

Chaoquan Zhang

Reconstruction of 3D Building Models with Semantic Information Using Crowdsourcing Approaches

Thesis for the degree of Philosophiae Doctor

Trondheim, November 2023

Norwegian University of Science and Technology
Faculty of Engineering
Department of Civil and Environmental Engineering



Norwegian University of
Science and Technology

NTNU

Norwegian University of Science and Technology

Thesis for the degree of Philosophiae Doctor

Faculty of Engineering

Department of Civil and Environmental Engineering

© Chaoquan Zhang

ISBN 978-82-326-7168-7 (printed ver.)

ISBN 978-82-326-7167-0 (electronic ver.)

ISSN 1503-8181 (printed ver.)

ISSN 2703-8084 (electronic ver.)

Doctoral theses at NTNU, 2023:396



Printed by Skipnes Kommunikasjon AS

Reconstruction of 3D Building Models with Semantic Information Using Crowdsourcing Approaches

Preface

This PhD study was conducted at the Department of Civil and Environmental Engineering at the Norwegian University of Science and Technology (NTNU) under the supervision of Professor Hongchao Fan. The thesis is submitted as partial fulfilment of the requirements for the degree of Philosophiae Doctor.

The research was financed by the Norwegian Research Council as part of the research project “Lambda Road”.

Abstract

Over the last decade, increasing municipalities have decided to gear up the construction of smart cities. The transformation to a smart city requires to digitalize the whole city as faithfully as possible. To accomplish this goal, the modelling of the city should consider two major aspects: 1) level of details (LoDs) of 3D building models and 2) the inclusion of semantic information. The LoDs concept following the City Geography Markup Language (CityGML) 2.0 standard depicts the content of 3D building models and meanwhile, LoD3 building models are the research focus of this thesis. 3D building models at LoD3, along with semantic information, play an essential role in smart city-related applications. They not only are appropriate for visualization tasks, but also are beneficial to further advanced analysis, e.g., solar energy potential estimation, flood simulation, urban planning, post-disaster assessment, and telecommunication like the simulation of 5G signal propagation etc. However, LoD3 3D building models are only available in some small regions, and they are lack of the semantic information. Moreover, the cost of generating LoD3 models is high both in time and in labor, since it is impossible for existing methods to detect and reconstruct LoD3 buildings in an automated manner. Data acquisition is another challenge as well because it requires expensive equipment/sensors. Hence, this PhD thesis aims to propose possible solutions to resolve above challenges.

This thesis concentrates on the utilization of crowdsourcing approaches for reconstruction of LoD3 3D building models with semantic information. Given the difficulty of automatically detecting and reconstructing LoD3 building models, and the challenge in data collection, an interactive approach is proposed to reconstruct LoD3 3D building models with semantics from VGI (Volunteered Geographic Information) data. Specifically, a web-based interactive 3D building modelling platform, VGI3D, is developed to reconstruct 3D building models from street-level images, which contain rich information of façade structures, thus ensuring that the reconstructed 3D models have semantic information. VGI3D is designed to have simple interoperability in order to attract and encourage more volunteers for 3D model contributions, and also with the ambition of becoming a VGI platform to collect LoD3 building models with semantics by using the power of crowdsourcing. Moreover, a limited usability testing is conducted among expert and non-expert participants, which proves the usefulness of VGI3D and its promising value for the 3D modelling community.

