNet weights are given in detail here.

4.2. Study Area and Dataset

The dataset used for training, testing, and validation for the DL models was generated from orthophotos with a spatial resolution of 10 cm. Photogrammetry operators manually delineated the building footprints based on stereo imagery [50]. Manual adjustment of roof edges was carried out by visual inspection of orthophotos with building footprint vectors. Since building footprints lack attribute information, the roof edges were adjusted by visually comparing orthophotos with building footprints. In addition to visual assessments, the roofs under partial or complete occlusion of trees or shadows were eliminated. For the classification task, a roof library comprising six types of roofs that are commonly observed worldwide (flat, hip, half hip, gable, pyramid, and complex) is used. Figure 3.1 illustrates the orthophoto of the study area with building footprints in a general and close up view. An example of a roof vector before and after correction can be seen in Figure 3.2.

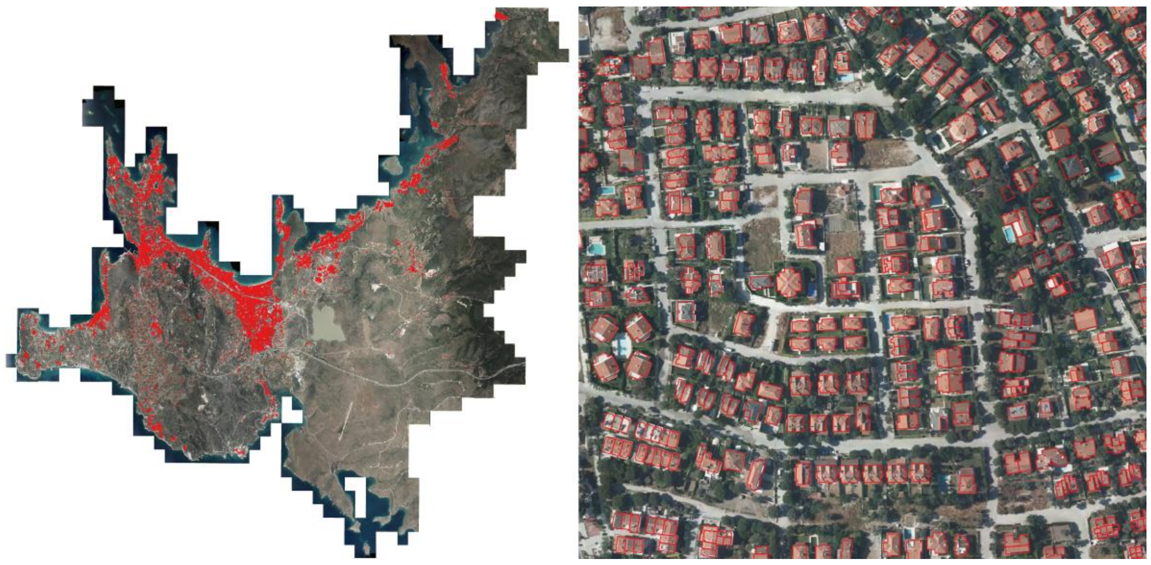


Figure 4.1 A general view (left) and close view (right) of the study area with building footprint vectors

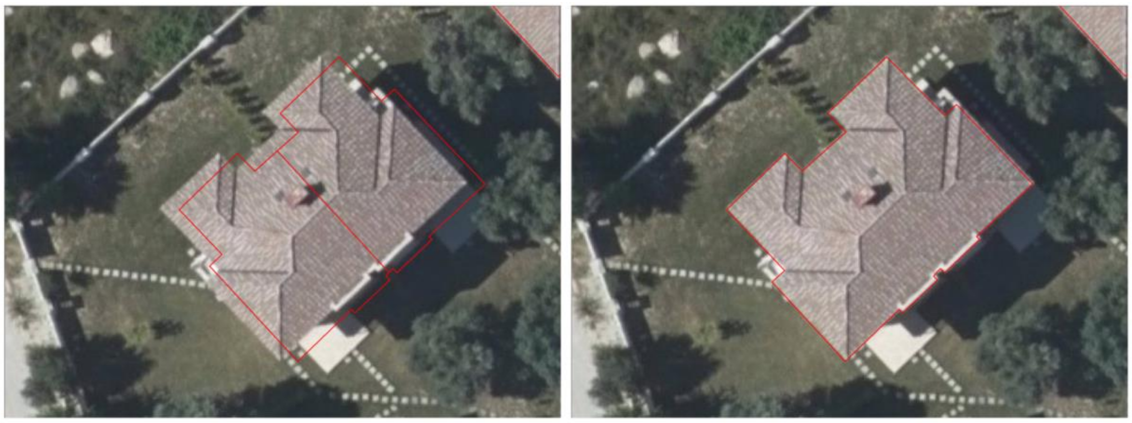


Figure 4.2 An example of a roof vector before (left) and after (right) correction

A sample number of roofs for each type is chosen during the creation of the dataset as a way of maintaining a balance among the roof types. Despite the total number of images for each roof type, the number of samples obtained for classes such as half hip and pyramids is lower than those obtained for other roof types. A number of different ratios were used for the number of training, validation, and test images, and the ratio that yielded the best results was selected. The number of training, validation, and test images for each roof type is given in Table 4.1.

Roof Type	**Training (72%)**	**Validation (18%)**	**Test (10%)**	**Total**
Complex	1620	405	225	2250
Flat	1260	315	175	1750
Gable	1260	315	175	1750
Hip	1260	315	175	1750
Pyramid	1080	270	150	1500
Halfhip	720	180	100	1000

Table 4.1 Number of training, validation, and test images for each roof type.

A further step was taken in the process of clipping the roof patches automatically from the orthophotos by using the building footprints as a reference. Since the study area is tiled into over 900 orthophotos, there is a possibility that a roof could be on the border, which would result in multiple orthophotos covering the same building. Consequently, an orthophotos mosaic with a resolution of 10 cm was generated before clipping to prevent this problem from occurring. Roof image tiles were split and organized based on their type. An illustration of sample roofs from each class can be found in Figure 3.3.

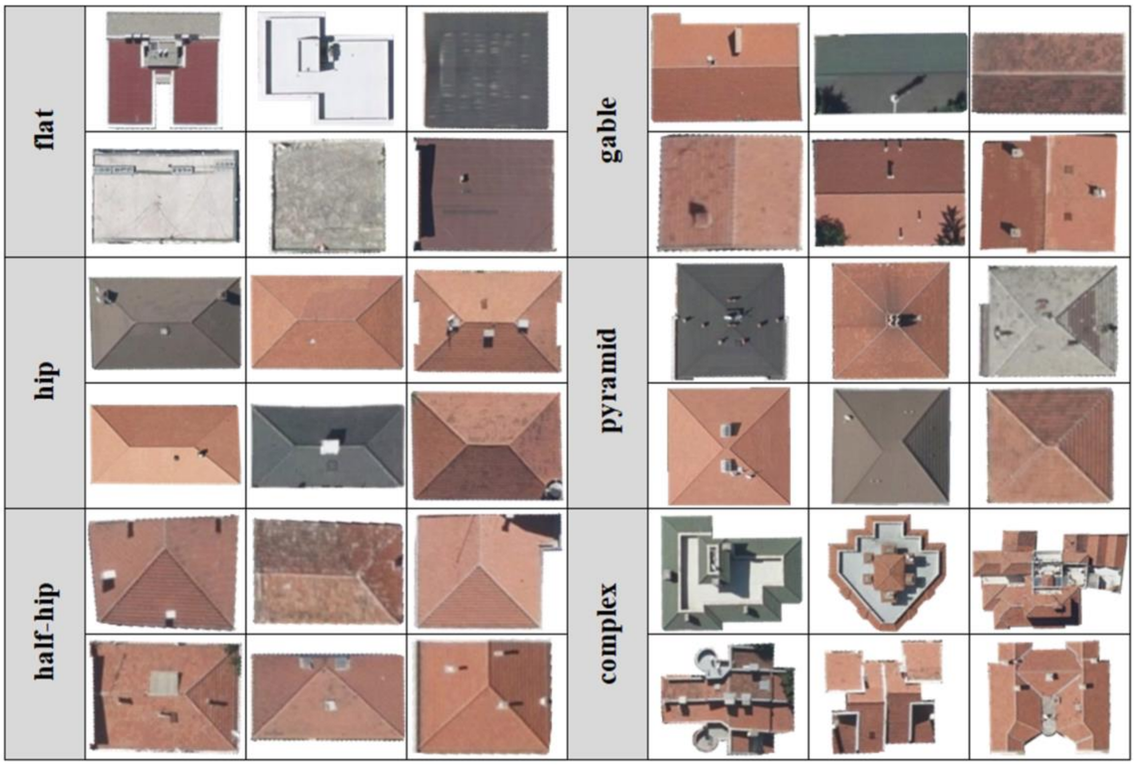
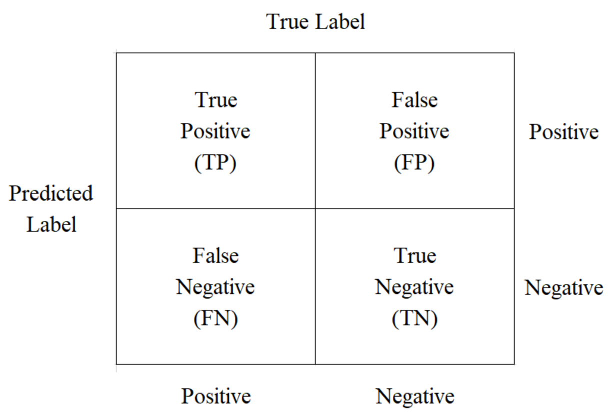


Figure 4.3 Sample roof tiles from each class (Cesme, Turkey).

4.3. Methodology

It has been demonstrated by many studies that deep neural networks are superior for computer vision tasks compared to conventional machine learning methods. A critical step in training the CNNs is finding the optimal hyper-parameters to achieve high accuracy results. The amount of parameters used in a neural network depends on its design. The performance of neural networks is usually enhanced when they are trained on large amounts of data.

A shallow CNN model is also developed for roof classification task, and the results of the model compared to popular fine-tuned CNNs. Classification results of three fine-tuned CNNs using pre-trained ImageNet weights, namely EfficientNet [127], ResNet [128], and VGG-16 [65] are compared with the developed shallow CNN model. Generated roof dataset is split as training (72%), validation (18%), and testing (10%) assessing performance of each DL method. Generated input dataset is used as input for the CNN. Considering the fact that the performance of neural networks is usually enhanced when they are trained on large amounts of data, a data augmentation for training neural networks was applied. A comparison of precision, recall, F1-score, and accuracy measures of the different CNNs is conducted in order to evaluate their classification performance.



$$\text{Precision} = \frac{TP}{TP + FP}$$

$$\text{Recall} = \frac{TP}{TP + FN}$$

$$\text{F1- score} = \frac{2\ x\ precision\ x\ recall}{precision + recall}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FN + TN + FP}$$

Figure 4.4 Depiction of accuracy measures and their equations.

Implemented shallow CNN consists of 312,550 parameters in five convolutional blocks and the convolutional layers (Conv2D) use a 3x3 kernel size (Figure 3.5). Pooling layers (MaxPooling2D, GlobalAveragePooling2D) use the default pool size of 2x2. To avoid overfitting, a batch normalization with a momentum of 0.01 was applied. Instead of using a flattened layer, a global average pooling layer was used to reduce shallow CNN parameters. All CNNs are trained for 150 epochs using Adam optimizer with a learning rate of 0.0003, 64 batch size, and categorical cross-entropy loss function. A detailed view of the shallow CNN model is given in Table 4.2.

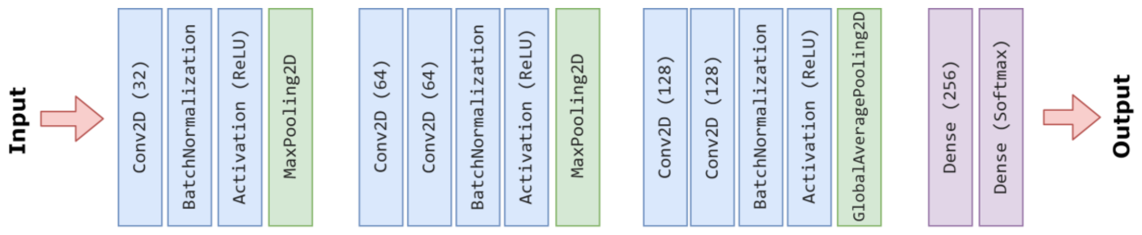
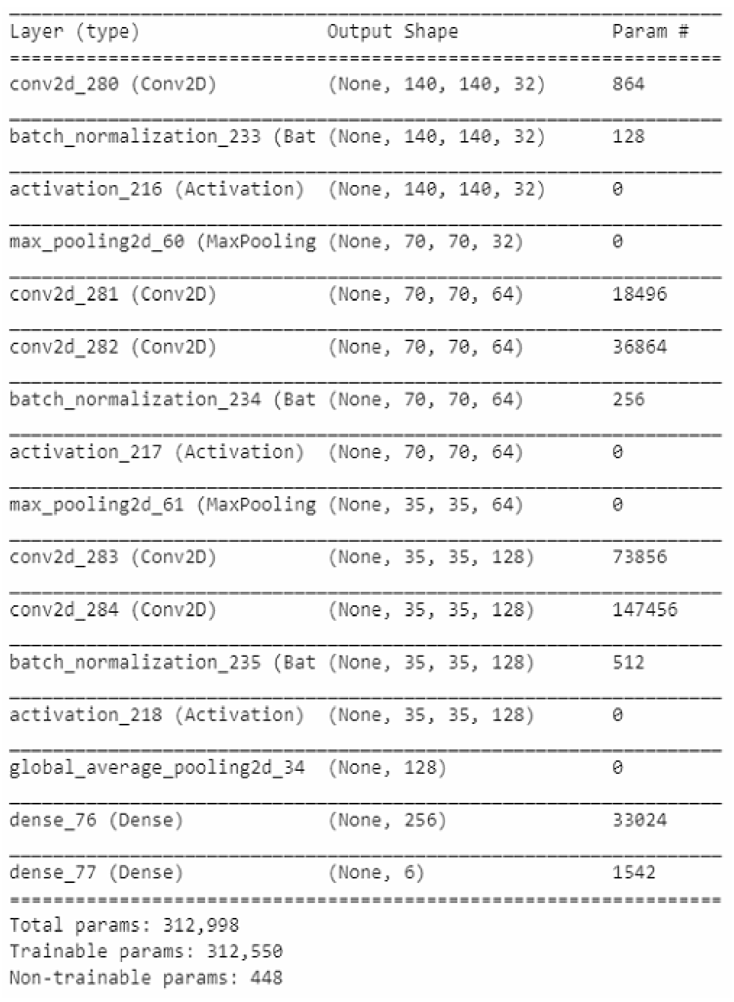


Figure 4.5 Shallow CNN model architecture



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_280 (Conv2D)          (None, 140, 140, 32)      864
_________________________________________________________________
batch_normalization_233 (Bat (None, 140, 140, 32)      128
_________________________________________________________________
activation_216 (Activation)  (None, 140, 140, 32)      0
_________________________________________________________________
max_pooling2d_60 (MaxPooling (None, 70, 70, 32)        0
_________________________________________________________________
conv2d_281 (Conv2D)          (None, 70, 70, 64)        18496
_________________________________________________________________
conv2d_282 (Conv2D)          (None, 70, 70, 64)        36864
_________________________________________________________________
batch_normalization_234 (Bat (None, 70, 70, 64)        256
_________________________________________________________________
activation_217 (Activation)  (None, 70, 70, 64)        0
_________________________________________________________________
max_pooling2d_61 (MaxPooling (None, 35, 35, 64)        0
_________________________________________________________________
conv2d_283 (Conv2D)          (None, 35, 35, 128)       73856
_________________________________________________________________
conv2d_284 (Conv2D)          (None, 35, 35, 128)       147456
_________________________________________________________________
batch_normalization_235 (Bat (None, 35, 35, 128)       512
_________________________________________________________________
activation_218 (Activation)  (None, 35, 35, 128)       0
_________________________________________________________________
global_average_pooling2d_34  (None, 128)               0
_________________________________________________________________
dense_76 (Dense)             (None, 256)               33024
_________________________________________________________________
dense_77 (Dense)             (None, 6)                 1542
=================================================================
Total params: 312,998
Trainable params: 312,550
Non-trainable params: 448
```

Table 4.2 Implemented shallow CNN parameters.

The fine-tuning method involves freezing the base model of pre-trained CNN, and training only a selected few top layers with the new dataset. This method generally enhances the performance of neural networks trained with fewer data. For the classification task, VGG-16, EfficientNetB4, and ResNet-50 networks using ImageNet weights are fine-tuned. Fine-tuning of CNNs is carried out by using weights that have been pre-trained ImageNet [129] dataset. All networks are trained with 64 batch size using cross-entropy loss function and Adam optimizer. After replacement, the modified networks were trained for 10 epochs. Only the fully connected layers are trained during the fine-tuning while the base network layers are frozen. The base network layers are then trained for 10 epochs, then the entire network is trained for 50 epochs for fine-tuning.

4.4. Results

A comparison of the DL models was conducted based on F1-Score, precision, recall, and accuracy measures. A random sample of 10% of each type (1000 images) of the roof was selected from the dataset to test the classification performance of the CNNs. This section presents and discusses the classification results in detail, as well as the methods used.

The shallow CNN model achieved an overall performance of 80% accuracy on the test dataset. Classification results of shallow CNN for each roof type are given in Table 4.3. Normalized confusion matrix of classification results with shallow CNN is given in Figure 4.6. The total number of images used for testing for each roof class is given in the "Support" column of the table. The flat roof type achieved the highest precision and F1-score among the six roof types. Due to the higher number of samples compared to other roof types, complex roofs are classed as the second-best performance roof based on their performance. Half-hip roof type achieved the lowest results as a result of a lower number of samples compared with other roof types.

Roof Type	Precision	Recall	F1-score	Support
Complex	87%	80%	83%	225
Flat	89%	84%	86%	175
Gable	76%	86%	81%	175
Halfhip	74%	62%	67%	100
Hip	78%	75%	76%	175
Pyramid	73%	86%	79%	150
Accuracy			**80%**	1000
Macro avg.	79%	79%	79%	1000
Weighted avg.	80%	80%	80%	1000

Table 4.3 Precision, Recall, and F1-score results of shallow CNN

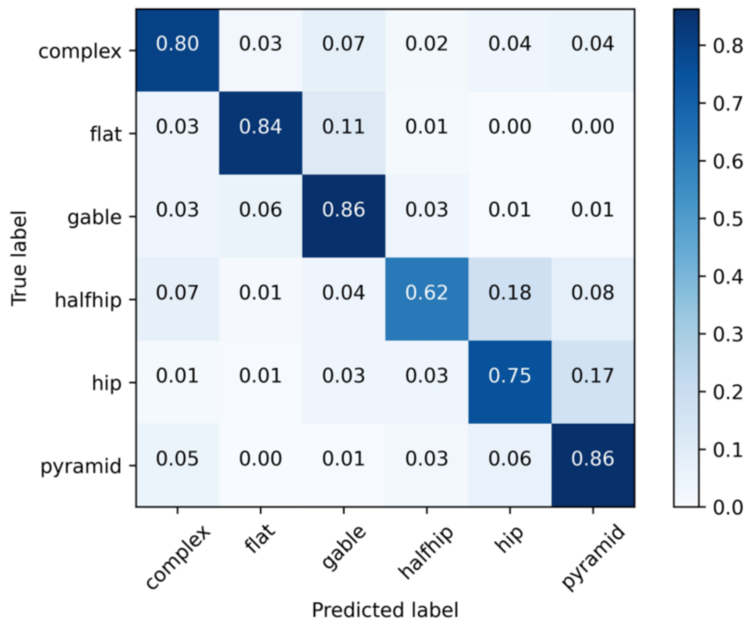


Figure 4.6 Normalized confusion matrix of shallow CNN

VGG-16 achieved 6% better overall accuracy compared to shallow CNN. In terms of precision as well as F1-score values, the pyramid roof type achieved the best results. The half-hip roof type achieved the lowest classification score, as did the shallow CNN. Table 4.4 shows the classification results of fine-tuned VGG-16 network. Normalized confusion matrix of classification results with VGG-16 is given in Figure 3.7.

Roof Type	**Precision**	**Recall**	**F1-score**	**Support**
Complex	90%	80%	84%	225
Flat	81%	93%	86%	175
Gable	90%	84%	87%	175
Halfhip	82%	80%	81%	100
Hip	83%	92%	87%	175
Pyramid	94%	89%	91%	150
Accuracy			**86%**	1000
Macro avg.	86%	86%	86%	1000
Weighted avg.	87%	86%	86%	1000

Table 4.4 Precision, Recall, and F1-score results of fine-tuned VGG-16

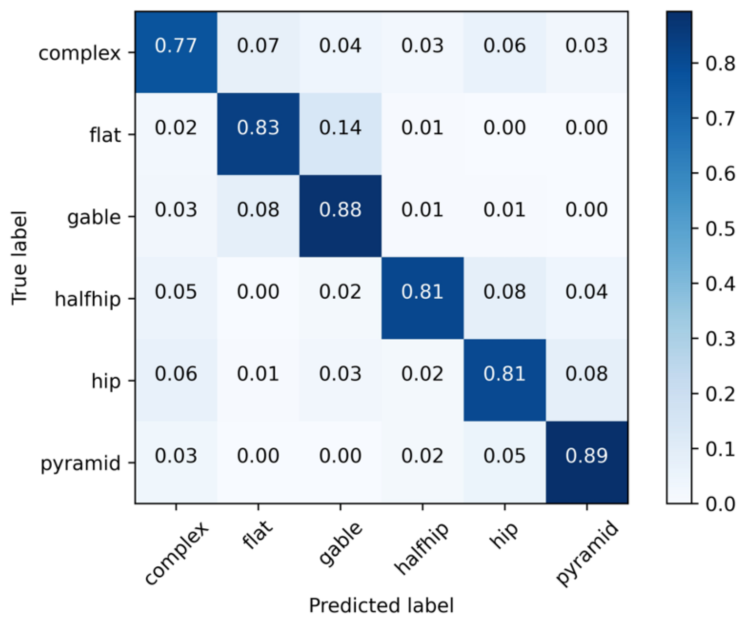


Figure 4.7 Normalized confusion matrix of fine-tuned VGG-16

The EfficientNet achieved 84.4% accuracy on the ImageNet dataset in 2019, placing it among the state-of-the-art CNNs. EfficientNet divides the original convolution into two stages by reducing computational costs by a significant amount with minimal loss of accuracy. It uses linear activation at the last layer in each block to avoid loss of information from ReLU. The model can therefore balance depth, resolution, and width to achieve the highest results possible. Precision, Recall, and F1-score results of fine-tuned EfficientNetB4 are given in Table 4.5. Normalized confusion matrix of classification results with EfficientNetB4 is given in Figure 3.8.The pyramid roof type achieved the highest F1-score, which is similar to the VGG-16 model with an overall accuracy of 83%. It was found that both half-hip roof types (85%) and complex roof types (84%), as well as pyramid roof types (85%), achieved the best results for classification.

Roof Type	Precision	Recall	F1-score	Support
Complex	85%	77%	81%	225
Flat	82%	83%	83%	175
Gable	79%	88%	83%	175
Halfhip	85%	81%	83%	100
Hip	82%	81%	81%	175
Pyramid	84%	89%	87%	150
Accuracy			**83%**	1000
Macro avg.	83%	83%	83%	1000
Weighted avg.	83%	83%	83%	1000

Table 4.5 Precision, Recall, and F1-score results of fine-tuned EfficientNetB4

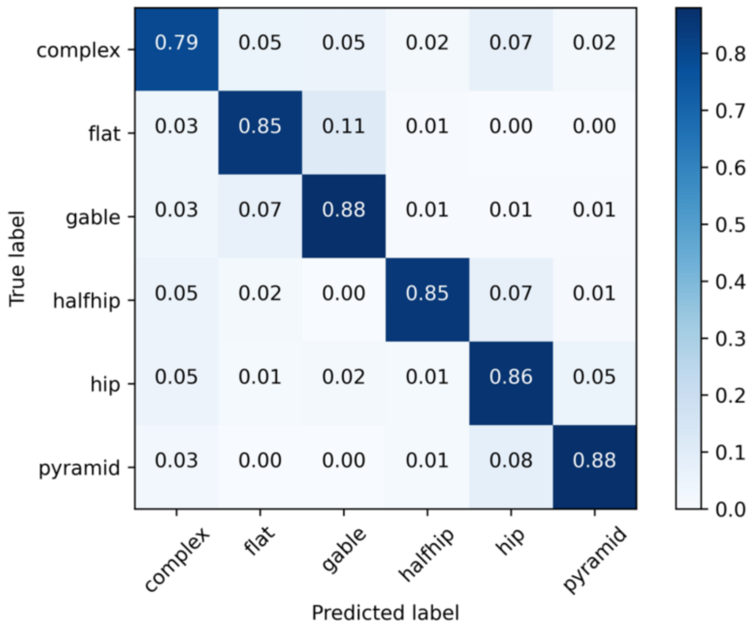


Figure 4.8 Normalized confusion matrix of fine-tuned EfficientNetB4

ResNet-50 is a widely used CNN for classification tasks. The model won the ImageNet challenge in 2015 and is capable of training deep CNNs with thousands of layers. The model achieved an overall accuracy of 85%, which is slightly lower than VGG-16 model. The pyramid roof type achieved the highest score, while the half-hip roof type also achieved high precision and F1-scores, coming in second place. Precision, Recall, and F1-score results of fine-tuned ResNet-50 are given in Table 4.6. Normalized confusion matrix of classification results with ResNet-50 is given in Figure 3.9.

Roof Type	Precision	Recall	F1-score	Support
Complex	86%	79%	82%	225
Flat	84%	85%	85%	175
Gable	82%	88%	85%	175
Halfhip	89%	85%	87%	100
Hip	81%	86%	83%	175
Pyramid	90%	88%	89%	150
Accuracy			**85%**	1000
Macro avg.	85%	85%	85%	1000
Weighted avg.	85%	85%	85%	1000

Table 4.6 Precision, Recall, and F1-score results of fine-tuned ResNet-50.

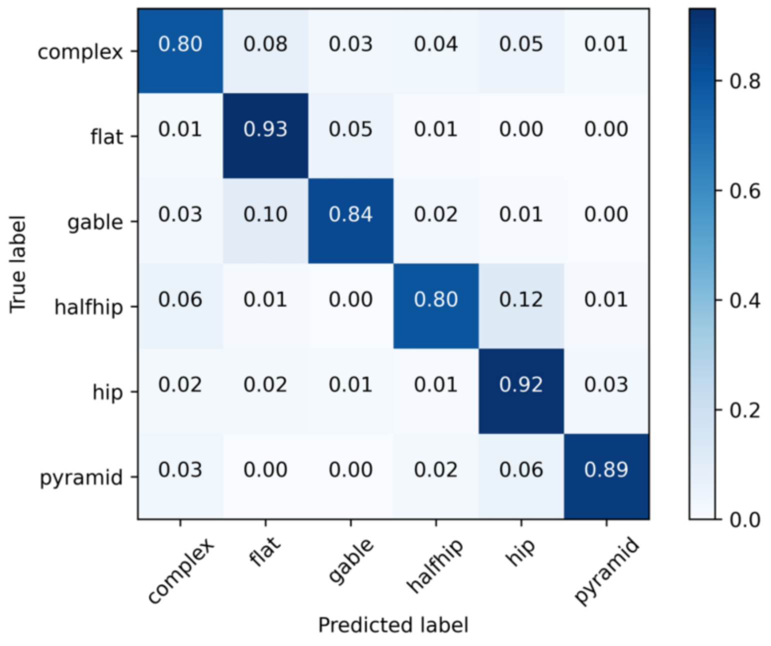


Figure 4.9 Normalized confusion matrix of fine-tuned ResNet-50

4.5. Discussions and Conclusions on Roof Type Classification

In this section of the thesis, a roof type dataset was generated based on VHR aerial imagery. Proposed method can be used to perform automatic roof type detection and classification tasks. The dataset includes six different types of roofs that can be classified according to their characteristics. Classification performance of shallow CNN compared with that of three fine-tuned CNNs namely VGG-16, EfficientNetB4, and ResNet-50. In this section, CNN models are fine-tuned that use pre-trained ImageNet weights before performing a comparison with the other models. A fine-tuned VGG-16 model, which was able to achieve an overall accuracy of 86%, was found to be the best performing model for classifying roof types. Implemented shallow CNN achieved an overall accuracy of 80%. By fine-tuning the models that use pre-trained weights, the performance problems that are usually associated with smaller datasets can be overcome. Based on the accuracy of the four models, it has been found that the values of class accuracy for each model were not uniform. Halfhip class was found to achieve lower scores with the VGG-16 model while it was found to have the highest accuracy value with ResNet-50.

Based on the classification performance of CNNs, there appears to be a promising initial result of the method, however, training data size can be increased in the future to improve model performance. Although accuracy of the used models did not differ significantly from each other, fine-tuned CNNs achieved better scores compared to implemented shallow CNN. Due to the fact that the dataset did not have the same number of samples for each roof type, it can be said that roof classes with more samples, such as the complex roof type, are more likely to achieve better results compared to other roof types in general. The results of the roof classification can be improved in the following ways: the dataset can be expanded with more images, CNN parameters can be optimized, and new categories of roof types can be introduced to the dataset. As a result of the fine-tuned models having promising results, other popular networks may be used in the future to extend the study's findings.

It can be concluded from the overall evaluations that the results of the fine-tuned networks perform better compared to the untrained shallow CNN. In contrast, the shallow CNN model also showed high accuracy although training images in dataset were fewer than the pre-trained weights for other CNN models. As a result, the model can be further tuned to increase the performance of the prediction. Building roof types affect the affect the overall appearance, functionality, and durability of a building. Different roof types have different shapes, slopes, and angles, which can greatly impact the overall appearance of a building. The shape and design of a roof can also affect ventilation and airflow within a building.

A roof with a steep pitch or multiple slopes like a hip roof can create better air circulation, which can help to regulate temperature and prevent moisture buildup. The type of roof can also impact energy efficiency, particularly in terms of insulation and heat retention. For example, a flat roof may require more insulation to prevent heat loss in colder climates.

On the other hand, the output of roof type classification approaches are useful inputs for semi-automatic building modeling software that are based on pre-defined libraries. Although roof-type library based semi-automatic approaches represent an earlier stage in the city modeling applications, they are fast and efficient to build models at large scale. The proposed study demonstrated that high accuracy can be achieved with shallow CNNs that can be applied regionally, although the number of roof types is the main limitation.

Roof types can be used as an important feature for procedural building reconstruction. Procedural building reconstruction refers to the automatic generation of 3D models of buildings from 2D images or point cloud data. In this process, roof types can provide valuable information about the shape and style of a building's roof, which can be used to generate more accurate and realistic 3D models. One approach to using roof types for procedural building reconstruction is to classify the roof type of each building based on its characteristics, such as the number of slopes, roof pitch, and overall shape.

Once the roof types are classified, they can be used to generate 3D models of buildings with similar roof types. This can be done using procedural modeling techniques, such as rule-based or parametric modeling, which generate 3D models based on a set of rules or parameters that define the building's shape and style. By using roof types as a key input for these models, the resulting 3D models can be more accurate and realistic, as they reflect the characteristics of real-world buildings with similar roof types.

This study automated the classification of roof types using deep learning, allowing it to be automated more quickly with high accuracy. In future studies, it may be possible to improve results by using a larger dataset or by experimenting with different deep learning algorithms. The main challenge is to generate dataset for deep learning, which requires manual effort for assigning each roof type to a class. Data fusion from different sources, such as RGB+height information may improve classification performance of the models.

In different computer vision and image recognition tasks, Vision Transformers [130] are now becoming more popular as competitive alternatives to CNNs. The vision transformer architecture replaces the convolutional layers used in traditional CNNs with a transformer encoder. The input image is first divided into a fixed number of non-overlapping patches, which are flattened and fed into the transformer encoder. The transformer encoder then processes the patches and outputs a classification label. One advantage of the vision transformer architecture is that it does not rely on hand-crafted features or prior knowledge about the images. Instead, it learns to extract meaningful features directly from the image patches using self-attention mechanisms. This allows the model to achieve state-of-the-art performance on several image classification benchmarks, even with limited amounts of training data. Overall, transformers offer a promising approach for image classification, and their effectiveness in this domain is an active area of research. As a future work, transformed-based classification architectures can be used for roof type classification or similar tasks.

5. AUTOMATIC LOD 2.2 BUILDING RECONSTRUCTION WITH DEEP LEARNING

In this chapter, investigations on another approach for extracting the roof geometries in the form of vectors (lines) are presented and discussed. A DL-based framework for fully automated vectorization of LoD 2.2 roof details in city-scale is proposed. This method does not depend on any roof library, and can also detect complex roof shapes. As a part of this section, a new dataset is generated from existing 3D city models and VHR true orthophotos. Performance of the proposed framework assessed both quantitatively and qualitatively. Also, robustness of the method is analyzed on cases such as blocking objects over roofs like trees and shadow.