Usually, the complete roof structures are not visible in street-level images. In VGI3D the roof models are automatically generated by selecting a specific roof type, but their geometric accuracy is not guaranteed. In addition, some buildings have complex roof structures, but they are excluded in the predefined simple roof types of VGI3D. To address these issues, an improved multi-task pointwise network is proposed. This network can simultaneously segment instances (i.e., individual roof planes) and semantics (i.e., groups of roof planes with

similar geometric shapes) in standard airborne laser scanning (ALS) point clouds. The segmented roof planes can then be reconstructed into polygon meshes and combined with the façade structures generated from VGI3D. As a result, more accurate and photorealistic 3D building models can be created eventually. Furthermore, to train the proposed network, a new roof dataset (RoofNTNU) with 7 typical roof types in West Europe is established, by taking ALS point clouds with standard point density as training data for automatic and more general segmentation. The experiments on RoofNTNU dataset demonstrate the effectiveness of the proposed method, achieving promising segmentation results: the mean precision (mPrec) of 96.2% for instance segmentation task and mean accuracy (mAcc) of 94.4% for semantic segmentation task.

The VGI3D with simple interoperability alone may not be sufficient to motivate more volunteers to contribute. After the 3D building models are reconstructed, it is necessary to integrate them into a virtual 3D city environment, which allows users to be able to view and interact with their 3D models in a 3D scene, presenting a natural way to perceive 3D objects. This can not only increase their sense of fulfillment in contributing to the VGI community, but also can potentially attract more valuable users from the 3D modelling domain, as well as even increase the interests of non-experts. To accomplish this goal, a 3D visualization platform is therefore developed, where digitizes the real city environment including 3D terrain, 3D building models, 3D road networks, and other road-related 3D objects (e.g., traffic lights/signs, trees, etc.).

For the implementation of the 3D visualization platform, the 3D terrain covering the whole Norway is generated from Digital Terrain Model (DTM) data and is optimized using triangulated irregular network (TIN) and LoDs for faster rendering. 3D building models are obtained from three means: hybrid generation of ALS point clouds and OpenStreetMap (OSM) footprints (LoD2); VGI3D generation (LoD3); and crowdsourced SketchUp models (LoD3). All of them are georeferenced and saved as CityGML format and then are visualized on the platform relying on 3D Tiles technique. Regarding the 3D road networks, a collision detection is utilized to project the original 2D road polylines onto 3D terrain. Then, Catmull-Rom spline algorithm is employed to expand the 3D road polylines to 3D polygons. To collect road-related objects (traffic signs/lights), an automatic method is proposed to detect road objects from VGI street-level images and place them to the approximate correct positions. Two convolutional neural networks (CNNs) are applied to detect and classify the road objects. Additionally, to locate the detected objects, an attributed topological binary tree (ATBT) is firstly established based on urban rules for image sequences to depict the coherent relations of topologies, attributes and semantics of the road objects. Then the ATBT is further matched with map features on OpenStreetMap (OSM) to determine the correct placed positions. A case study is conducted and achieves near-precise localization results in terms of completeness and positional accuracy.

To summarize, crowdsourcing is a powerful tool to enhance the reconstruction of 3D building models with semantic information, thus accelerating the digitalization of smart cities. VGI3D is a user-friendly and efficient system that requires less input and simple user interaction, making it ideal for quick yet relatively detailed (i.e., LoD3) modelling. RoofNet is a significant enhancement to VGI3D in roof reconstruction. The combination of VGI3D and 3D visualization platform is beneficial to attract more users and motivate them for 3D building contributions at the same time.

The thesis also presents the research directions for future work. For example, it is important to assess the quality of LoD3 3D building models generated by VGI3D, which has not been done yet. Hence, quality evaluation should be a priority in the future. Additionally, VGI3D is still in its early stage of development and has plenty of room for further improvement and optimization. On one hand, it is necessary to further strengthen our CNN model and enable it to detect façade elements as accurately as possible so as to reduce the user interaction costs for updating 3D models. On the other hand, from suggestions/comments of usability testing, it is apparent that reconstructing complex buildings will facilitate applications that place high demands on the 3D models.

Keywords

3D building modelling; CityGML; OpenStreetMap; user interoperability; VGI street-level images; roof plane segmentation; roof dataset; airborne laser scanning (ALS) point clouds; instance segmentation; 3D visualization; object detection; attributed topological binary tree (ATBT)

Acknowledgements

Many persons have contributed to making this thesis possible and have accompanied me on the journey of this PhD study. I am grateful to all of them by various dedication, involvement and encouragement.