5.1. Motivation for Automatic LoD2.2 Building Reconstruction

Automatic extraction of roof line structures beyond LoD2.0 is still a difficult problem due to complexity of roof structures and the difficulty of detecting smaller objects such as chimneys and small windows. Very high resolution aerial images make several details visible over the roofs, making it difficult to detect them. In this section, a framework for fully automated vectorization of roof line segments in LoD2.2 using line segment detection networks is proposed. The approach can be applied at city scale. Unlike the library-based methods, the proposed approach does not depend on pre-defined roof types and can also reconstruct complex roofs and objects smaller than 1 m^2. A training dataset with more than 2,2 million lines and more than 139 k buildings measured in LoD2.2 from true orthophotos with 8 cm resolution with sub-pixel accuracy were used. The results show an improvement to the recent studies on the extraction of LoD2.0 roof structures. The robustness of the method was also analyzed w.r.t. roof-blocking objects such as trees and shadow areas.

5.2. Study Area and Data Preparation

Ankara is the capital city of Türkiye with 5.1 million population, making it Turkiye's second most populated city after Istanbul. In this study, experiments were performed on a custom dataset that covers Gölbaşı town near Ankara. A general and close view of the study area with training and test tile distribution are given in Figure 5.1. A total of 139k+ buildings with roof details in LoD 2.2 were used in this research. There are several types of roofs in the study area, which are generally used in various parts of Türkiye, resulting in a broad variety of roof types.

A total of 2.2 million lines of roof details were used in this study. LoD 2.2 roof details were manually measured by experienced photogrammetry operators using aerial stereo imagery. Together with the LoD 2.2 roof details, high-resolution true orthophotos of 8 cm GSD were used in the dataset. All data used in this study were provided by GDLRC. The roof details with an area of less than 1 m^2 were kept in the dataset. An example of a LoD2.2 building roof with multiple small roof details overlaid with very high-resolution true orthophoto is given in Figure 5.2. The DSM of the study area was generated by GDLRC from stereo aerial imagery. DTM is generated by applying ground filtering to DSM. Additionally, a second DSM was created using height values of 3D roof geometries.

Several pre-processing steps were carried out during the data preparation phase. Since all line segment detection networks used in the study received inputs in different data structures, different data processing scripts were generated for each network. The data processing pipeline scripts have been written in Python to automate operations such as image tiling, coordinate transformation, ground truth generation, and merge of output files into shapefiles as batch processes.

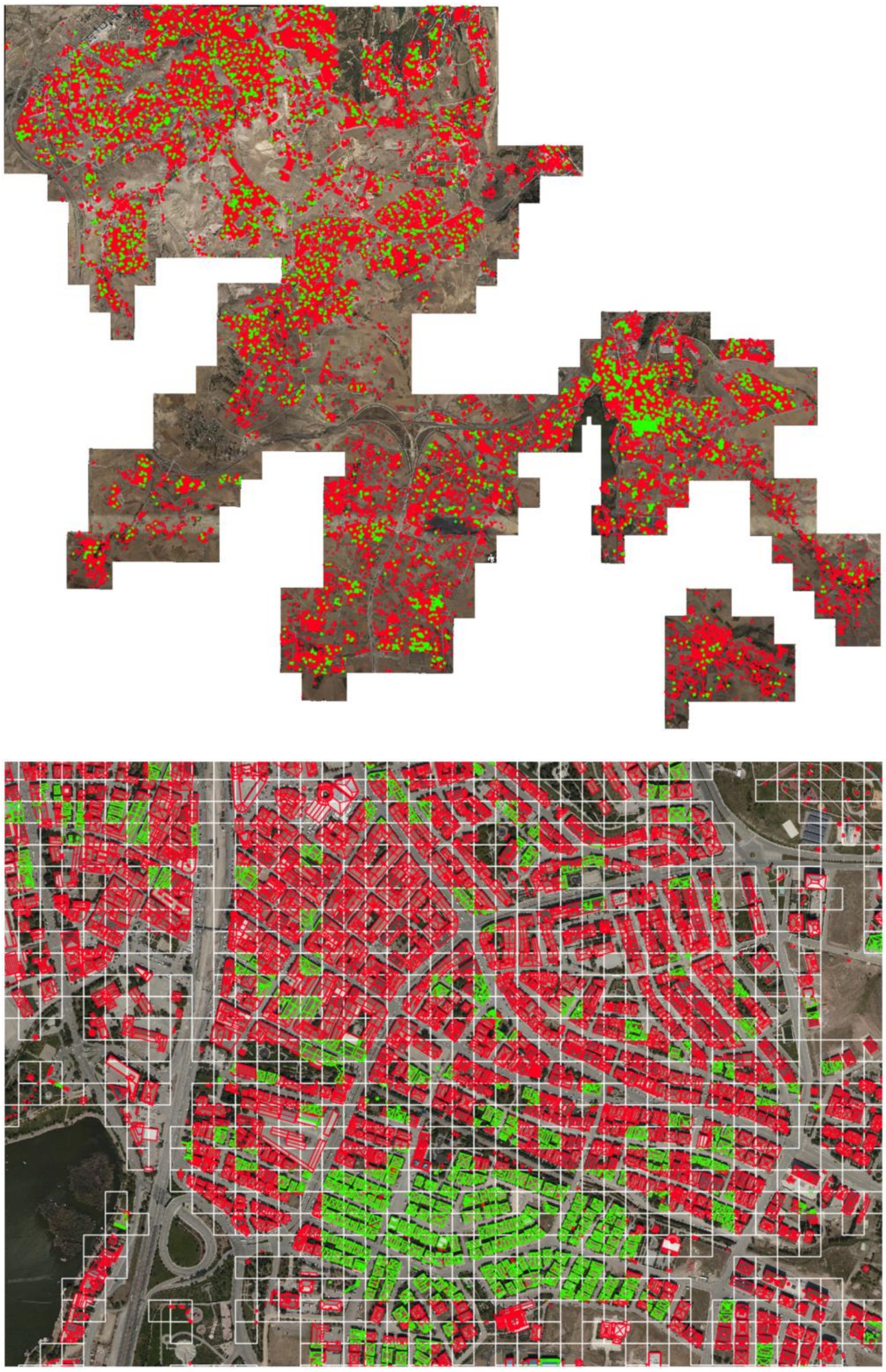


Figure 5.1 A general and zoomed view of the study area with training tiles (red) and test tiles (green) distribution

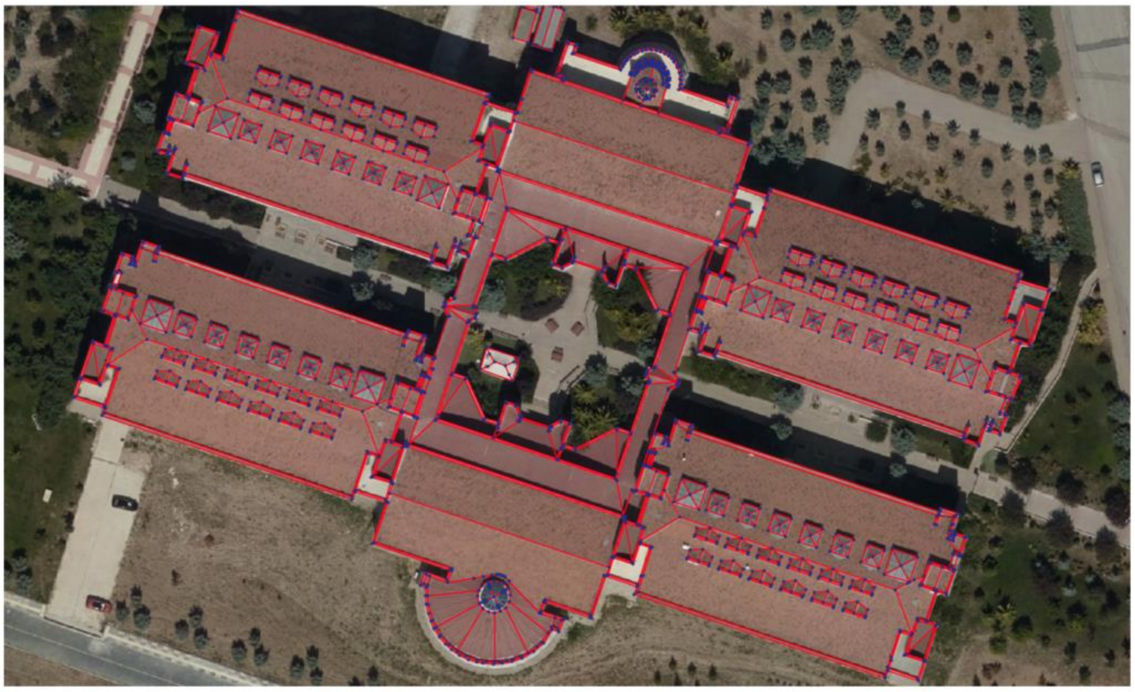


Figure 5.2 A close view of complex LoD2.2 building roof structure

8 cm resolution true orthophoto covering the entire study area was divided into a non-overlapping grid with 512 x 512 tiles (see Figure 5.3 for an example). Vector data of roof details for each tile was clipped in the same grid. Ground truth data in the projected coordinate system was then reprojected into pixel coordinates with sub-pixel accuracy (up to 15 digits). Since it would not be efficient to use images with very few lines in the dataset, image tiles that do not contain any buildings or contain very few lines (less than 7) are automatically excluded with a Python script. The remaining dataset contains 30,446 tiles with 512x512 pixel size and approximately 2.2 million roof detail lines at LoD 2.2. Since validation data was not used in similar line segment detection studies and benchmark datasets, the dataset was divided into training and testing. Both the total number of image tiles and lines in the dataset were split as 90% for training and 10% for testing. The number of lines and tiles after data split is given in Table 5.1.

Data Type	Percentage	Number of lines	Number of tiles
Training	90%	1,982,313	27,402
Test	10%	220,257	3,044
Total	100%	2,202,570	30,446

Table 5.1 Total number of lines and image tiles used for training and test data

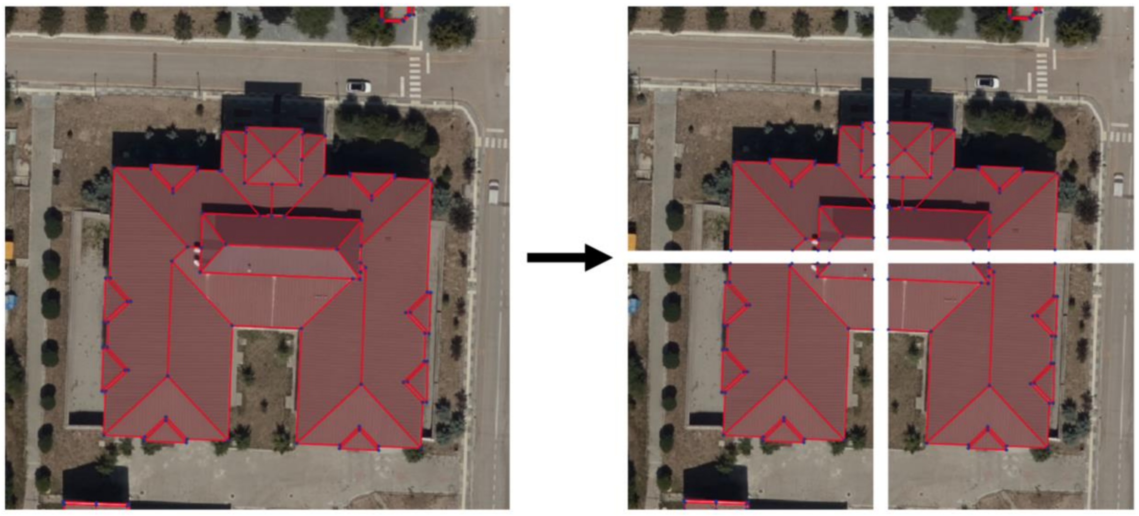


Figure 5.3 An example of a non-overlapping tiled image with roof lines and junction points

In order to ensure heterogenous geographical distributions, the test data were randomly distributed throughout the entire study area. A small area with neighboring tiles was also tested to analyze the method's city-scale reconstruction performance. Images of roofs that are blocked by trees and shadows in test data is also included to test algorithm robustness.

5.3. Methodology

To fully automate the vectorization of city-scale LoD2.2 roof structures, this study proposes a multi-step methodology that uses in-house- developed custom Python scripts with line segment detection networks to fully automate the vectorization process. In the first step, a selected line segment detection network was trained using the proposed LoD2.2 roof line segment training dataset. Once the training has been completed, roof segments are predicted from the input image tiles using the trained model. Using Python scripts developed in-house, roof segments are vectorized, merged, and reprojected after being predicted from image tiles. After vectorizing, post-processing was applied to the generated dataset to eliminate redundant junctions, fix line connections between tiles, and simplify necessary lines. As the last step in the methodology, LoD2.2 roof structures are exported into the intended format of a vector file such as ESRI Shapefile, GeoJSON, etc. An overview of the proposed methodology for fully-automatic vectorizing city-scale LoD2.2 roof structures is presented in Figure 5.4.

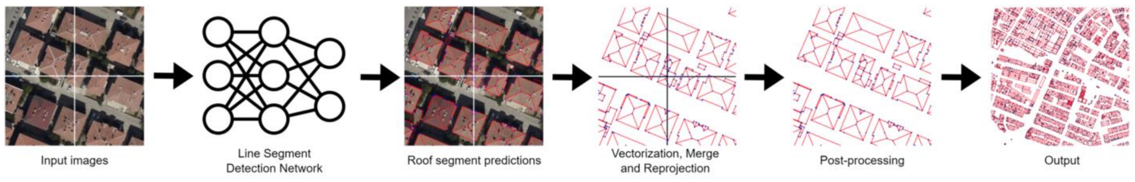

Input images
Line Segment Detection Network
Roof segment predictions
Vectorization, Merge and Reprojection
Post-processing
Output

Figure 5.4 Overall view of the methodology

Several line segment detection networks have been proposed by researchers to detect and extract planar lines in images. In this study, ULSD (Unified Line Segment Detection) [131] is adopted for predicting roof line segments from VHR RGB imagery. This network was chosen due to its high efficiency for detecting line segments in benchmark datasets such as the Wireframe [132] and YorkUrban [133] datasets. In addition to extracting planar structures, ULSD can extract curved lines as well, using its workflow to include the Bezier Curve. Stacked hourglass network [134] come in the form of a U-shape and are mostly used for human pose estimation. Used DL network are trained from scratch using the generated LoD 2.2 roof line dataset. As last, roof line segments and junctions were predicted in the images of the test data given as input.

5.3.1. Data Augmentation

Data augmentation can be used for increasing the training data by applying several pre-processing techniques to a dataset. As part of this study, the training dataset was augmented to quadruple it by increasing from a total of 27,402 images to 109,608. Horizontal flipping, vertical flipping, and horizontal shifting were used for this purpose. The ground truth that incorporates the line information of the augmented images has also been produced. Figure 5.5 shows an augmented sample image and ground truth data for the generated images.

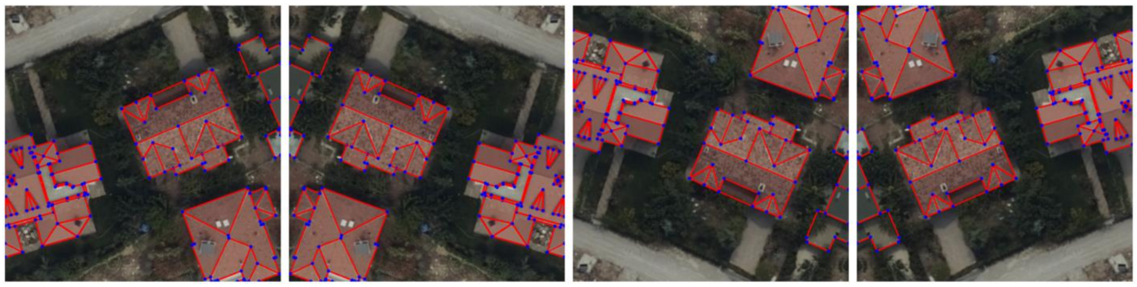


Figure 5.5 Augmented files with ground truth a. original file, b. horizontal flip, c. vertical flip, d. mirroring

5.3.2. Evaluation Measures

Heatmap-based measures were originally developed for evaluating boundary detection methods. The main issue in their use lies in that they do not penalize overlapping lines or evaluate line connections properly, which makes them ineffective for wireframe detection. To improve the assessment of line segment quality and the structural quality of wireframes, Zhou et al. [135] proposed Structured Average Precision (sAP) measure for assessing the accuracy of detected lines and junctions.

The msAP metric is calculated by comparing the predicted line segments with the ground truth line segments. The metric computes the average precision of the predicted line segments at different intersection over union (IoU) thresholds, and then calculates the mean of the average precision scores. The IoU is a measure of overlap between the predicted and ground truth line segments.The msAP metric is useful because it provides a comprehensive evaluation of the line segment detection algorithm's performance across different levels of IoU threshold, which is important in real-world applications where different levels of accuracy may be required.

In line segment detection, the line uncertainty is often quantified using a measure called the angular uncertainty, which represents the range of angles over which the line can be considered a valid match to the ground truth line. The angular uncertainty is usually defined as a threshold on the angle difference between the predicted line and the ground truth line. In the calculation of msAP, the line uncertainty is taken into account by considering only those predicted lines that have an angular uncertainty below a certain threshold. This is done to ensure that only the most accurate and precise predictions are included in the evaluation.

Specifically, during the calculation of the average precision for a given IoU threshold, only those predicted lines that have an angular uncertainty below the threshold are considered as valid detections. Any predicted line with an angular uncertainty above the threshold is considered a false positive and is not included in the calculation of the average precision. By including the angular uncertainty in the evaluation of line segment detection algorithms, msAP provides a more comprehensive and realistic measure of performance, considering both accuracy and precision of the predictions.

The quality of line segments is evaluated by the structural average precision under the threshold of 5, 10, and 15 pixels (sAP^5, sAP^{10}, sAP^{15}) and mean structural average precision (msAP). The quality of junctions is evaluated by the vectorized junction mean AP (mAP^J), which is computed over the threshold of 0.5, 1.0, and 2.0 pixels.

5.3.3. Implementation Details

Non-overlapping RGB image tiles with a size of 512 x 512 pixels were used during the training and testing phases. Once the tiles were generated, they were fed into the backbone network for producing feature maps. Stack-hourglass network settings are based on defaults settings of as L-CNN [135] and HAWP [136]. Learning rate and weight decay are set to 4×10^{-4} and 1×10^{-4}, respectively. The learning rate decayed at the 25th epoch. A data augmentation experiment was conducted to examine the results of data augmentation as a part of the study. Batch size was set to 32 for both testing and training. Stacked hourglass network with 30 epochs is usually sufficient for obtaining accurate results, and a higher number of training epochs usually do not provide significant improvement. All model trainings are performed on a single Quadro RTX 8000 GPU with 48 GB of memory. It takes 25 hours to complete the model training.

5.4. Results

In this section, quantitative and qualitative results of line segment detection networks used in the study are presented. The robustness of the methods is also tested based on images with trees, shadows, and other obstructions blocking the roofs.

5.4.1. Quantitative Results

A summary of the results can be found in Table 5.2. Despite the increased complexity of the extraction of LoD 2.2 roof details, the proposed method achieved very high performance. Unlike the other methods, ULSD utilizes the Bezier curve for detecting line segments. As a result of the experiments, order 4 for defining Bezier curve complexity was found to provide the most accurate measures and used here. Since the ULSD was mainly used the Wireframe dataset which is a benchmark dataset for indoor wireframe parsing, it still performed well for roof segment extraction. Based on the results, ULSD was able to achieve state-of-the-art results for roof line segment extraction task.

Data	sAP^5	sAP^{10}	sAP^{15}	msAP	mAP^J	FPS
No Augmentation	52.5	56.3	58.3	55.7	69.0	42.0
4x Augmentation	**55.3**	**59.0**	**60.8**	**58.4**	**72.8**	**42.8**

Table 5.2 Evaluation results of non-augmented and augmented data

It was shown that the performance of the model has been enhanced through the use of augmented data. Compared to the original data (without augmentation), msAP increased by 2.7%, and mAP^J increased by 3.8%. The visual inspection also revealed better extraction of the small details of the roof when using data augmentation.

5.4.2. Qualitative Results

Figure 5.6 shows a reconstruction of predicted roof segments over a threshold (80% here) using ULSD network on the LoD2.2 dataset**.** In terms of both measures and visual results, the ULSD was successful in detecting the LoD2.2 roof segments in most cases. It was able to detect roof structures that were even smaller than 1 m^2 and was also capable of detecting line segments in complex roof structures. Additionally, the visual results also show that ULSD generates fewer redundant lines outside buildings as compared to other methods. ULSD achieved good results when detecting buildings with curved roof structures due to the use of the Bezier curve. However, the ULSD method has failed to detect some of the roof structures as well.

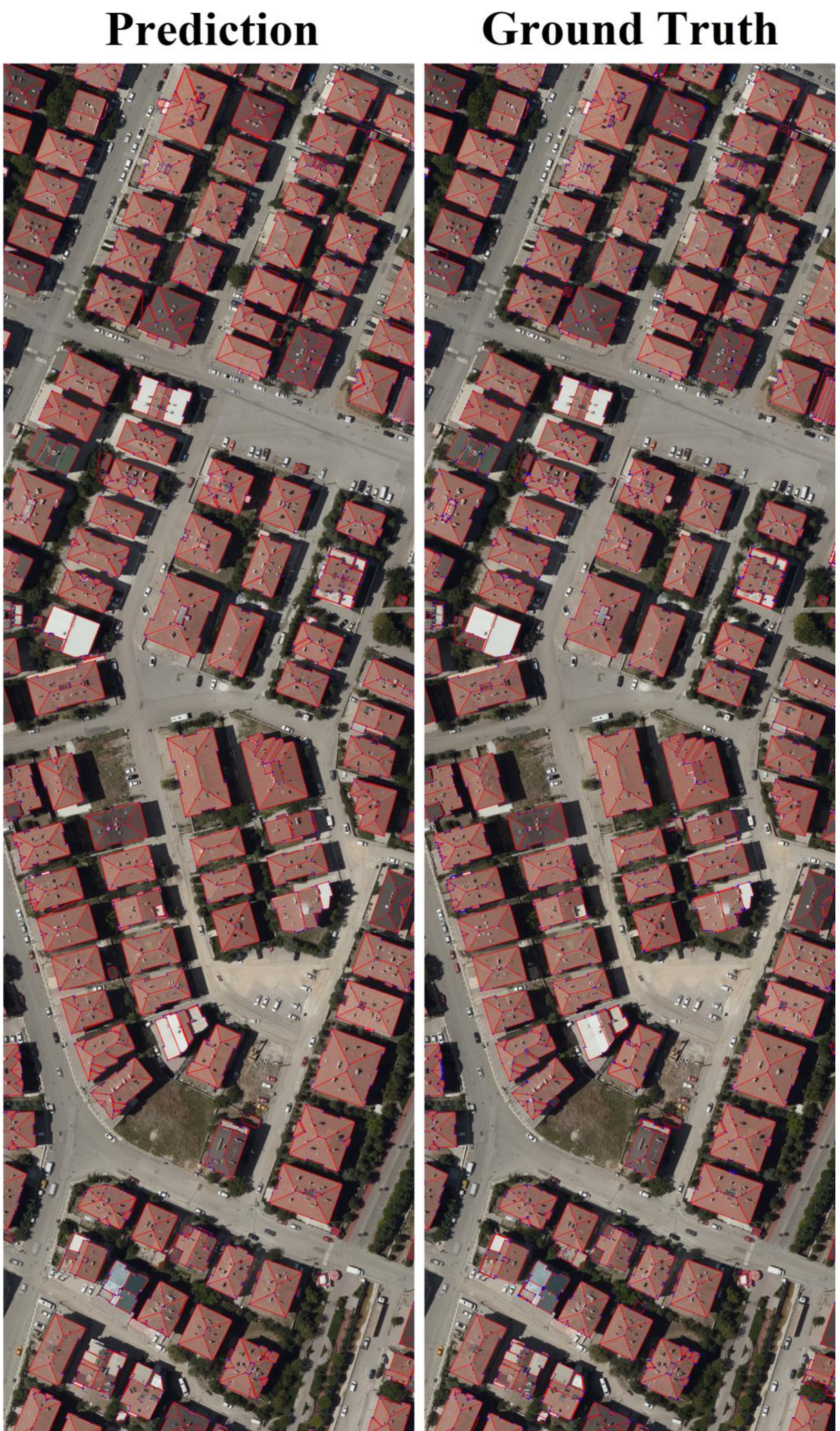


Figure 5.6 Ground truth and predictions in a test area (0.8 threshold)

5.4.3. Method Robustness

The main objective of this section was the evaluation of the robustness of algorithms w.r.t. roofs of buildings completely or partially covered by trees, as well as against shadows cast by buildings. Many computer vision methods are still struggling to correctly reconstruct or detect blocked building roofs. To evaluate the robustness of the line segment detection networks, a number of manually selected tiles for test data with roofs that were partially or fully covered with trees. Roof line segments with over 80% score for predictions were visualized. A separate analysis for blocking trees and shadows was given. A comparison of the results from the different methods of predicting tile images with trees blocking roofs given in Figure 5.7. Predictions indicate that the developed framework was also successful in predicting roofs blocked with trees in most cases.

A comparison of the results from the different methods of predicting tile images in shadow areas can be seen in Figure 5.8. The ULSD did not generate any redundant lines and does not detect edges of shadow areas as lines. Although neural networks outperform the conventional methods, they are still far from perfect. Their performance is also heavily dependent on the number of examples with blocked roofs or shadows in training data. More blocked roof segments can be used to train the model to increase its performance and robustness for such cases. A total of 58 lines were detected with a score of 80% or higher out of 71 roof structures obscured by trees or shadows.

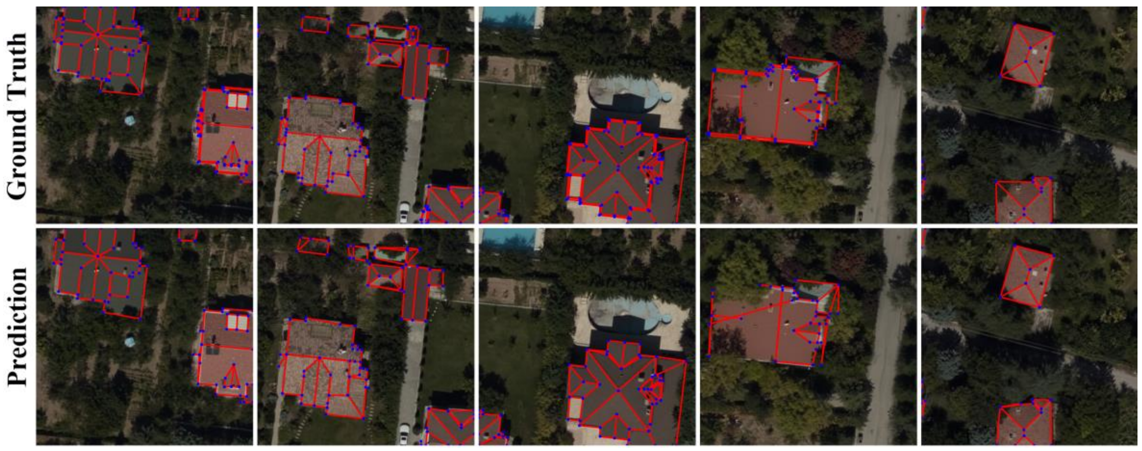


Figure 5.7 Ground truth and predicted roof structures with blocked trees

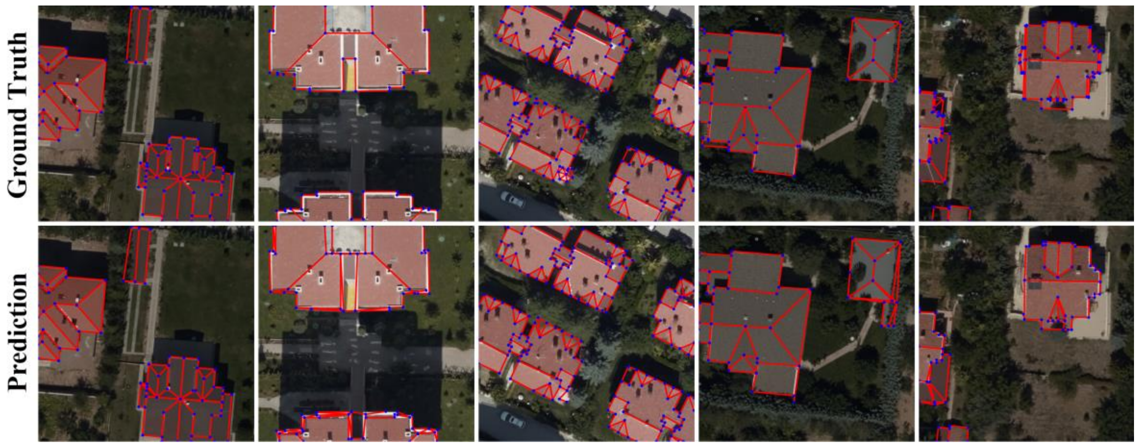


Figure 5.8 Ground truth and predicted roof structures with shadowy areas

5.5. Vectorization and Post-processing

The vectorization process was carried out with a Python script developed for this purpose. By using roof line predictions and worldfiles containing the coordinates and orientation information of each image, the entire vectorization process can be carried out automatically as the batch process. The roof lines on the test data were predicted using the model obtained after the training. For each tile of the test data, the predicted lines and their scores are produced together and saved as a *numpy* script. Only lines above a given threshold are regarded as vector data. The roof lines predicted for different threshold scores are given in Figure 5.9.

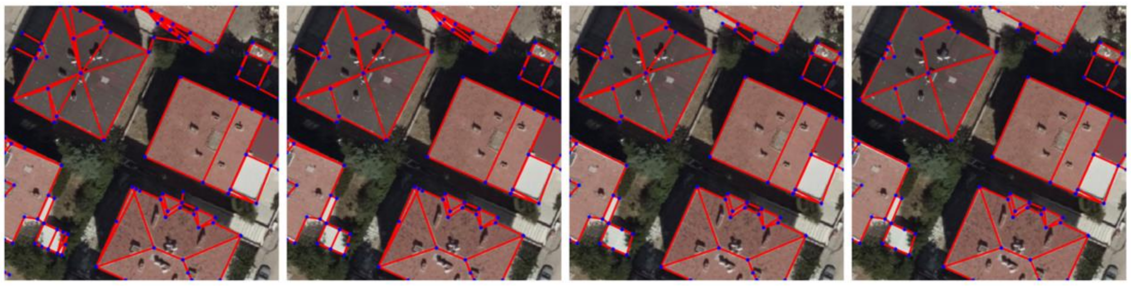


Figure 5.9 Visualization of roof structures for 0.2, 0.4, 0.6, and 0.8 score thresholds.

In the coordinate transformation process, each predicted line was automatically reprojected from the pixel coordinate system to an Earth referenced coordinate system (EPSG:5255). All roof lines were generated with a sub-pixel accuracy for training and testing. Predicted roof lines were converted from pixel coordinates to projected coordinates using a worldfile that contains coordinate and rotation information for each tile. The data from Gölbaşı district used in this study were converted from pixel coordinate system to TUREF / TM33 - EPSG:5255 coordinate system. The EPSG code parameter used in the script can be changed to convert to a desired coordinate system. A final step involves automatically merging the roof lines of each tile into a single vector file as a batch process. The merged vector file can then be exported in the desired format such as ESRI Shapefile or GeoJSON. Merged roof detail prediction tiles with very high-resolution true orthophoto basemap is given in Figure 5.10.

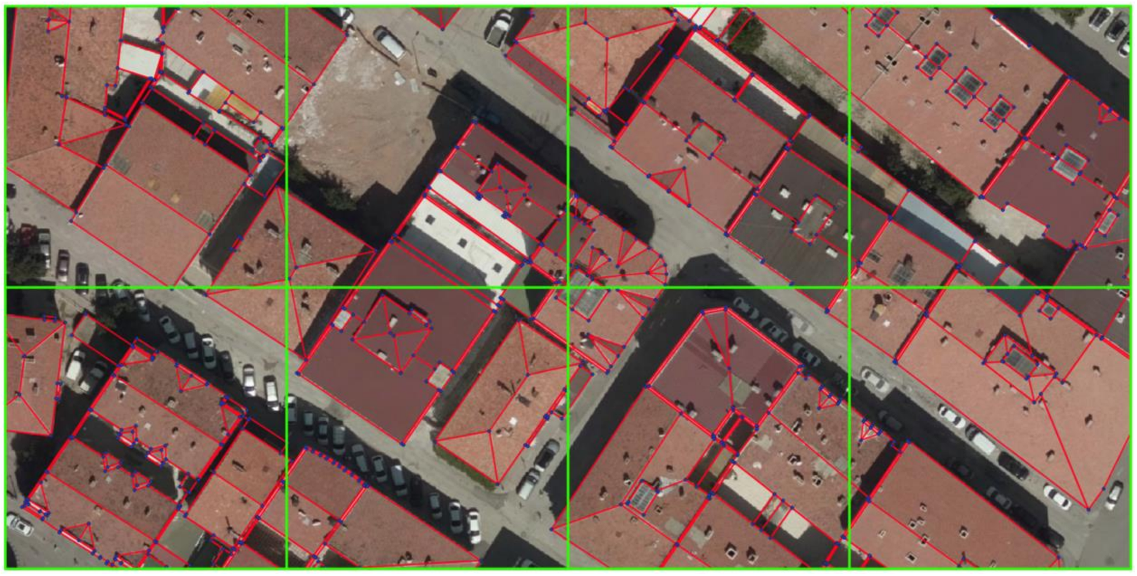


Figure 5.10 Merged roof structures of 8 image tiles.

A post-processing process was applied to the generated data after vectorization and merging processes. A custom workflow is developed for post-processing based on Python. Since orthophotos were not clipped based on building footprints, a single building usually appears in more than one image tile. As a result, there are some tiny gaps between line endpoints between predicted roof lines in image tiles. The result is that single lines are split into multiple lines and redundant lines, thus redundant junctions are created. As part of post-processing, the gaps between the roof lines in the transitions between the tiles were removed, redundant lines and junctions were removed, and multiple continuous lines are merged into a single line.

As a first step, the endpoints of each line segment were joined to eliminate any gaps in the roof line structures extracted in different tiles. Second, the lines were merged into MultiLineString. A final step in the reduction of the size of the data is to simplify it by removing unnecessary junctions and breaks that appear along the lines to reduce its size using the Douglas-Pecker simplification algorithm. Once the post-processing was completed, junction points and the overall number of lines decreased by 24.8% and 35.4%, respectively when compared with the predicted raw data. Figure 5.11 shows the post-processing steps for vector data of a building in more than one tile along with ground truth data.

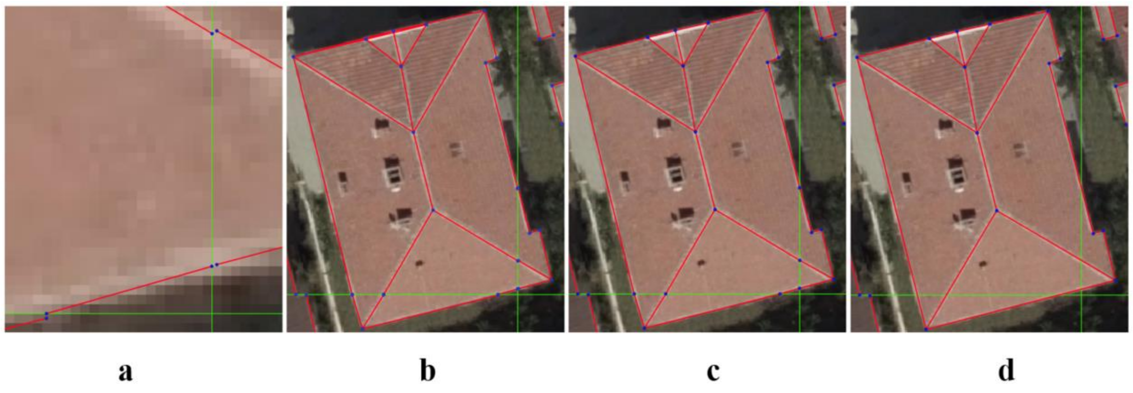


Figure 5.11 a – close view of the predicted line, b – snapped, c – simplified, d - ground truth.

5.6. 3D Building Reconstruction

Here, 3D building models were produced from the predicted roof structures and the elevation data. Following the generation of building geometries, multiple data types are used, such as raster DSM data and photogrammetric point clouds for 3D building reconstruction. The results reveal that when building vectors are overlayed with DSM or point clouds, the geometries of buildings do not match elevation data most of the time, which leads to false reconstruction of building geometries. In most cases, the generated DSM or point cloud is not able to give an accurate representation of critical roof points such as corners and junctions, which make up the roof structure. As a result, it was not possible to generate accurate 3D building models using the data provided. A view of the generated roof structures overlaid with the DSM and point cloud is given in Figure 5.12.

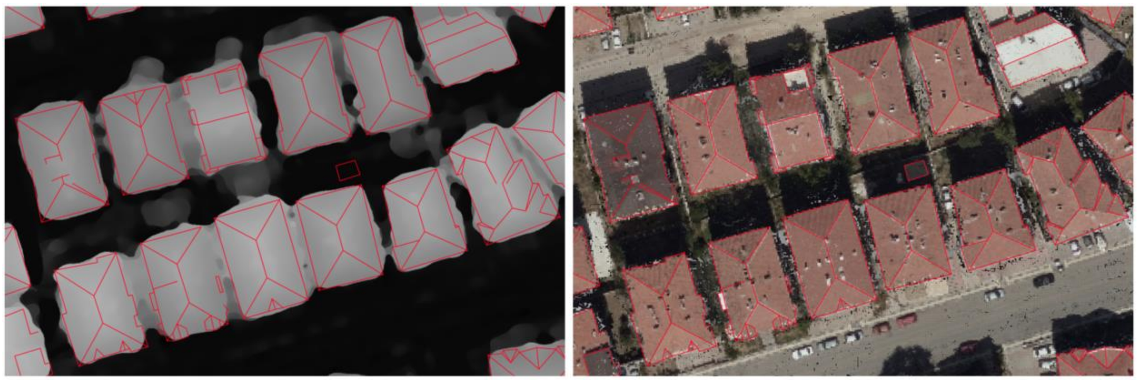


Figure 5.12 Overlayed vector data with raster DSM and photogrammetric point cloud.

Thus, a new DSM is generated based on 3D roof data as ground truth provided by the GDLRC for reconstructing more reliable and visually complete 3D roof models. By utilizing new DSM, it was possible to achieve a better representation of 3D roof models. A close view of generated 3D roofs is given in Figure 5.13.

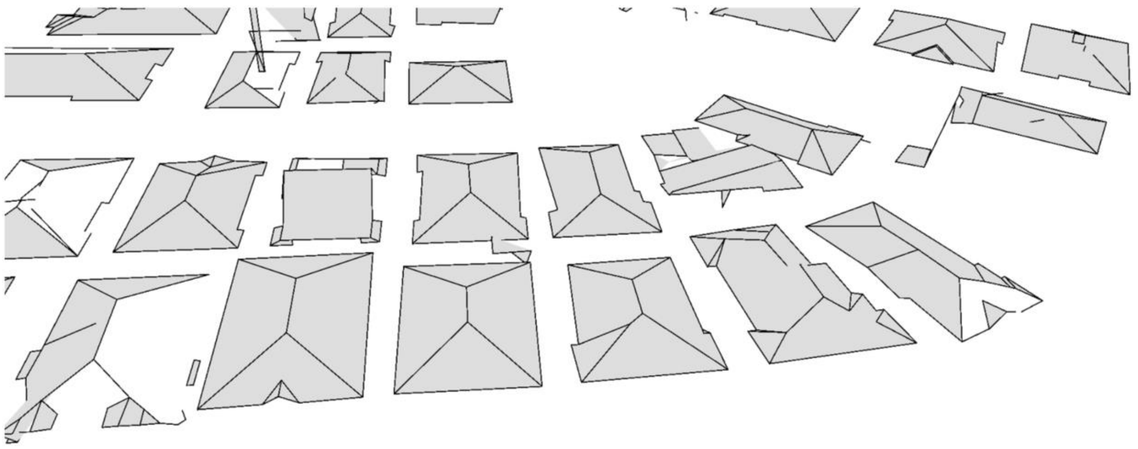


Figure 5.13 Reconstructed 3D roof models using the DSM.

A FME workbench was developed and tested to reconstruct 3D buildings from 3D roof structures. The workbench takes 3D roof geometries and a DTM as input data and returns 3D building models. Firstly, building geometries were converted into polygon geometry from line geometry to define each closed polygon separately. Then, each building polygon was dissolved and the roof structure line is dissected, and only the footprint of the building is left. When building footprints have been generated, each junction point in the building footprint is assigned a new height value, and the junction heights are lowered to fit the overlaying DTM and used as the building floor. The final part of the process is to connect each junction point that overlays each other on top of each other and then reconstruct 3D building model. An overview of the steps is given in Figure 5.14, and view of the reconstructed 3D building from different viewing angles is given in Figure 5.15.

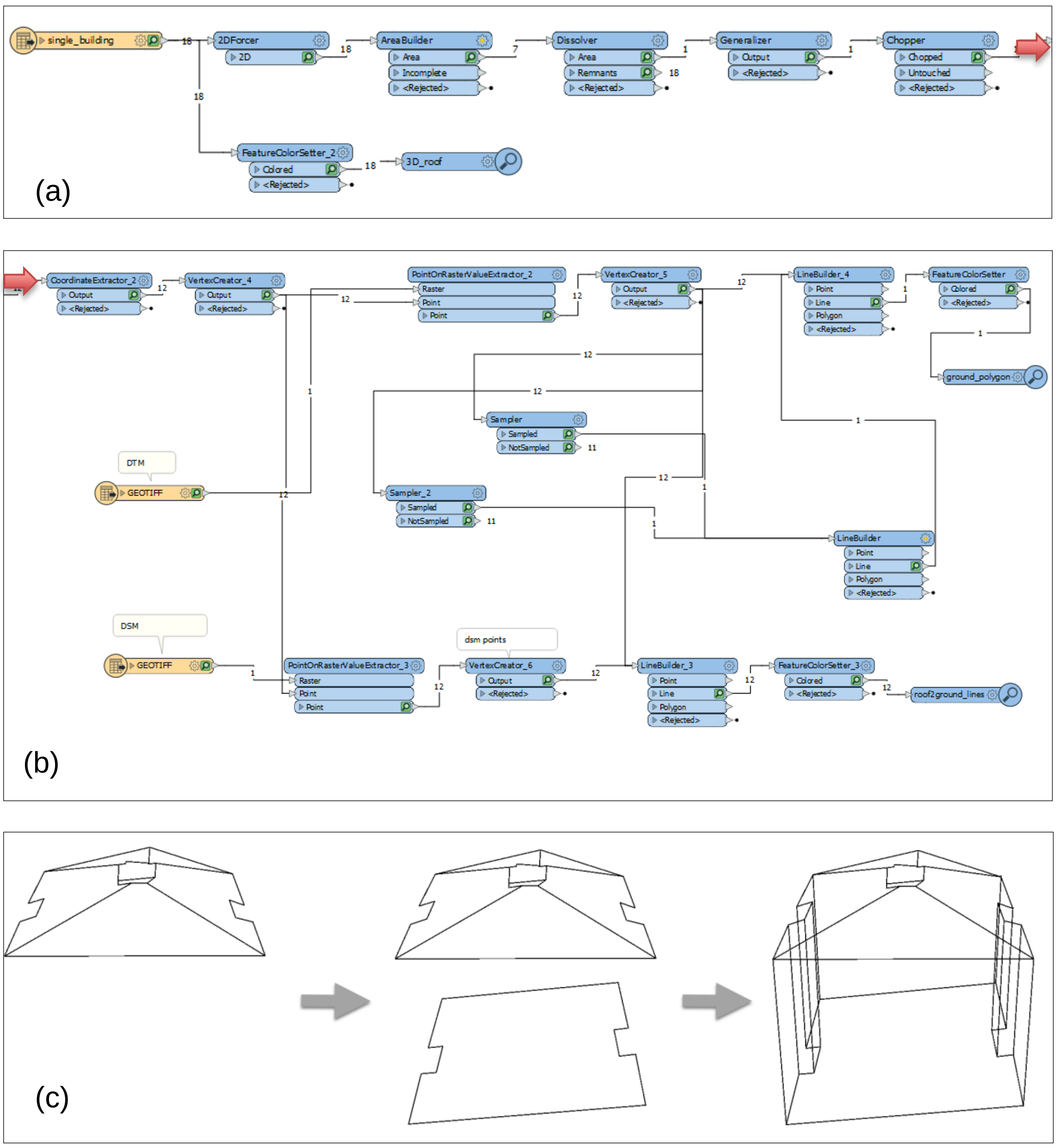


Figure 5.14 Building model reconstruction steps. (a) FME workbench script part 1; (b) FME workbench script part 2; (c) illustration of the stages.

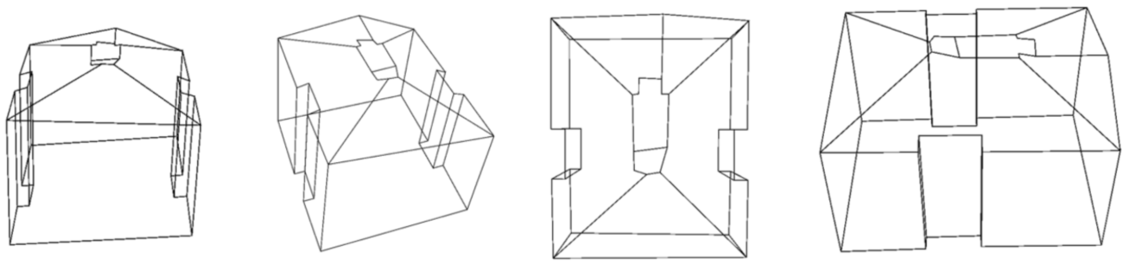


Figure 5.15 Views of a reconstructed building from different viewing angles.

5.7. Discussions and Conclusions on Automated LoD2.2 Building Reconstruction with DL

In this section, a DL-based framework for fully automatic vectorization of LoD2.2 roof structures from very high resolution true orthophotos was presented. Extracted roof structures were then used for 3D building reconstruction with DSM and DTM. Also, first LoD2.2 roof line segmentation dataset for line segment detection networks with more than 139k buildings and 2.2 million lines was presented. In this section, state-of-the-art results were achieved for roof segment detection with the proposed DL-based framework was achieved by 58.4% in msAP and 73.1% in mAP^J. Additionally, the results of this study show that deep learning methods are capable of solving some problems such as roofs that have been blocked by trees or shadows.

When compared with the roof type classification (Chapter 3) and building footprint extraction (Chapter 4) approaches, the fully automatic vectorization of LoD2.2 roof structures provides certain advantages, such as extracting building footprints and roof structures directly as vectors. A vectorization process is not required in this approach, as it is in pixel-based segmentation. Also, this method has shown to be more robust than the other methods considering trees, shadows, and the other elements that cover roofs. A simplification process is usually required to reduce the line complexity after pixel-to-vector conversion. This approach directly produces planar building footprints and roof structures. The outputs of this method can also be used for reconstructing 3D buildings using different height data, such as LiDAR or photogrammetric DSM.

The quality of reconstructed 3D building models can be assesed by using software tools to compare the ground truth model and reconstructed model numerically. This can involve calculating metrics such as volume, surface area, or the number of vertices or faces in the models. Another method is to use a visualization tool to compare the two models visually. This can involve looking for differences in geometry, texture, and other visual features.

There are only a limited number of line segment detection networks available in the literature. The existing networks were not designed to extract building footprints or roof structures, but instead to detect line segments in indoor environments. Therefore, these networks need to be trained from scratch using a roof segment dataset. Currently, a large number of city models are available as open data, but most of them are shared as LoD2. Also, since aerial photos or satellite images of the same region are required along with roof details to train neural networks, there are limited numbers of cities that present these two data together as open data.

Yet, even if data are available, specific input data should be produced for each method based on the input structure of the line segment detection network. It would be very difficult to manually generate training data for thousands of lines and images, so this data should be analyzed and converted into training data with an automatized way compatible with the networks' input data structure.

The processes applied here were automated with scripts, and the existing 3D roof models with true orthophotos are converted into training data for line segment detection networks as a framework. It can be expanded with more training data. It is also planned to expand the framework to extract 3D roof structures from orthophotos and building height information obtained from point clouds or DSMs. Moreover, it is planned to extend the dataset from other cities with different roof types so that the model works worldwide with high accuracy, and is not limited to a specific area.

Combining conventional edge detection methods with deep learning methods may also lead to improved results in tasks such as object recognition, segmentation or line segment detection. Conventional edge detection methods, such as the Canny edge detector, Sobel edge detector, or Laplacian of Gaussian operator, can extract edge features from images with high accuracy. However, they may not be effective in complex scenes or under challenging lighting conditions. On the other hand, CNNs can automatically learn features from raw data, including edges, textures, and shapes, and can capture more complex features that are difficult to detect using conventional methods. However, they require a large amount of labeled training data, which may not always be available. By combining these two methods, strengths of both methods can be leveraged to produce more accurate and robust results. For example, conventional edge detection methods can be used to preprocess images and extract initial edge maps, which are then used as input to CNNs for further processing. This approach can reduce the amount of noise and false positives in the edge maps, and can also help to localize objects more accurately.

6. EFFICIENT VISUALIZATION OF 3D CITY MODELS

This chapter investigates CesiumJS virtual web globe for web-based visualization and Unity game engine with VR support for visualizing 3D city models. Performance assessments were carried out using different visualization technologies for multi-LoD 3D city models. Different optimization techniques were applied to the generated city models for efficient and reliable visualization. In addition, issues experienced during the development of the web-based platform and game engine were analyzed and solutions were proposed. An in-depth analysis of the advantages and disadvantages of each visualization technology is also provided.

6.1. Problem Statement

A 3D city model typically consists of buildings but they are visualized together with a DTM and other city objects like bridges, roads, city furniture, and similar urban features as these models describe the general shape and structure of the city. In addition, semantic information is usually included to perform analyses and queries at a higher level, thus requiring more storage and effort compared to traditional city models. Visualizing thousands of buildings together with basemaps, DTM, and city furniture in a single scene requires optimization for efficient visualization. Also, another critical step for visualization is coordinate system transformations and format conversions. Cesium supports spesific coordinate systems and file formats. DTM or basemap can be visualized in different formats with varying file sizes. Selected data types and format conversions are given in detail here. This section demonstrates of how all city components can be efficiently visualized together using web-based and game engines with VR support. Detailed description of the study area and generated multi-LoD 3D city model can be found in [58]. The web-based model can be visited at www.bizimsehir.org.

6.2. Web-based Visualization of 3D City Models with CesiumJS

Many 3D city model applications require accurate digital terrain model for accurate visualization and improved representation of a city. An underlying DTM enhances the visual quality of 3D city models. Low-resolution DTMs may fail to place buildings well on the terrain, causing them to appear to be inside the terrain or above it. Hence, for the purpose of accurate modeling and preventing visual distortions, it is crucial to have a DTM that is also compatible with the buildings. A number of factors make it necessary to visualize high-resolution DTM alongside 3D city models, including: a) to provide stakeholders with a more realistic visual experience, b) to ensure that earth surface data is complete, c) to improve decision making reliability, and d) the product can be used for a wider range of purposes.