I really appreciate Prof. Hongchao Fan, my supervisor, for your supervision, guidance, innovative and inspired ideas, and both freedom and time you gave me to try out different possibilities. I am highly thankful for your no-regular-meeting style, which helped me save much time on making slides. Instead, your office door is always opened, making it simpler to discuss with you at any time with no need to book an appointment ahead. My sincere gratitude for your valuable suggestions, high-quality paper review comments, all our discussions and your vision that helped shape this PhD work into what it is.

I am grateful to my Master's supervisor, Prof. Bo Mao. Without his recommendation letter, I would not have this valuable opportunity to study abroad and realize one of my dreams.

Special thanks to my dear friends Gefei Kong, Rouzbeh Shabani, Rui Tao and Lei Zhang for your company and support during those difficult moments of pandemic times. They have always been motivating and inviting me to participate in all possible social activities in Trondheim, including festival celebration, hiking, travelling, etc. I also would like to thank Sindre Johannessen and Almudena Sebastian Gonzalez for those sweet and unforgettable moments we had when living in Voll Studentby, and later their continuous caring and greeting towards me, even though they had graduated and left Trondheim.

To my dear colleagues and friends from other research groups at IBM, I am grateful for your kind words, warm smiles and joyful lunch conversations. I would like to express my gratitude to Maren Berg Grimstad, PhD coordinator, who takes good care of all PhD students from all aspects, such as by listening to our different opinions and making every effort to help solve hardships we are facing. Professor Terje Midtbø and Associate Professor Albert Lau, thank you for your open ears and sophisticated advice. Associate Professor Linfang Ding, I am thankful for your help to spend time on proofreading and improve the thesis writing quality. Thank you, Kenneth Sundli, for resolving annoying and unexpected IT issues many times and shielding my workstation by your advanced IT technologies.

To the Master students that I have co-supervised, Marie, Vilde, Fredrik and Alfred, I would like to thank each of you for your interest and dedication to study Geomatics and I also learned a lot from you guys at the same time.

Last but not the least, it is time to give my deepest gratitude to my wonderful parents and grandma for their unconditional love, care, support and encouragement throughout the whole period of PhD study. To my three cousins, Qian, Huan and Xiaoqi, I am so lucky to have you

guys by my side since I remember and to provide absolute trust in me. To my aunts and uncles, I am grateful to get endless support and care from you since childhood.

Many thanks! 谢谢! Tusen Takk!

Abbreviations

AABBTree	Axis Aligned Bounding Box Tree
ADE	Application Domain Extension
ALS	Airborne Laser Scanning
API	Application Programming Interface
ATBT	Attributed Topological Binary Tree
b3dm	Batched 3D Model
BIM	Building Information Models
B-rep	Boundary Representation
CityGML	City Geography Markup Language
CNN	Convolutional Neural Network
CSG	Constructive Solid Geometry
CSS	Cascading Style Sheets
DSM	Digital Surface Model
DTM	Digital Terrain Model
EO	Earth Observation
GIS	Geographic Information System
GNSS	Global Navigation Satellite Systems
GPU	Graphics Processing Unit
HT	Hough Transform
k-NN	k-Nearest Neighbors
LoD	Level of Detail
MLS	Mobile Laser Scanning
MRF	Markov Random Field
MVC	Model-View-Controller
NLOS	Non Line of Sight
OGC	Open Geospatial Consortium
OSM	OpenStreetMap
POI	Point of Interest
RANSAC	RANdom SAmples Consensus
SDK	Software Development Kit
SfM	Structure from Motion
TIN	Triangulated Irregular Network

TLS	Terrestrial Laser Scanning
UAV	Unmanned Aerial Vehicle
VGI	Volunteered Geographic Information
VR	Virtual Reality
WebGL	Web Graphics Library