CesiumJS can visualize high-resolution terrain models in different file types and formats, as well as optimize them during streaming. Currently, only a single DTM can be visualized in a scene, which makes it complicated to visualize large regions that have multiple DTMs with varying resolutions. In addition to regular grids with heightmap format, DTMs can also be visualized using TINs with a quantized mesh format. In the Heightmap format, the terrain is represented as regular grids at multiple resolutions. On the other hand, quantized-mesh format pre-renders a TIN mesh in advance for every tile and has the capability of optimizing the mesh for different types of terrain. In contrast to a Heightmap format with a regular spatial distribution, rugged terrain surfaces are visualized more detailed structure.

Streaming and converting high-resolution DTMs were performed on the Cesium ION platform. A DTM must be pre-processed before it can be converted to the terrain file formats supported by Cesium ION. As an example, a single-band raster of DTM must be defined in mean sea level (EMG96) or WGS84 ellipsoid without involving any nodata values. To remove discontinuities in the terrain model and ensure visual completeness, Cesium ION uses high-resolution DTM in conjunction with the Cesium World Terrain. A view of Cesium World Terrain unmerged (left) and merged (right) high-resolution DTM can be seen in Figure 6.1.

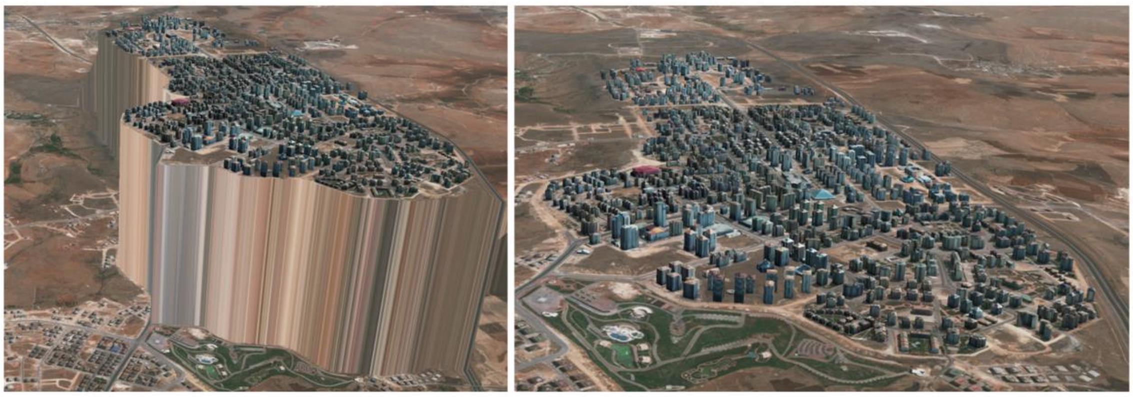


Figure 6.1 A view of Cesium World Terrain unmerged (left) and merged (right) high-resolution DTM

The DTM of the LoD3 city model was pre-processed to align the city plans with the existing components. Visualizing the LoD3 city model using the original DTM would result in visual distortions since the DTM doesn't fit the 3D city plan. Therefore, a new DTM was created that fits to the designs for the project area. LoD3 city model with underlying DTM before and after processing is given in Figure 6.2.

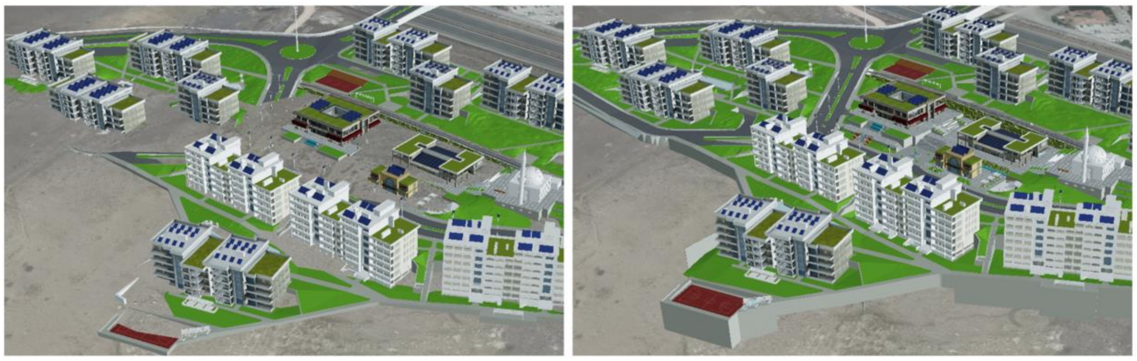


Figure 6.2 LoD3 city model with underlying DTM before and after modification

A 3D city model will be visually complete when it is visualized accompanied by a high-resolution basemap that is visually coherent with the building models. By doing this, users will achieve a realistic impression of the virtual globe, and 3D objects will be positioned accurately on it. The photogrammetric processing workflow with Agisoft Metashape included the production of high-resolution orthophotos (10 cm GSD). Ministry of Urbanization and Environment (MoEU) also produced and provided true orthophotos with the same GSD as part of the Bizimsehir project [58]. True orthophotos are superior to orthophotos because of their reduced distortions, particularly at building edges. A view of orthophoto and trueorthophoto basemaps is given in Figure 6.3.

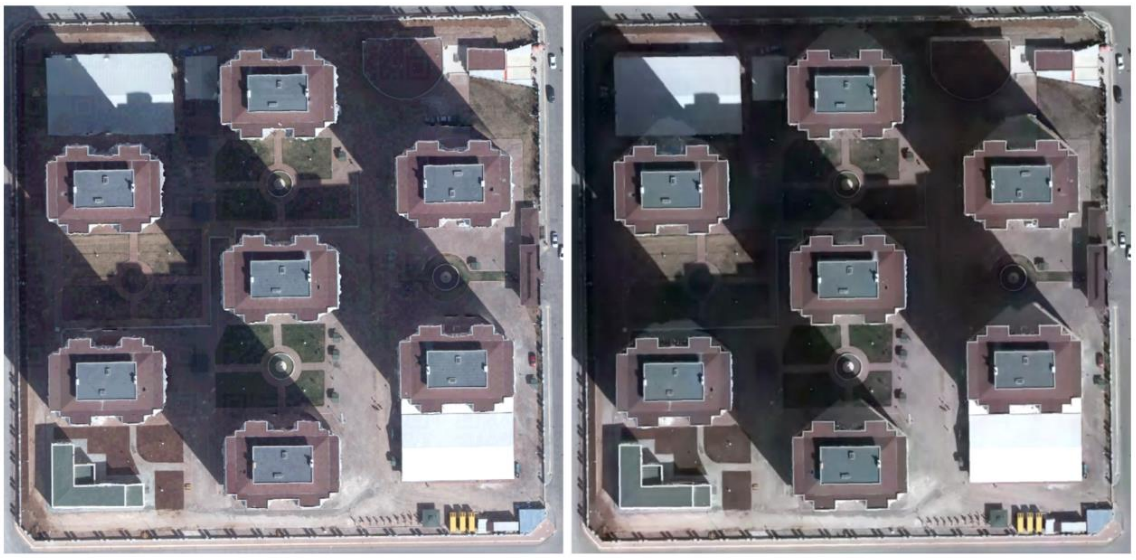


Figure 6.3 A view of orthophoto and true orthophoto basemaps

Georeferenced map tiles are typically served as basemap layers using the OGC standards Tile Map Service (TMS) and Web Map Tile Service (WMTS). As a result of the large file size of high-resolution imagery, more hardware resources are required. Due to this, large basemaps are divided into smaller tiles, which can be streamed based on the angle of the user's view. Several high-resolution imagery providers are supported by CesiumJS, including Cesium ION, Bing Maps, ESRI World Imagery, Mapbox Satellite, and Sentinel-2. It is possible to stream high-resolution imagery or tiled imagery layers from georeferenced raster data using Cesium ION. The Cesium ION platform is capable of creating TMS and WMTS layers from raster imagery uploaded in a variety of formats, including GeoTIFF, Erdas Imagine, JPEG, and PNG.

TMS and WMTS imagery layers were created by converting generated true orthophotos into a single ZIP file and uploading it to Cesium ION. A dataset will usually contain more than one overlapping tile of imagery, so if more than one raster file has multiple overlapping tiles of imagery within it, all rasters must have the same GSD to avoid inconsistencies within the dataset. Generated true orthophoto basemap, high-resolution DTM, and textured LoD2 city model visualized on the CesiumJS virtual globe in a single scene. A final accuracy assessment was conducted by using DTM and building models to compare true orthophoto basemaps with other imagery layers. Figure 6.4 shows a comparison between true orthophoto, Bing Maps, Mapbox Satellite, and ESRI World Imagery as basemap layers.

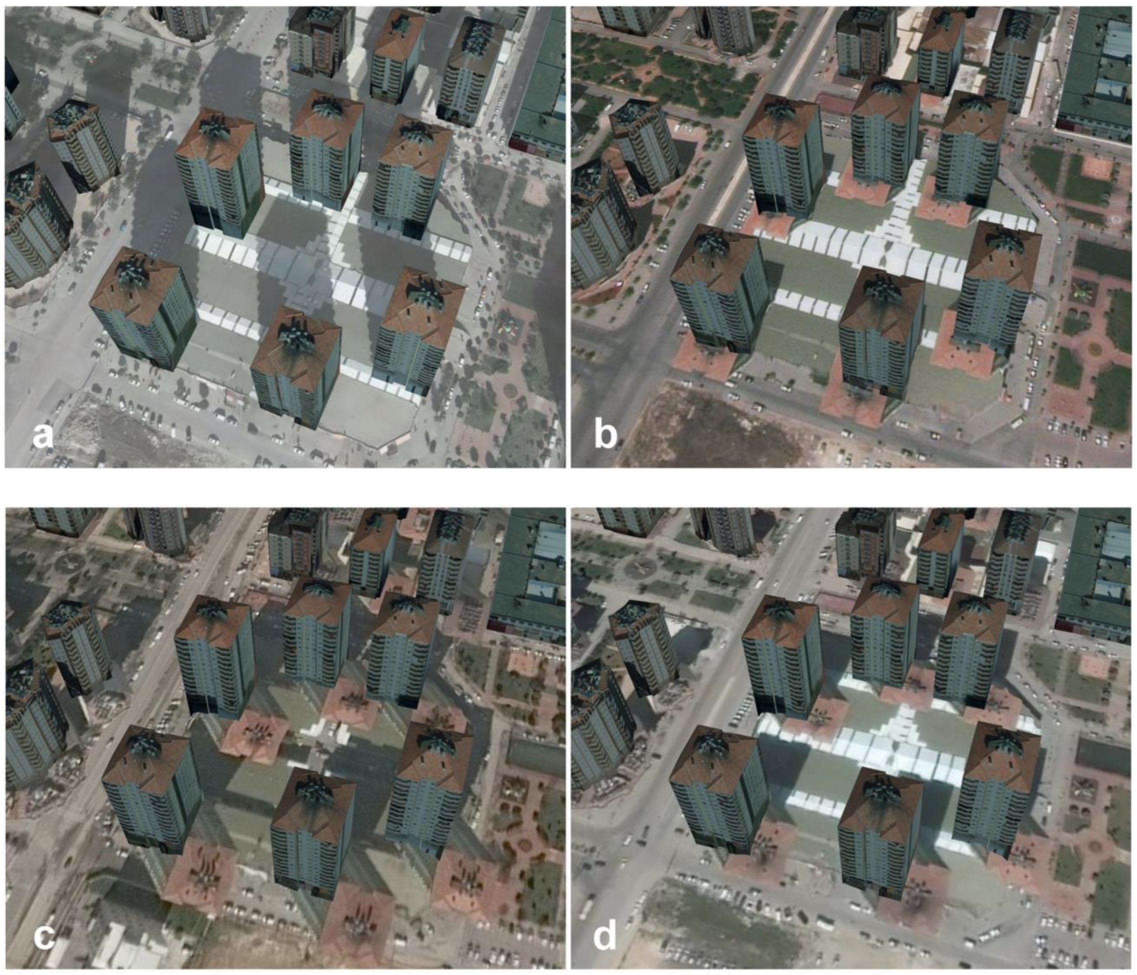


Figure 6.4 A comparison between true orthophoto (a), Bing Maps (b), Mapbox Satellite (c), and ESRI World Imagery (d) as basemap layers

As a last step, 3D city models with different level of details (LoD2 and LoD3) were added to the web scene as the final step of visualization. The LoD2 city model was automatically textured based on high resolution aerial photographs. It will be more difficult for the GPU to handle high-resolution building textures, so texture optimization was needed to reduce the texture size without compromising the visual quality. There are different formats that textures can be stored in, including PNG, JPG, BMP, etc.) with different quality levels.

CityGML supports different texture formats, but JPEG textures were found to be optimal for optimizing texture sizes while maintaining visual quality.

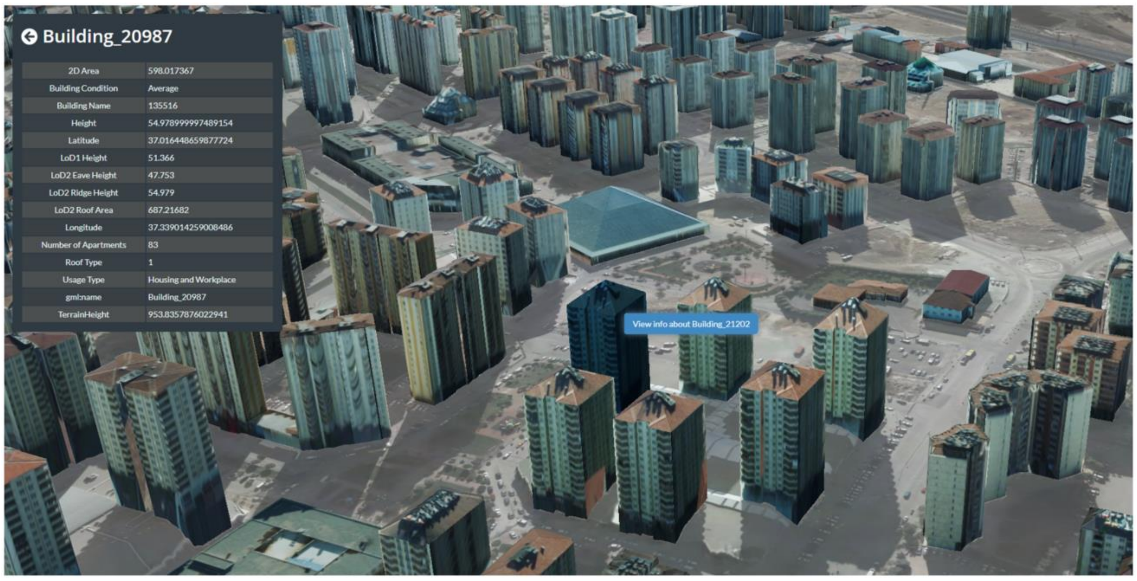
Models of buildings and furniture for the LoD3 city model, along with city plans and textures, were all stored separately in CityGML files. It is important to optimize CityGML and building textures in order to achieve efficient visualization. The texture patch for each façade of the building is created by clipping images from aerial pictures that are used in CityGML. In consequence, for a couple of buildings, CityGML files can easily consist of hundreds of megabytes of textures. CityGML files contained redundant texture data, and when those files were stored separately, they added to the size of the files as well as created duplicate textures. This problem was resolved by merging files with shared textures into a single CityGML, and then removing duplicate textures as the first step towards optimizing textures. Buildings and city furniture were generated as separate CityGML files along with their textures, then merged into a single file.

The use of a single texture for the whole building, rather than separate textures for each façade, reduces hardware usage and increases visualization efficiency significantly. A comparison of textures before and after optimization is given in Table 6.1.

LoD2 Models PNG Textures	**LoD2 Models JPEG Textures**	**City Plan and Furniture (Before)**	**City Plan and Furniture (After)**	**LoD3 Buildings (Before)**	**LoD3 Buildings (After)**
15454 files	15454 files	608 files	33 files	3847 files	154 files
709 MB	100 MB	1233 MB	83.1 MB	568 MB	94.4 MB

Table 6.1 Comparison of texture sizes before and after optimization [58].

3D city models, city plans and city furniture in CityGML format converted to 3DTiles using Cesium ION. The CityGML format is capable of storing the semantics and the properties of each building. Semantics in CityGML can be incorporated into 3D tiles for each building. LoD2 city model with enriched semantics is visualized in a single scene together with LoD3 city model. LoD2 model contains semantics such as 2D area, roof type, usage and other features. LoD3 city model contains only automatically generated semantics such as latitude, longitude and building height. It is possible to see the semantics of the building by selecting a building on the developed web interface. Each building can be selected separately and some queries such as measuring heights or areas can be performed in the developed web interface. It's possible to style or hide buildings according to their attributes. A view of the web interface that shows attributes for a LoD2 building is shown in Figure 6.5. LoD3 city model with 3D city plans is given in Figure 6.6.


Building_20987

2D Area	598.017367
Building Condition	Average
Building Name	135516
Height	54.978999997489154
Latitude	37.016448659877724
LoD1 Height	51.366
LoD2 Eave Height	47.753
LoD2 Ridge Height	54.979
LoD2 Roof Area	687.21682
Longitude	37.339014259008486
Number of Apartments	83
Roof Type	1
Usage Type	Housing and Workplace
gml:name	Building_20987
TerrainHeight	953.8357876022941

Figure 6.5 A view of attributes for selected LoD2 building in the web interface

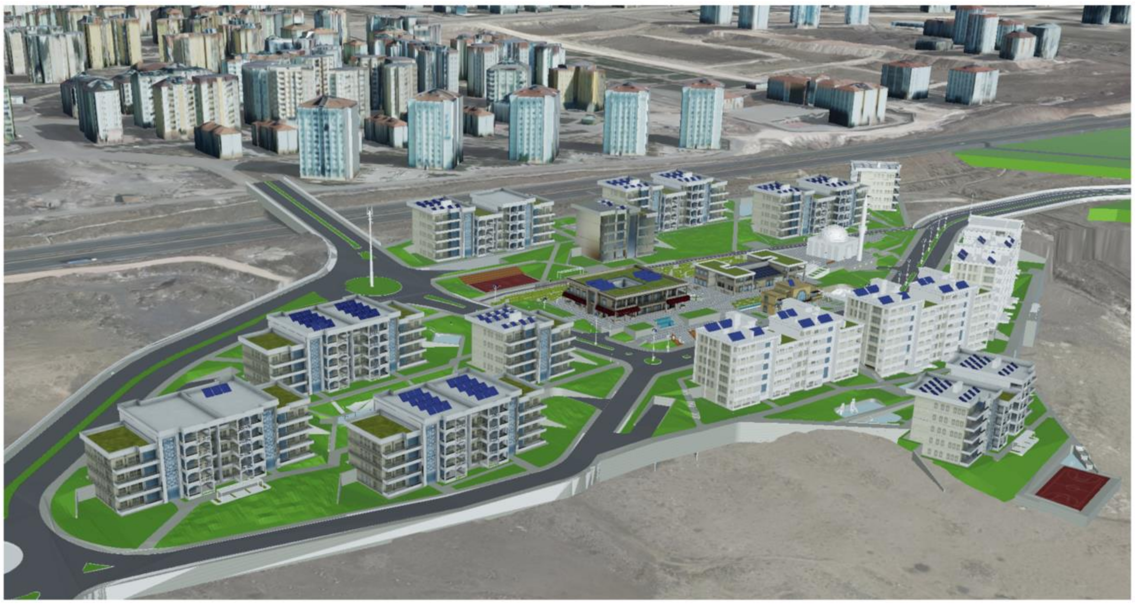


Figure 6.6 LoD3 city model with 3D city plans in the web interface.

6.3. 4D Data Visualization with CZML

In this subsection, a geo-visualization interface for UAV and airplane photogrammetric flights using CesiumJS Virtual Globe has been developed for 4D data visualization. Image trajectory elements such as camera rotations and image perspective center coordinates measured during the flight were used for 4D visualization approach. Exterior orientation parameters (EOPs) and the interior orientation parameters (IOPs) of the images were used to visualize aircraft's flight path in time, camera position, and the image footprints on the ground, as well as the airplane's rotation. CesiumJS virtual globe is used to visualize the flight and footprint of the images captured during the flight. A 4D visualization of the photogrammetric image acquisition has been created based on the data obtained from airplane and unmanned aerial vehicle (UAV) platforms during the photogrammetric image acquisition process.

Flight paths reconstructed from image metadata, known as EOPs, and acquisition times. The time difference between two image acquisitions is calculated and used to animate image acquisitions precisely. Simulating aircraft rotations during flight was conducted using the roll, pitch, and yaw angles from EOPs. Considering CesiumJS only supports WGS84 coordinate system, the image perspective center coordinates in EOPs have been reprojected from the Universal Transverse Mercator (UTM) projection system to the WGS84 system for the visualization.

The image footprint vectors are generated by using a custom FME workbench that uses the image EOPs and IOPs and assumes that the imaging conditions are nadir. Figure 6.7 gives an overview of the developed custom FME workbench. The perspective center of the images was retrieved from EOPs in CSV format (Comma-separated Value) as first step. "VertexCreator" transformer is used to create point features using image perspective center coordinates. "Bufferer" transformer is used to generate image footprints polygons around the points using a dummy size of 1 m x 1 m as a reference. Sensor size (image width and height) and nominal GSD (Ground Sampling Distance) are used to simulate the footprint of image on the terrain. As an example, UAV cameras with image sizes of 6000x4000 pixels and a 5 cm GSD have image footprints of 300m x 200m on the ground.

In the next step, image footprint polygons are generated by resizing generated dummy polygons for each image reference footprint polygon using the "Scaler" transformer.

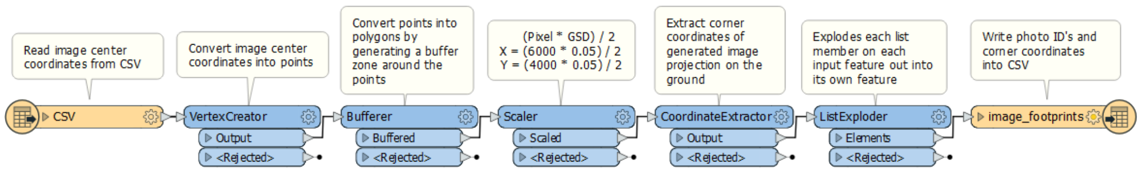

Read image center coordinates from CSV
Convert image center coordinates into points
Convert points into polygons by generating a buffer zone around the points
(Pixel * GSD) / 2
X = (6000 * 0.05) / 2
Y = (4000 * 0.05) / 2
Extract corner coordinates of generated image projection on the ground
Explodes each list member on each input feature out into its own feature
Write photo ID's and corner coordinates into CSV
CSV
VertexCreator
Output
<Rejected>
Bufferer
Buffered
<Rejected>
Scaler
Scaled
<Rejected>
CoordinateExtractor
Output
<Rejected>
ListExploder
Elements
<Rejected>
image_footprints

Figure 6.7 Developed FME workbench for generating image footprints.

As a final step in the process, the coordinate values of image footprints on the ground are calculated with "CoordinateExtractor" transformer. For each image footprint, the corner coordinates and image IDs are exported for intermediate processing for conversion to CZML. Figure 6.8 shows an example of visualization of an image footprint on the ground. As mentioned previously, a constant GSD value was used for all images, and the images were taken in the nadir direction for the entire flight.

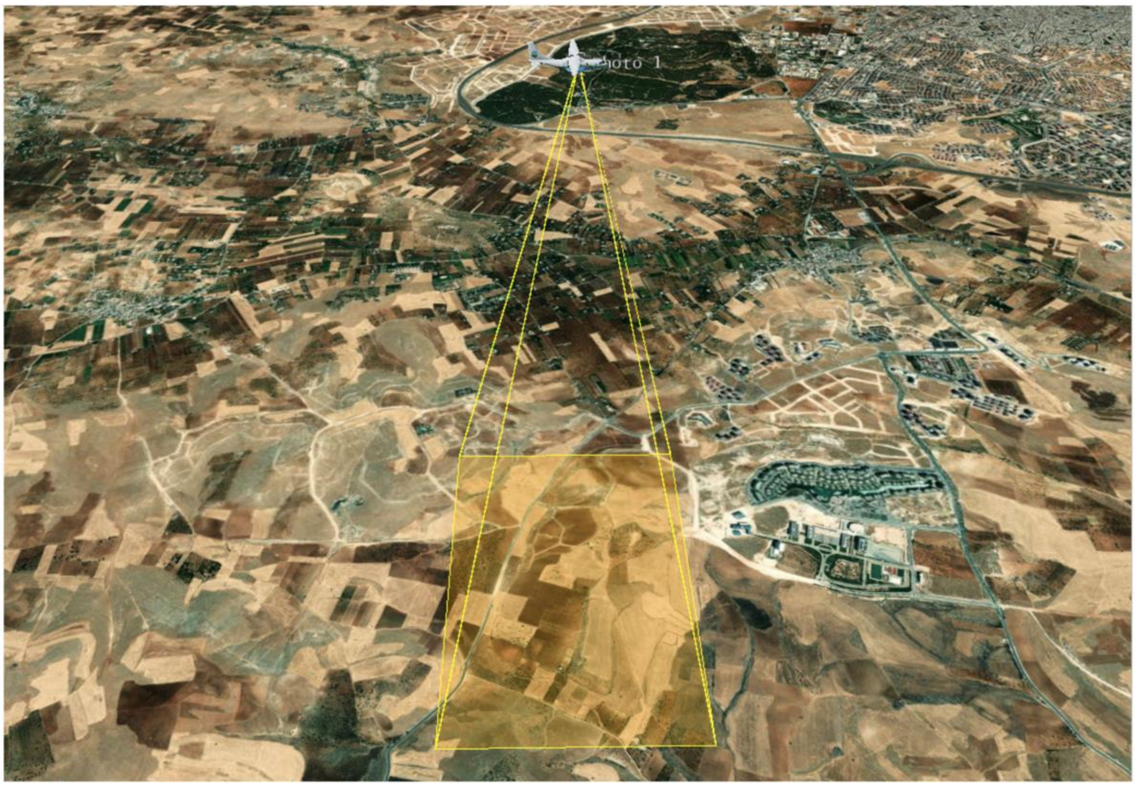


Figure 6.8 An example of visualization of an image footprint on the ground.

Geospatial data and 3D models can be visualized using CZML in a time-dynamic (four-dimensional) environment using CesiumJS. Georeferenced data types can be visualized on Cesium Virtual Globes using CZML, including 3D models as 3D Tiles, as well as geometric shapes.

The animated 3D models are rendered in the glTF format. Depending on the type of photogrammetric flight, such as an aerial photogrammetric flight or an UAV, visualization can be performed using an animated 3D airplane or UAV. Since glTF uses quaternions (x, y, z, w) to animate the rotations, aircraft rotations from EOPs are converted to quaternions. Cesium's built-in capabilities help visualize aircraft rotations by first converting the heading, pitch, and roll values of the aircraft from degrees to radians, and then using the radians to quaternions conversion tool to convert the radians to quaternions. The web interface also features a camera icon for each location of the image perspective center. These icons can be interacted with to view image thumbnails and to see information about that image such as ID, coordinates, acquisition time, and EOPs of selected image. Aircraft models included in the default Cesium model library (airplanes and UAVs) used in the study can be seen in Figure 6.9. As a last step, flight data, including flight path, aircraft rotation, image footprints, and flight speed derived from distance and interval information, then converted into a single CZML (Cesium Language) file for visualization on the CesiumJS web globe.

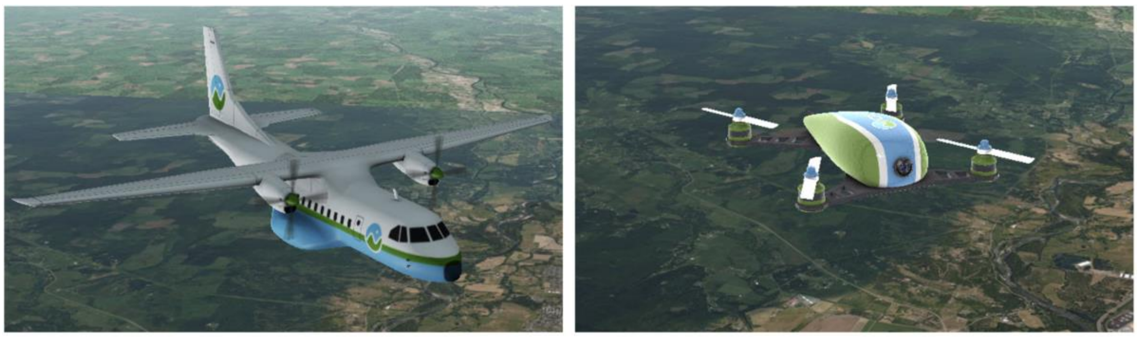


Figure 6.9 Airplane and UAV model.

There are many different visualization projects that can be used with CZML. CZML format does not have a static template, so CZML templates should therefore be created for each project, so that they meet the specific requirements of that project. Visualizing CZML data can be accomplished in a variety of ways. In addition to uploading and visualizing generated CZML data online with Cesium ION, Cesium Sandcastle provides the capability of visualizing generated data locally as well. The Cesium Sandcastle app is both a live-coding app as well as a geovisualization interface application for viewing CesiumJS examples in a local environment or online. It is possible to visualize a CZML file either on Cesium Sandcastle by manually coding it, or by uploading it to Cesium ION as a CZML file. Cesium Sandcastle allows users to to share developed CesiumJS projects with other Cesium Sandcastle users with share option, however, it is not a suitable tool for sharing data and projects between users.

Geospatial data can be uploaded and visualized more easily Cesium ION platform, which is designed to handle large volumes of 3D geospatial data, in addition to hosting, optimizing, and streaming them. Through Cesium ION, users have access to Cesium World Terrain, several imagery layers, and the ability to convert geospatial datasets into desired format for efficient visualization. As part of this thesis, generated CZML dataset and 3D model are also visualized and published online with Cesium ION. A view of photogrammetric image acquisition visualization with an UAV with an 80% forward overlap is given in Figure 6.10. Figure 6.11 shows photogrammetric image acquisition of 736 images in Gaziantep, Turkey with an 80% forward and 60% lateral overlap.

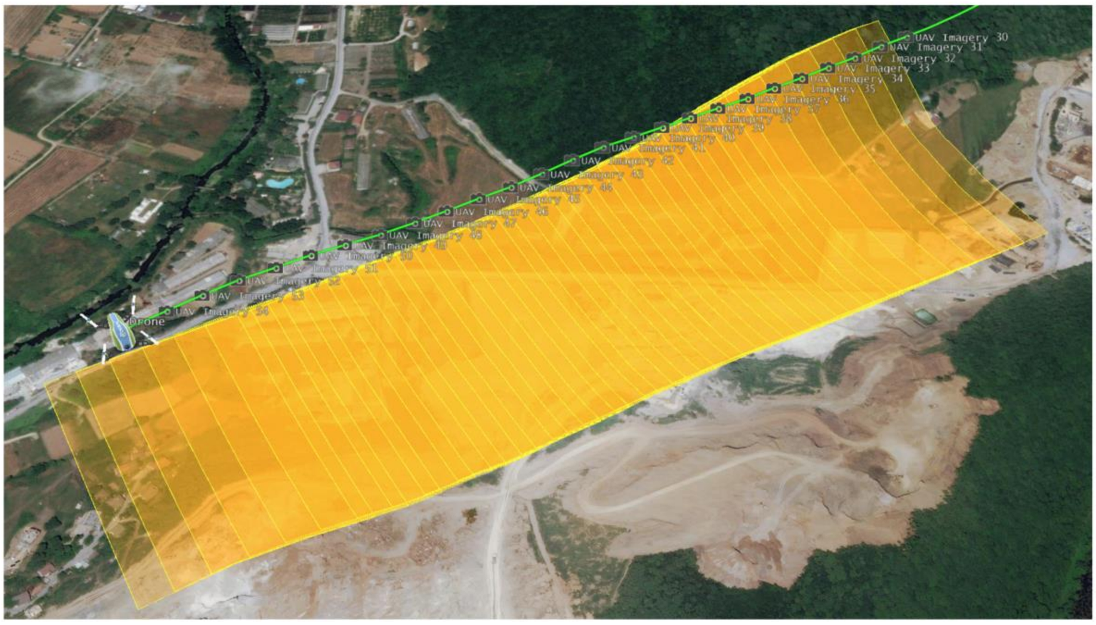


Figure 6.10 Photogrammetric image acquisition visualization with an UAV with an 80% forward overlap [58].

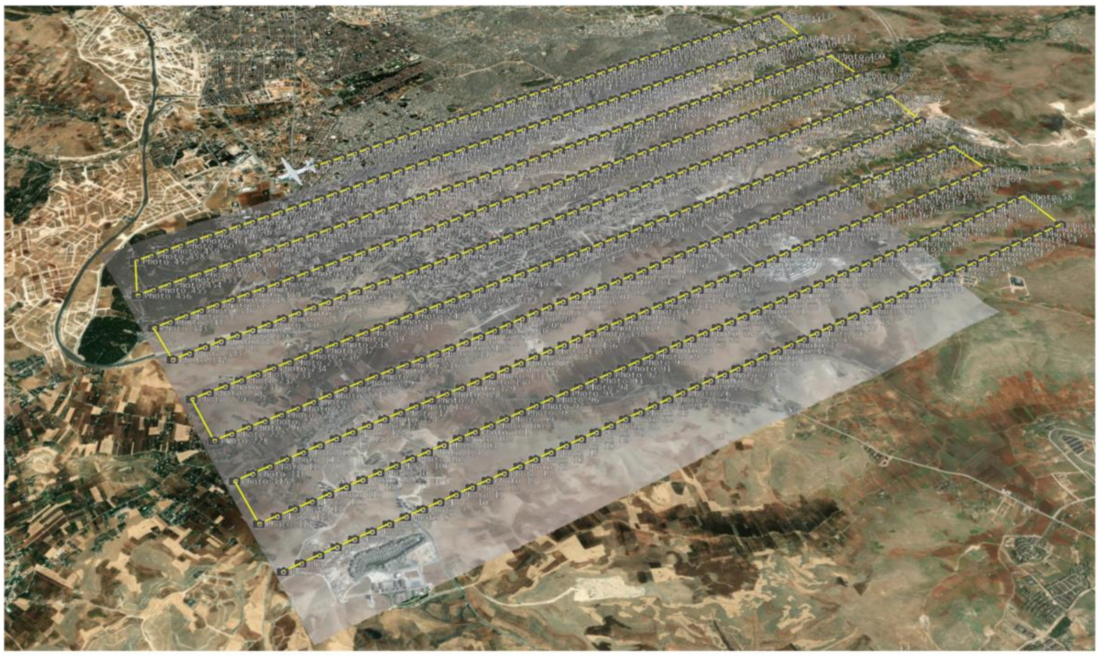


Figure 6.11 Photogrammetric image acquisition of 736 images in Gaziantep, Turkey with an 80% forward and 60% lateral overlap [58].

Users may identify errors in aerial photogrammetry flights with the help of a 4D visualization. This will enable them to prevent similar errors in future missions. It is thus possible to gain a better understanding of the definitions of photogrammetric terms, such as image overlap, image resolution, flying height, and flight route by visualizing photogrammetric flights. Furthermore, visualizing photogrammetric flights allows for the analysis of the geometrical errors in the acquisition of images even in adverse weather conditions such as rain and wind.

6.4. Game Engines and Virtual Reality

This section presents an approach to visualizing LoD3 city models in the Unity game engine with VR support. Virtual Reality have become increasingly important in a wide range of fields, including urban planning, architecture, civil engineering, and even entertainment. Exploring 3D city models in VR can provide a wealth of information and insights that are difficult to obtain from traditional 2D maps and models. They allow urban planners and architects to create and visualize new designs and proposals in a virtual space before actually building them, which can save time, money, and resources. Civil engineers can use VR to simulate the behavior of infrastructure and transportation systems, helping them to identify potential problems and optimize designs. In addition, virtual 3D city models can be used in education and outreach, allowing the public to better understand and engage with urban planning and development.

Game engine visualization of 3D city models requires a different approach than web-based visualization. Most game engines rely heavily on processors and graphics cards (GPUs). Various strategies can be used in game engine optimization to reduce CPU and GPU usage. The scene can be optimized by loading only the models in the scene that are visible in that point of view. Texture and polygon count are also important aspects of optimization. The scene should provide the user with as realistic a presentation as possible with the fewest number of polygons.

The LoD3 city model, city plans, and city furniture were created in a CAD environment and exported to Unity in the scope of this study. For models with high polygon counts, polygon optimization has been performed before exporting them to the game engine while maintaining their visual quality. Since CAD environments generally do not support projected coordinates, the entire city model must be manually reprojected from local coordinates to projected coordinates. Buildings and objects were manually placed on the reprojected city plan during this process. Sound effects and animations were added to the scene to enhance its realism. The final step is to combine elements of the city, such as its buildings, city plan, and furniture, and present them in VR environment.

Although exploring 3D city models with virtual reality offers a much more realistic experience than web-based visualization, various hardware is required to use this technology. Exploring generated scenes in a VR environment requires both a VR headset and a VR-ready GPU. The limited accessibility of this hardware makes VR exploration of 3D city models possible only for users with required equipment.

Data that visualized in the previous chapter using CesiumJS was then visualized using Unity game engine in a more detailed scene. True orthophoto basemaps, high-resolution DTM and generated 3D city models including buildings, ground plans, and city furniture are visualized together in a single scene in Unity game engine to obtain the most accurate visualization possible. A view of the LoD3 city model in Unity game engine from different viewpoints can be seen in Figure 6.12.

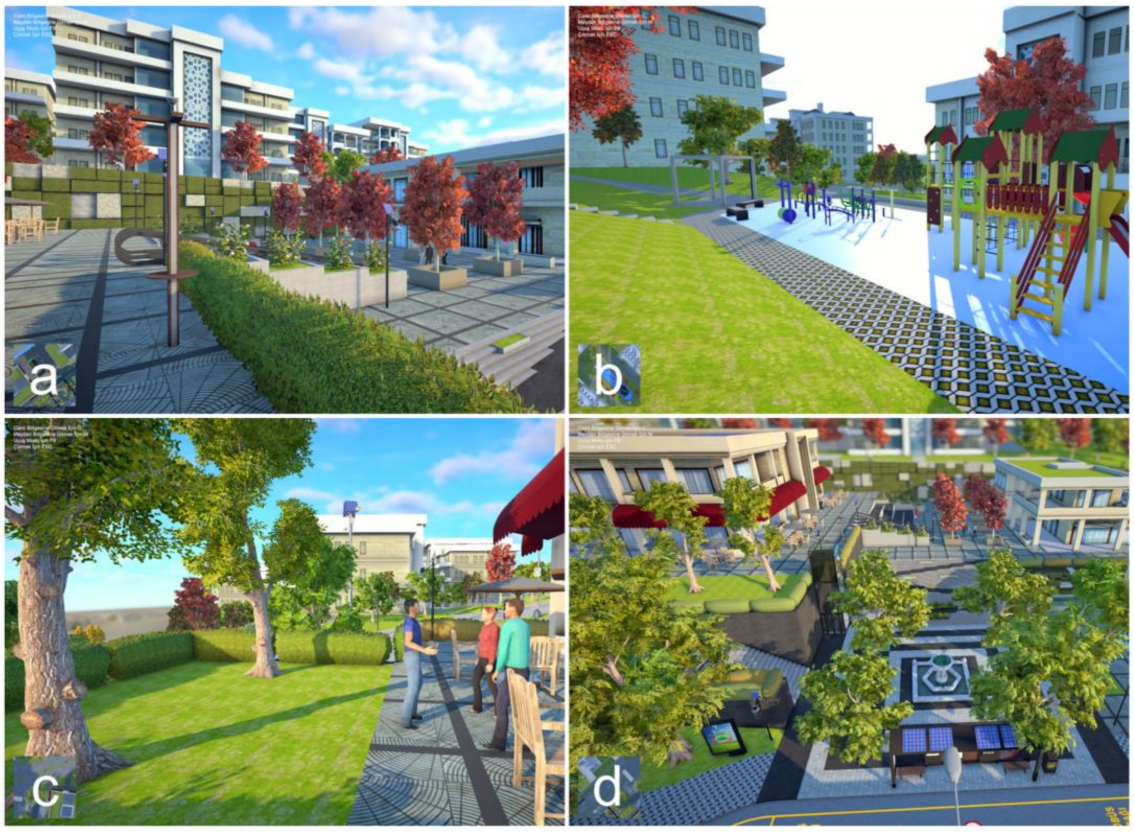


Figure 6.12 A view of the LoD3 city model in Unity game engine from different viewpoints [137].

6.5. Discussions and Conclusions on Efficient Visualization of 3D City Models

Web-based visualization and game engine visualization are two completely different visualization platforms that requires different visualization optimizations. CesiumJS provides an efficient way to visualize geospatial data by WebGL-based rendering, dynamic level of detail and automatizing format conversion using Cesium ION. On the other hand, game engines and VR offers more realistic experience for users. Game engines and VR allow users to experience and explore the city in a more engaging way, leading to a better understanding of the city's features and characteristics.

Users can experience cities in a more realistic way with virtual reality by being more involved in the scene. Visitors can visit virtual reality cities in the design phase before they are constructed. Future cities can be visualized and explored by stakeholders in VR environment. The VR environment also allows users to interact with objects and explore the city at street level. As a result, users can give more accurate feedback and contribute to the city's development more effectively. As a result, the city's inhabitants, as well as architects and city planners, can contribute to the design of the city. The use of VR in urban development can therefore enhance the city in a variety of ways. Virtual reality also allows users to explore cities virtually from around the world.

There are many advantages to visualizing 3D city models in game engines. It is possible to visualize 3D city models that are as realistic as possible by using game engines. This allows users to experience a realistic environment. The details of the architecture of buildings, vehicles on the streets and the movements of people in a virtual 3D city model provide a realistic experience in a game engine. Also, game engines provide users with an interactive way to view and interact with cities and their details. Through the game engine, users are able to touch, move and interact with city objects. Game engines allow easy customization of virtual city models. City details, content, and appearance can be modified by users. Therefore, virtual city models can be created for different scenarios in this way.

7. CONCLUSIONS AND FUTURE WORK

In this section, conclusions and future work are presented in two separate sections based on the studies that were conducted under the scope of the thesis. The conclusion section contains the main findings, the critical analysis of the results, what is really novel, where progress has been made and if the project objectives have been achieved. The future work section contains directions for future research and potential improvements for each study.

7.1. Conclusions

In this thesis, the three main stages of 3D city modeling, i.e., data acquisition and pre-processing, building extraction and modeling, and visualization, were investigated and several optimization solutions were provided for each stage. Regarding the data acqusition and pre-processing, aerial photogrammetry was preferred due to the provision of both visible images and 3D geometries. For building extraction and modeling, different approaches such as roof type classification for library-based model production, building footprint segmentation with CNNs, and roof line extraction with line detection segmentation networks were investigated. For the visualization, web globes and game engine based VR were analyzed.

Many applications use building footprints to extract or clip buildings from images or DSM. The F1-Score and IoU (IoU) improved by 3.27% and 5.90% when nDSM data was fused as a fourth band to RGB data. There have been some improvements in the results that have been obtained by using nDSM from roofs that are obscured by objects, such as shadows and trees, but this problem is still not completely resolved. Upon conversion of predicted building footprints from pixel-based segmentation to vector data, a smoothing process was performed on the vector data for correcting the un-smoothness of the vector data. In some cases, other objects with a color value that is close to the roof color may be classified as buildings as well. This study was conducted in a complex and unstructured area of roof buildings but at the same time promising results were obtained, despite the complexity of the study area. There is one disadvantage of this method of dividing buildings under one roof, which is that it is impossible to distinguish the building lines when there are more than one building under the same roof.

Roof types are very important due to several reasons. The type of roof can significantly impact the energy efficiency of a building. Different roof types have different load-bearing capacities and can withstand different levels of wind, snow, and other environmental factors. Understanding the roof type is crucial for assessing the structural stability of a building. A dataset containing 10.000 unique roof images for 6 different types of roof was introduced in the roof type classification section. DL-based roof type classification was achieved with a very high F1-score with 86% as the best result. A more comprehensive analysis of the results indicates that roof types with more data were more likely to be better learned by the model and achieved higher classification scores than those with less data. Despite the fact that the "complex" roof type has the most data among all the roof types in the dataset, it was not classified with the highest level of accuracy since it does not have a standard geometric structure. There are many different areas in which roof type information can be used, such as procedural building reconstruction and solar energy applications.

As a part of the thesis, a DL-based framework for fully automatic vectorization of LoD2.2 roof structures in city-scale. First LoD2.2 roof line segmentation dataset for line segment detection networks with more than 139 k buildings and 2.2 million lines was also introduced. DL-based method have achieved state-of-the-art results for roof segment detection by 58.4% in msAP and 73.1% in mAP^J. Additionally, the results of this study show that deep learning methods are capable of solving some problems such as roofs that have been blocked by trees or shadows.

Generally, extracting roof structure from RGB images is a more reliable method than extracting roof structure from LiDAR or photogrammetric point clouds. Since only VHR stereo aerial imagery is used as primary data source for this thesis, 3D buildings are reconstructed using predicted roof structures from RGB images and photogrammetric DSM. In spite of the fact that a high level of accuracy was obtained from roof structure extraction, 3D building reconstruction based on heights from photogrammetry-based DSM did not turn out to be satisfactory. Since photogrammetry-based DSMs do not present heights to the same degree of accuracy as LiDAR point clouds, the use of heights obtained from photogrammetry-based DSM is likely to lead to incorrect 3D roof reconstruction. This means that using stereo aerial imagery and LiDAR data together will provide a more accurate way to generate the geometry of a 3D roof and building.

The primary objective of efficient visualization is to present the 3D city data in a way that is clear, concise, and easily understandable, while preserving the accuracy and detail of the original model. This may involve the use of visualization techniques such as 3D rendering, interactive navigation, or virtual reality, to create an immersive experience that allows stakeholders and citizens to explore and interract with the city data in a meaningful way. The ultimate objective of efficient visualization of 3D city models is to provide a powerful tool for urban planning, analysis, and decision-making.

As a part of thesis, visualization of multi-LoD 3D city models in different platforms and investigated different performance optimization procedures. Web-based visualization and game engine visualization require completely different approaches. The visual representation of city models should be aimed at maintaining the highest possible level of visual detail while at the same time minimizing the computer hardware load. A more efficient approach would be to provide the details of the building or city through textures rather than using more polygons for the models. Based on the results of a research, the web-based visualization library CesiumJS is a suitable solution for displaying 3D city models along with basemaps and DTMs from different geospatial datasets. Game engine visualization of 3D city models offers a much more realistic experience than web-based visualizations. By visualizing city models in VR and presenting them before the reconstruction of the city, citizens and stakeholders can take part in making decisions about the future of the city.

As a result of its intuitive interface and web-based system, these systems are not only used by professionals, but also by the general public. Since CesiumJS provides support for a variety of geospatial data, including DTM, basemaps, and textured 3D city models, it is currently the most suitable open-source web globe for visualizing 3D city models and developing WebGIS interfaces. CesiumJS library can be customized to meet the requirements of the project. This thesis has also lists the following additional useful features of Cesium; the ability to customize the interface; the ability to stream, style, and interact with 3D tiles; support for visualizing DTMs in different resolutions; information boxes for selecting objects and displaying attributes, and the ability to optimize GPU, CPU, and power utilization. Unreal and Unity game engines can import geospatial datasets hosted in Cesium ION without data conversion or processing.

7.2. Future Work