List of tables

Table 2.1. The existing VGI data platforms by active contributions appeared in VGI-related researches (reproduced from paper (Fan et al., 2022)).	16
Table 3.1. Overview of correspondence among semantic and instance labels, plane geometries and roof types in the ground truth. Please note that do not confuse semantic label color with instance label color, as they belong to two different tasks.	38
Table 3.2. Summarized urban rules used in attributed topological binary trees (ATBT).	42
Table 4.1. Statistics about raw data collected from participants. Mean(μ), standard deviation(σ), lower quartile(Q1), middle quartile(Q2), upper quartile(Q3), interquartile range (IQR). 3DBMs: 3D Building Models; 3DBMing: 3D Building Modelling; 3DMing: 3D Modelling.	49
Table 4.2. Correlation table showing Spearman's rho and significance values for parameters. Values in red present strong positive correlation; values in blue present moderate positive correlation; other values show weak correlation. 3DBMs: 3D Building Models; 3DBMing: 3D Building Modelling; 3DMing: 3D Modelling.	50
Table 4.3. Summary of buildings with various sizes and façade complexities for sensitive analysis. The fifth column shows the interactive modelling time (semi-auto), time of interactively updating incorrect façade elements (update), exporting 3D model time (export) and updated façade elements in (·). Blue is for door, cyan is for window and green is for balcony.	53
Table 4.4. Quantitative results of instance segmentation on RoofNTNU dataset.	57
Table 4.5. Quantitative results of semantic segmentation on RoofNTNU dataset.	58
Table 4.6. Instance and semantic segmentation results of all experiments of architecture study on RoofNTNU.	60
Table 4.7. Quantitative comparison of mIoU and pixel accuracy between PSPNet and DeepLabv3+.	67

List of figures

Figure 1.1. Examples of smart city-related applications requiring 3D building models with semantic information.	3
Figure 1.2. Research questions in context of the different boundary levels.	11
Figure 3.1. The overall research framework of the thesis.....	27
Figure 3.2. A typical workflow of the VGI3D. In the user interaction module, users provide façade boundaries, roof type, façade orientation and number of floors. Then, images are inputted into façade elements extraction module to automatically obtain the locations of the windows, doors and balconies. In the 3D building modelling module, the locations are transformed from 2D to 3D space and are then rotated according to the façade orientation. Finally, the 3D building model is reconstructed and displayed to users.	28
Figure 3.3. (a) The architecture of our RoofNet with two proposed modules: (b) the feature fusion (FF) module; (c) the joint features learning (JFL) module.	34
Figure 3.4. Visualization of defined roof types in the dataset RoofNTNU. The left figure is the abstract representation of roof types 1-6, while the right figure shows some practical examples of roof type combination/7.	37
Figure 3.5. The distribution of the roof types in RoofNTNU dataset.....	39
Figure 3.6. The sketch of 2D road polylines mapping onto terrain meshes (borrowed from the work of Vaaranemi et al. (2011)).....	40
Figure 3.7. The workflow of the automated detection and localization of road objects from street-level images.	41
Figure 3.8. ATBT generation and its self-updating considering the scene depth information. The image numbered 1-3 are gradually approaching to the intersection.....	43
Figure 4.1. Abstract overview of the VGI3D architecture.....	47
Figure 4.2. Box plot of data for illustrating skewness.	49
Figure 4.3. Examples of sparse scatter plots with nonlinearity in appearance but including bound data points internally.....	50
Figure 4.4. Visual comparison results of ASIS, JSNet and the proposed method on the instance segmentation task. Different colors stand for different instances (that is, roof planes). The colors of the same instance in ground truth and prediction are not necessarily the same.....	55