For building footprint extraction, line segment detection networks can be used to extract building boundaries directly as vectors without having to first perform pixel-based binary segmentation and then vectorization. This means that instead of two different processes, building boundaries can be retrieved in a vector format in a single step. In such a case, if there are multiple buildings under a single roof, line segment detection networks can also be used to detect the roof boundaries. Since the dataset is a relatively small dataset, it may be possible to expand the dataset by using photogrammetric data from other cities or by using data augmentation techniques to generate new data from existing dataset. Each band's segmentation performance can be evaluated separately, then bands that reduce model accuracy can be excluded. Building bounding boxes can also be detected using object detection algorithms such as YOLO or using YOLO together with SAHI.

The roof structure extraction can also be improved if the height data is derived from LiDAR point clouds instead of photogrammetric DSM data, as this will allow a better reconstruction of 3D buildings and roofs. Through the use of DL-based height prediction methods, it is possible to directly extract roof structures in 3D. With the addition of height estimation and edge detection methods to the framework, and the use of line segment detection networks, roof segment detection accuracy can be further improved. The dataset can also be extended using data from other cities with different types of roofs. This will enable the model to be adapted by globally. It is possible to improve the post-processing algorithm even further in order to provide better results from the predicted roof segments.

As future efforts, the classification of roof types can be extended to include roof types other than the 6 roof types that are used in the generated dataset. It should be noted that since the dataset used here is a relatively small dataset, it could be expanded by using roofs from different cities or by augmenting the data. In addition to the CNNs used in this thesis, other CNNs can be used for roof type classification task. It is possible to detect and classify roofs directly from aerial photographs or satellite images with DL-based object detection. This eliminates the need to clip each building from orthophotos with building footprints. It is possible to tune hyperparameters such as batch size and learning rate by experimenting with different values to determine the optimal value for achieving best results. There is also the possibility of improving classification results by adding height information (DSM) as a fourth band to the RGB images.

As a future step, the web interface developed during this project can be enhanced by adapting the models to a 3D GIS platform or a database. By enriching the semantic data of the building models, more analyses can be conducted on the models in city-scale. The Cesium ION platform is able to visualize and analyze 3D geospatial data in a number of different ways on the web, as well as in the Unity and Unreal game engines. The created scenes can be enhanced visually through the use of game engines, while also providing a more realistic experience through the use of sound effects and animations.

REFERENCES

[1] Biljecki, F., H. Ledoux, and J. Stoter, An improved LoD specification for 3D building models. Computers, Environment and Urban Systems. 59: p. 25-37, **2016**

[2] OpenStreetMap, https://www.openstreetmap.org/, Last Access: 27.03.2022

[3] Microsoft Building Footprints, https://www.microsoft.com/en-us/maps/building-footprints, Last Access: 27.03.2022

[4] Heipke, C. and F. Rottensteiner, Deep learning for geometric and semantic tasks in photogrammetry and remote sensing. Geo-spatial Information Science. 23(1): p. 10-19, **2020**

[5] Sezgin, M. and B.l. Sankur, Survey over image thresholding techniques and quantitative performance evaluation. Journal of Electronic imaging. 13(1): p. 146-168, **2004**

[6] Maini, R. and H. Aggarwal, Study and comparison of various image edge detection techniques. International journal of image processing (IJIP). 3(1): p. 1-11, **2009**

[7] Hojjatoleslami, S. and J. Kittler, Region growing: a new approach. IEEE Transactions on Image processing. 7(7): p. 1079-1084, **1998**

[8] Illingworth, J. and J. Kittler, A survey of the Hough transform. Computer vision, graphics, and image processing. 44(1): p. 87-116, **1988**

[9] Lewis, J.P. Fast template matching. in Vision interface. 1995. Quebec City, QC, Canada.

[10] Alidoost, F., H. Arefi, and F. Tombari, 2D image-to-3D model: Knowledge-based 3D building reconstruction (3DBR) using single aerial images and convolutional neural networks (CNNs). Remote Sensing. 11(19): p. 2219, **2019**

[11] Partovi, T., F. Fraundorfer, S. Azimi, D. Marmanis, and P. Reinartz, Roof Type Selection based on patch-based classsification using deep learning for high Resolution Satellite Imagery. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences-ISPRS Archives. 42(W1): p. 653-657, **2017**

[12] Axelsson, M., U. Soderman, A. Berg, and T. Lithen. Roof type classification using deep convolutional neural networks on low resolution photogrammetric point clouds from aerial imagery. in 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP). 2018. IEEE.

[13] Alidoost, F. and H. Arefi, KNOWLEDGE BASED 3D BUILDING MODEL RECOGNITION USING CONVOLUTIONAL NEURAL NETWORKS FROM LIDAR AND AERIAL IMAGERIES. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences. 41, **2016**

[14] Mohajeri, N., D. Assouline, B. Guiboud, A. Bill, A. Gudmundsson, and J.-L. Scartezzini, A city-scale roof shape classification using machine learning for solar energy applications. Renewable Energy. 121: p. 81-93, **2018**

[15] Qin, Y., Y. Wu, B. Li, S. Gao, M. Liu, and Y. Zhan, Semantic segmentation of building roof in dense urban environment with deep convolutional neural network: A case study using GF2 VHR imagery in China. Sensors. 19(5): p. 1164, **2019**

[16] Ölçer, N., D. Ölçer, and E. Sümer, Roof type classification with innovative machine learning approaches. PeerJ Computer Science, **2023**

[17] Bittner, K., M. Körner, F. Fraundorfer, and P. Reinartz, Multi-task cgan for simultaneous spaceborne dsm refinement and roof-type classification. Remote Sensing. 11(11): p. 1262, **2019**

[18] Castagno, J. and E. Atkins, Roof shape classification from LiDAR and satellite image data fusion using supervised learning. Sensors. 18(11): p. 3960, **2018**

[19] Assouline, D., N. Mohajeri, and J.-L. Scartezzini. Building rooftop classification using random forests for large-scale PV deployment. in Earth resources and environmental remote Sensing/GIS Applications VIII. 2017. SPIE.

[20] Rottensteiner, F., G. Sohn, J. Jung, M. Gerke, C. Baillard, S. Benitez, and U. Breitkopf, The ISPRS benchmark on urban object classification and 3D building reconstruction. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences I-3 (2012), Nr. 1. 1(1): p. 293-298, **2012**

[21] Rottensteiner, F., G. Sohn, M. Gerke, J.D. Wegner, U. Breitkopf, and J. Jung, Results of the ISPRS benchmark on urban object detection and 3D building reconstruction. ISPRS journal of photogrammetry and remote sensing. 93: p. 256-271, **2014**

[22] Marmanis, D., K. Schindler, J.D. Wegner, S. Galliani, M. Datcu, and U. Stilla, Classification with an edge: Improving semantic image segmentation with boundary detection. ISPRS Journal of Photogrammetry and Remote Sensing. 135: p. 158-172, **2018**

[23] Buyukdemircioglu, M., R. Can, and S. Kocaman, Deep learning based roof type classification using very high resolution aerial imagery. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. 43: p. 55-60, **2021**

[24] Qin, R., J. Tian, and P. Reinartz, 3D change detection–approaches and applications. ISPRS Journal of Photogrammetry and Remote Sensing. 122: p. 41-56, **2016**

[25] Li, Z., J.D. Wegner, and A. Lucchi. Topological map extraction from overhead images. in Proceedings of the IEEE/CVF International Conference on Computer Vision. **2019**.

[26] Yi, Y., Z. Zhang, W. Zhang, C. Zhang, W. Li, and T. Zhao, Semantic segmentation of urban buildings from VHR remote sensing imagery using a deep convolutional neural network. Remote sensing. 11(15): p. 1774, **2019**

[27] Kada, M. and D. Kuramin, ALS Point Cloud Classification Using Pointnet++ and KPCONV with Prior Knowledge. Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 46: p. 91-96, **2021**

[28] Jiwani, A., S. Ganguly, C. Ding, N. Zhou, and D.M. Chan, A semantic segmentation network for urban-scale building footprint extraction using rgb satellite imagery. arXiv preprint arXiv:2104.01263, **2021**

[29] Li, Z., Q. Xin, Y. Sun, and M. Cao, A deep learning-based framework for automated extraction of building footprint polygons from very high-resolution aerial imagery. Remote Sensing. 13(18): p. 3630, **2021**

[30] Sun, X., W. Zhao, R.V. Maretto, and C. Persello, Building outline extraction from aerial imagery and digital surface model with a frame field learning framework. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. 43: p. 487-493, **2021**

[31] Bittner, K., F. Adam, S. Cui, M. Körner, and P. Reinartz, Building footprint extraction from VHR remote sensing images combined with normalized DSMs using fused fully convolutional networks. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 11(8): p. 2615-2629, **2018**

[32] Zhao, W., C. Persello, and A. Stein, Building outline delineation: From aerial images to polygons with an improved end-to-end learning framework. ISPRS journal of photogrammetry and remote sensing. 175: p. 119-131, **2021**

[33] Xu, Y., L. Wu, Z. Xie, and Z. Chen, Building extraction in very high resolution remote sensing imagery using deep learning and guided filters. Remote Sensing. 10(1): p. 144, **2018**

[34] Ok, A.O., J.D. Wegner, C. Heipke, F. Rottensteiner, U. Soergel, and V. Toprak, Matching of straight line segments from aerial stereo images of urban areas. ISPRS Journal of Photogrammetry and Remote Sensing. 74: p. 133-152, **2012**

[35] Buyukdemircioglu, M., S. Kocaman, and M. Kada, Deep learning for 3D building reconstruction: A review. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences: p. 359-366, **2022**

[36] Hensel, S., S. Goebbels, and M. Kada, BUILDING ROOF VECTORIZATION WITH PPGNET. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences, **2021**

[37] Zhang, F., N. Nauata, and Y. Furukawa. Conv-mpn: Convolutional message passing neural network for structured outdoor architecture reconstruction. in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2020.

[38] Zhao, W., C. Persello, and A. Stein, Extracting planar roof structures from very high resolution images using graph neural networks. ISPRS Journal of Photogrammetry and Remote Sensing. 187: p. 34-45, **2022**

[39] Qian, Z., M. Chen, T. Zhong, F. Zhang, R. Zhu, Z. Zhang, K. Zhang, Z. Sun, and G. Lü, Deep Roof Refiner: A detail-oriented deep learning network for refined delineation of roof structure lines using satellite imagery. International Journal of Applied Earth Observation and Geoinformation. 107: p. 102680, **2022**

[40] Alidoost, F., H. Arefi, and M. Hahn, Y-SHAPED CONVOLUTIONAL NEURAL NETWORK FOR 3D ROOF ELEMENTS EXTRACTION TO RECONSTRUCT BUILDING MODELS FROM A SINGLE AERIAL IMAGE. ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences. 5(2), **2020**

[41] Kenzhebay, M., Planar roof structure extraction from Very High-Resolution aerial images and Digital Surface Models using deep learning. **2022**, University of Twente.

[42] Muftah, H., T. Rowan, and A. Butler, Towards open-source LoD2 modelling using convolutional neural networks. Modeling Earth Systems and Environment: p. 1-17, **2021**

[43] Musialski, P., P. Wonka, D.G. Aliaga, M. Wimmer, L. Van Gool, and W. Purgathofer. A survey of urban reconstruction. in Computer graphics forum. 2013. Wiley Online Library.

[44] Haala, N. and M. Kada, An update on automatic 3D building reconstruction. ISPRS Journal of Photogrammetry and Remote Sensing. 65(6): p. 570-580, **2010**

[45] Wang, R., 3D building modeling using images and LiDAR: A review. International Journal of Image and Data Fusion. 4(4): p. 273-292, **2013**

[46] Kada, M. and A. Wichmann, Sub-surface growing and boundary generalization for 3D building reconstruction. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 3, **2012**

[47] Nan, L. and P. Wonka. Polyfit: Polygonal surface reconstruction from point clouds. in Proceedings of the IEEE International Conference on Computer Vision. **2017**.

[48] Malihi, S., M.J. Valadan Zoej, and M. Hahn, Large-scale accurate reconstruction of buildings employing point clouds generated from UAV imagery. Remote Sensing. 10(7): p. 1148, **2018**

[49] Buyukdemircioglu, M. and S. Kocaman, A 3D campus application based on city models and WebGL. **2018b**

[50] Buyukdemircioglu, M., S. Kocaman, and U. Isikdag, Semi-automatic 3D city model generation from large-format aerial images. ISPRS International Journal of Geo-Information. 7(9): p. 339, **2018a**

[51] Xie, L., H. Hu, Q. Zhu, X. Li, S. Tang, Y. Li, R. Guo, Y. Zhang, and W. Wang, Combined rule-based and hypothesis-based method for building model reconstruction from photogrammetric point clouds. Remote Sensing. 13(6): p. 1107, **2021**

[52] Bizjak, M., B. Žalik, and N. Lukač, Parameter-Free Half-Spaces Based 3D Building Reconstruction Using Ground and Segmented Building Points from Airborne LiDAR Data with 2D Outlines. Remote Sensing. 13(21): p. 4430, **2021**

[53] Drešček, U., M. Kosmatin Fras, J. Tekavec, and A. Lisec, Spatial ETL for 3D building modelling based on unmanned aerial vehicle data in semi-urban areas. Remote Sensing. 12(12): p. 1972, **2020**

[54] Murtiyoso, A., M. Veriandi, D. Suwardhi, B. Soeksmantono, and A.B. Harto, Automatic Workflow for Roof Extraction and Generation of 3D CityGML Models from Low-Cost UAV Image-Derived Point Clouds. ISPRS International Journal of Geo-Information. 9(12): p. 743, **2020**

[55] Li, Y. and B. Wu, Automatic 3D reconstruction of complex buildings from incomplete point clouds with topological-relation constraints. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 5: p. 85-92, **2020**

[56] Li, Z. and J. Shan, RANSAC-based multi primitive building reconstruction from 3D point clouds. ISPRS Journal of Photogrammetry and Remote Sensing. 185: p. 247-260, **2022**

[57] Peters, R., B. Dukai, S. Vitalis, J. van Liempt, and J. Stoter, Automated 3D reconstruction of LoD2 and LoD1 models for all 10 million buildings of the Netherlands. Photogrammetric Engineering & Remote Sensing. 88(3): p. 165-170, **2022**

[58] Buyukdemircioglu, M. and S. Kocaman, Reconstruction and efficient visualization of heterogeneous 3D city models. Remote Sensing. 12(13): p. 2128, **2020**

[59] Dehbi, Y., F. Hadiji, G. Gröger, K. Kersting, and L. Plümer, Statistical relational learning of grammar rules for 3D building reconstruction. Transactions in GIS. 21(1): p. 134-150, **2017**

[60] Biljecki, F., H. Ledoux, and J. Stoter, Generating 3D city models without elevation data. Computers, Environment and Urban Systems. 64: p. 1-18, **2017**

[61] Biljecki, F. and Y. Dehbi, Raise the roof: Towards generating LoD2 models without aerial surveys using machine learning. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 4: p. 27-34, **2019**

[62] Park, Y. and J.-M. Guldmann, Creating 3D city models with building footprints and LIDAR point cloud classification: A machine learning approach. Computers, environment and urban systems. 75: p. 76-89, **2019**

[63] Farella, E.M., E. Özdemir, and F. Remondino, 4D Building Reconstruction with Machine Learning and Historical Maps. Applied Sciences. 11(4): p. 1445, **2021**

[64] Liu, C., D. Kong, S. Wang, Z. Wang, J. Li, and B. Yin, Deep3D reconstruction: Methods, data, and challenges. Frontiers of Information Technology & Electronic Engineering. 22(5): p. 652-672, **2021**

[65] Simonyan, K. and A. Zisserman, Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556, **2014**

[66] Wang, K. and J.-M. Frahm. Single view parametric building reconstruction from satellite imagery. in 2017 International Conference on 3D Vision (3DV). **2017**. IEEE.

[67] Zeng, H., J. Wu, and Y. Furukawa. Neural procedural reconstruction for residential buildings. in Proceedings of the European Conference on Computer Vision (ECCV). **2018**.

[68] Nishida, G., A. Bousseau, and D.G. Aliaga. Procedural modeling of a building from a single image. in Computer Graphics Forum. **2018**. Wiley Online Library.

[69] Agoub, A., V. Schmidt, and M. Kada, Generating 3D city models based on the semantic segmentation of lidar data using convolutional neural networks. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 4: p. 3-10, **2019**

[70] Goodfellow, I., J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, Generative adversarial networks. Communications of the ACM. 63(11): p. 139-144, **2020**

[71] Bittner, K. and M. Korner. Automatic large-scale 3d building shape refinement using conditional generative adversarial networks. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops. **2018**.

[72] Bittner, K., P. d'Angelo, M. Körner, and P. Reinartz, DSM-to-LoD2: Spaceborne stereo digital surface model refinement. Remote Sensing. 10(12): p. 1926, **2018**

[73] Beer, L., Automatic generation of lod1 city models and building segmentation from single aerial orthographic images using conditional generative adversarial networks. GI_Forum 2019. 7: p. 119-133, **2019**

[74] Kelly, T., P. Guerrero, A. Steed, P. Wonka, and N.J. Mitra, FrankenGAN: guided detail synthesis for building mass-models using style-synchonized GANs. arXiv preprint arXiv:1806.07179, **2018**

[75] Qian, Y., H. Zhang, and Y. Furukawa. Roof-gan: Learning to generate roof geometry and relations for residential houses. in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. **2021**.

[76] Zhang, L., Z. Li, A. Li, and F. Liu, Large-scale urban point cloud labeling and reconstruction. ISPRS Journal of Photogrammetry and Remote Sensing. 138: p. 86-100, **2018**

[77] Zhang, L. and L. Zhang, Deep learning-based classification and reconstruction of residential scenes from large-scale point clouds. IEEE Transactions on Geoscience and Remote Sensing. 56(4): p. 1887-1897, **2017**

[78] Leotta, M.J., C. Long, B. Jacquet, M. Zins, D. Lipsa, J. Shan, B. Xu, Z. Li, X. Zhang, and S.-F. Chang. Urban semantic 3D reconstruction from multiview satellite imagery. in Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops. **2019**.

[79] Xu, B., X. Zhang, Z. Li, M. Leotta, S.-F. Chang, and J. Shan, Deep learning guided building reconstruction from satellite imagery-derived point clouds. arXiv preprint arXiv:2005.09223, **2020**

[80] Yu, D., S. Ji, J. Liu, and S. Wei, Automatic 3D building reconstruction from multi-view aerial images with deep learning. ISPRS Journal of Photogrammetry and Remote Sensing. 171: p. 155-170, **2021**

[81] Gui, S. and R. Qin, Automated LoD-2 model reconstruction from very-high-resolution satellite-derived digital surface model and orthophoto. ISPRS Journal of Photogrammetry and Remote Sensing. 181: p. 1-19, **2021**

[82] Kapoor, A., H. Larco, and R. Kiveris. Nostalgin: Extracting 3D city models from historical image data. in Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. **2019**.

[83] Partovi, T., F. Fraundorfer, R. Bahmanyar, H. Huang, and P. Reinartz, Automatic 3-D building model reconstruction from very high resolution stereo satellite imagery. Remote Sensing. 11(14): p. 1660, **2019**

[84] Teo, T.-A. Deep-Learning for Lod1 Building Reconstruction from Airborne Lidar Data. in IGARSS 2019-2019 IEEE International Geoscience and Remote Sensing Symposium. 2019. IEEE.

[85] Kippers, R., M. Koeva, M. van Keulen, and S.O. Elberink, Automatic 3D building model generation using deep learning methods based on cityjson and 2D floor plans. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences. 46: p. 49-54, **2021**

[86] Yu, D., S. Wei, J. Liu, and S. Ji, ADVANCED APPROACH FOR AUTOMATIC RECONSTRUCTION OF 3D BUILDINGS FROM AERIAL IMAGES. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences. 43, **2020**

[87] Zhang, W., Z. Li, and J. Shan, Optimal model fitting for building reconstruction from point clouds. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing. 14: p. 9636-9650, **2021**

[88] Chen, Z., H. Ledoux, S. Khademi, and L. Nan, Reconstructing compact building models from point clouds using deep implicit fields. ISPRS Journal of Photogrammetry and Remote Sensing. 194: p. 58-73, **2022**

[89] Pepe, M., D. Costantino, V.S. Alfio, G. Vozza, and E. Cartellino, A novel method based on deep learning, GIS and geomatics software for building a 3D city model from VHR satellite stereo imagery. ISPRS International Journal of Geo-Information. 10(10): p. 697, **2021**

[90] Chen, Y., T. Hong, X. Luo, and B. Hooper, Development of city buildings dataset for urban building energy modeling. Energy and Buildings. 183: p. 252-265, **2019**

[91] Xu, Z., L. Zhang, H. Li, Y.-H. Lin, and S. Yin, Combining IFC and 3D tiles to create 3D visualization for building information modeling. Automation in Construction. 109: p. 102995, **2020**

[92] Rossknecht, M. and E. Airaksinen, Concept and evaluation of heating demand prediction based on 3D city models and the citygml energy ADE—Case study helsinki. ISPRS International Journal of Geo-Information. 9(10): p. 602, **2020**

[93] Isikdag, U. and K. Sahin, WEB BASED 3D VISUALISATION OF TIME-VARYING AIR QUALITY INFORMATION. International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences. 42(4), **2018**

[94] Kilsedar, C.E., F. Fissore, F. Pirotti, and M.A. Brovelli, Extraction and visualization of 3D building models in urban areas for flood simulation. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 42: p. 669-673, **2019**

[95] Jovanović, D., S. Milovanov, I. Ruskovski, M. Govedarica, D. Sladić, A. Radulović, and V. Pajić, Building virtual 3D city model for smart cities applications: A case study on campus area of the university of novi sad. ISPRS International Journal of Geo-Information. 9(8): p. 476, **2020**

[96] Pepe, M., D. Costantino, V.S. Alfio, M.G. Angelini, and A. Restuccia Garofalo, A CityGML multiscale approach for the conservation and management of cultural heritage: The case study of the old town of Taranto (Italy). ISPRS International Journal of Geo-Information. 9(7): p. 449, **2020**

[97] Dembski, F., U. Wössner, M. Letzgus, M. Ruddat, and C. Yamu, Urban digital twins for smart cities and citizens: The case study of Herrenberg, Germany. Sustainability. 12(6): p. 2307, **2020**

[98] Virtanen, J.-P., K. Jaalama, T. Puustinen, A. Julin, J. Hyyppä, and H. Hyyppä, Near real-time semantic view analysis of 3D city models in web browser. ISPRS International Journal of Geo-Information. 10(3): p. 138, **2021**

[99] Uggla, M., P. Olsson, B. Abdi, B. Axelsson, M. Calvert, U. Christensen, D. Gardevärn, G. Hirsch, E. Jeansson, and Z. Kadric, Future Swedish 3D City

Models—Specifications, Test Data, and Evaluation. ISPRS International Journal of Geo-Information. 12(2): p. 47, **2023**

[100] Dursun, İ., M. Aslan, İ. Cankurt, C. Yıldırım, and E. Ayyıldız, 3D city models as a 3D cadastral layer: the case of TKGM model. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences.-2022. 43: p. 507, **2022**

[101] CesiumJS - Geospatial 3D Mapping and Virtual Globe Platform, https://cesium.com/platform/cesiumjs/, Last Access: 27.03.2022

[102] 3DTiles, https://github.com/CesiumGS/3d-tiles, Last Access: 27.03.2022

[103] The GL Transmission Format (glTF), https://www.khronos.org/gltf/, Last Access: 27.03.2022

[104] Cesium ION, https://cesium.com/platform/cesium-ion/, Last Access: 27.03.2022

[105] Carbonell-Carrera, C., P. Gunalp, J.L. Saorin, and S. Hess-Medler, Think spatially with game engine. ISPRS International Journal of Geo-Information. 9(03): p. 159, **2020**

[106] Huo, Y., A. Yang, Q. Jia, Y. Chen, B. He, and J. Li, Efficient Visualization of Large-Scale Oblique Photogrammetry Models in Unreal Engine. ISPRS International Journal of Geo-Information. 10(10): p. 643, **2021**

[107] Merlo, A., C. Sánchez Belenguer, E. Vendrell Vidal, F. Fantini, and A. Aliperta, 3D model visualization enhancements in real-time game engines. International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. 5: p. W1, **2013**

[108] Kersten, T., F. Tschirschwitz, and S. Deggim. Development of a virtual museum including a 4D presentation of building history in virtual reality. in TC II & CIPA 3D Virtual Reconstruction and Visualization of Complex Architectures, 1–3 March 2017, Nafplio, Greece. **2017**. Copernicus.

[109] Walmsley, A.P. and T.P. Kersten, The IMPERIAL Cathedral in Königslutter (Germany) as an immersive experience in virtual reality with integrated 360 panoramic photography. Applied Sciences. 10(4): p. 1517, **2020**

[110] Kim, J.Y. and M.J. Kim, Exploring visual perceptions of spatial information for wayfinding in virtual reality environments. Applied Sciences. 10(10): p. 3461, **2020**

[111] Broucke, S.V. and N. Deligiannis. Visualization of real-time heterogeneous smart city data using virtual reality. in 2019 IEEE International smart cities conference (ISC2). **2019**. IEEE.

[112] Bouloukakis, M., N. Partarakis, I. Drossis, M. Kalaitzakis, and C. Stephanidis, Virtual reality for smart city visualization and monitoring. Mediterranean Cities and Island Communities: Smart, Sustainable, Inclusive and Resilient: p. 1-18, **2019**

[113] Tschirschwitz, F., C. Richerzhagen, H.-J. Przybilla, and T.P. Kersten, Duisburg 1566: transferring a historic 3D city model from google earth into a virtual reality application. PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science. 87: p. 47-56, **2019**

[114] Yagol, P., F. Ramos, S. Trilles, J. Torres-Sospedra, and F.J. Perales, New trends in using augmented reality apps for smart city contexts. ISPRS International Journal of Geo-Information. 7(12): p. 478, **2018**

[115] UrbanX: Urban planning in mixed reality, https://urbanxgis.wordpress.com/, Last Access: 27.03.2022

[116] Sanaeipoor, S. and K.H. Emami. Smart city: exploring the role of augmented reality in placemaking. in 2020 4th International Conference on Smart City, Internet of Things and Applications (SCIOT). **2020**

[117] Liu, F., T. Jonsson, and S. Seipel, Evaluation of augmented reality-based building diagnostics using third person perspective. ISPRS International Journal of Geo-Information. 9(1): p. 53, **2020**

[118] Santana, J.M., J. Wendel, A. Trujillo, J.P. Suárez, A. Simons, and A. Koch. Multimodal location based services—semantic 3D city data as virtual and augmented reality. in Progress in location-based services **2016**. Springer.

[119] Buyukdemircioglu, M., R. Can, S. Kocaman, and M. Kada, Deep Learning Based Building Footprint Extraction from Very High Resolution True Orthophotos and Ndsm. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences: p. 211-218, **2022**

[120] Ronneberger, O., P. Fischer, and T. Brox. U-net: Convolutional networks for biomedical image segmentation. in Medical Image Computing and Computer-Assisted Intervention–MICCAI 2015: 18th International Conference, Munich, Germany, October 5-9, 2015, Proceedings, Part III 18. **2015**. Springer.

[121] Chaurasia, A. and E. Culurciello. Linknet: Exploiting encoder representations for efficient semantic segmentation. in 2017 IEEE visual communications and image processing (VCIP). **2017**. IEEE.

[122] Geospatial Data Abstraction Library (GDAL), https://gdal.org/, Last Access: 27.03.2022

[123] Visvalingam, M. and J.D. Whyatt. The Douglas-Peucker algorithm for line simplification: re-evaluation through visualization. in Computer Graphics Forum. **1990**. Wiley Online Library.

[124] Redmon, J., S. Divvala, R. Girshick, and A. Farhadi. You only look once: Unified, real-time object detection. in Proceedings of the IEEE conference on computer vision and pattern recognition. **2016**

[125] Akyon, F.C., S.O. Altinuc, and A. Temizel. Slicing aided hyper inference and fine-tuning for small object detection. in 2022 IEEE International Conference on Image Processing (ICIP). **2022**

[126] Büyükdemircioğlu, M., Implementatıon And Web-Based Visualization Of 3D City Models. **2018**.

[127] Tan, M. and Q. Le. Efficientnet: Rethinking model scaling for convolutional neural networks. in International conference on machine learning. **2019**

[128] He, K., X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. in Proceedings of the IEEE conference on computer vision and pattern recognition. **2016**.

[129] Deng, J., W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. in 2009 IEEE conference on computer vision and pattern recognition. **2009**

[130] Han, K., Y. Wang, H. Chen, X. Chen, J. Guo, Z. Liu, Y. Tang, A. Xiao, C. Xu, and Y. Xu, A survey on vision transformer. IEEE transactions on pattern analysis and machine intelligence. 45(1): p. 87-110, **2022**

[131] Li, H., H. Yu, J. Wang, W. Yang, L. Yu, and S. Scherer, ULSD: Unified line segment detection across pinhole, fisheye, and spherical cameras. ISPRS Journal of Photogrammetry and Remote Sensing. 178: p. 187-202, **2021**

[132] Huang, K., Y. Wang, Z. Zhou, T. Ding, S. Gao, and Y. Ma. Learning to parse wireframes in images of man-made environments. in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. **2018**.

[133] Denis, P., J.H. Elder, and F.J. Estrada. Efficient edge-based methods for estimating manhattan frames in urban imagery. in Computer Vision–ECCV 2008: 10th European Conference on Computer Vision, Marseille, France, October 12-18, 2008, Proceedings, Part II 10. **2008**. Springer.

[134] Newell, A., K. Yang, and J. Deng. Stacked hourglass networks for human pose estimation. in Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VIII 14. **2016**. Springer.

[135] Zhou, Y., H. Qi, and Y. Ma. End-to-end wireframe parsing. in Proceedings of the IEEE/CVF International Conference on Computer Vision. **2019**.

[136] Xue, N., T. Wu, S. Bai, F.-D. Wang, G.-S. Xia, L. Zhang, and P.H. Torr, Holistically-Attracted Wireframe Parsing: From Supervised to Self-Supervised Learning. arXiv preprint arXiv:2210.12971, **2022**

[137] Buyukdemircioglu, M. and S. Kocaman, Development of a Smart City Concept in Virtual Reality Environment. The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences: p. 51-58, **2022**