Figure 4.5. Visual comparison results of PointNet++, ASIS, JSNet and the proposed method on the semantic segmentation task. Different colors represent different pairs of roof planes with the same geometric shapes.....	56
Figure 4.6. Key part of baseline ASIS (a) and four groups of experiments (b–e). Their encoders and decoders are consistent with RoofNet.....	59
Figure 4.7. The framework of the web-based 3D visualization platform.....	62
Figure 4.8. (a)-(b) provide a close-up view of Gløshaugen, NTNU, showcasing different elements of the 3D visualization platform, while (c)-(d) present a good perspective for better visualizing and perceiving the 3D terrain.	64
Figure 4.9. Example of 3D terrain visualization using two rendering optimization strategies.	65
Figure 4.10. Cartographic rendering of roads on high-resolution 3D terrain in wireframe mode, in Trondheim, Norway. Different colors represent the different road categories.....	66
Figure 4.11. Visual comparison of traffic lights and signs localization results. The first column represents the results generated by our proposed algorithm. The second column refers to the manually collected “reference data” from Google Maps.....	68

Contents

Preface	i
Abstract	iii
Acknowledgements	vii
Abbreviations	ix
List of tables	xi
List of figures	xiii
Contents	xv
Chapter 1	1
Introduction	1
1.1 Background and motivation	1
1.2 Research questions and research objectives	9
1.3 Thesis organization	12
1.4 List of publications.....	12
Chapter 2	15
Related work	15
2.1 Volunteered geographic information	15
2.1.1 VGI data platforms by active contributions	15
2.1.2 VGI as a data source for 3D modelling	16
2.2 Current approaches of 3D building model reconstruction	17
2.2.1 Traditional methods.....	17
2.2.2 Deep learning-based methods.....	20
2.3 Visualization of 3D building models	21
2.3.1 Standards for 3D visualization	22
2.3.2 Web-based 3D visualization frameworks.....	23
Chapter 3	25
Methodology	25
3.1 Research framework.....	25
3.2 Detailed façade modelling.....	27

3.2.1 Workflow	27
3.2.2 User interaction.....	28
3.2.3 Location extraction of façade elements	29
3.2.4 3D building reconstruction	31
3.3 Roof plane segmentation.....	33
3.3.1 Network architecture	33
3.3.2 Feature fusion (FF) module	35
3.3.3 Joint features learning (JFL) module.....	36
3.3.4 Roof plane ALS point cloud dataset.....	37
3.4 Web-based 3D visualization.....	39
Chapter 4.....	45
Implementation, results and discussion	45
4.1 3D building reconstruction using crowdsourcing method	46
4.1.1 VGI3D	46
4.1.2 Roof plane segmentation in ALS point clouds	53
4.2 Web-based 3D visualization platform.....	60
Chapter 5.....	69
Conclusion and future research.....	69
5.1 Conclusion.....	69
5.2 Limitations and future research.....	70
Bibliography	73
Research publications	83

Chapter 1

Introduction

1.1 Background and motivation

Buildings are one of the most essential components of physical cities, serving multiple functions such as human habitation, economic and entertainment activities. Likewise, 3D building models have become the cornerstone of smart city-related applications in the digital world and have turned into reliable carriers that fuse multi-source city information and express corresponding functions reasonably. In addition, 3D building models facilitate an easy understanding about spatial attributes of virtual urban objects for ordinary people, as the world people live in is inherently 3D and it is natural for them to perceive and interpret 3D scenes more easily.

3D building models are generally categorized into five different levels of details (LoDs) according to the international standard CityGML2.0 (Gröger and Plümer, 2012) of the Open Geospatial Consortium (OGC). The LoD0 represents the ground boundary (or footprint) of a building. The LoD1 represents a cube by extruding a LoD0 model. LoD2 models are composed of simplified roof structures and plain walls. LoD3 models are the upgraded LoD2 with detailed façade structures such as windows and doors, and they have more realistic roof structures by including dormant windows if exist. Lastly, LoD4 models are more complex by containing the indoor components (e.g., ventilation pipelines and furniture) based on LoD3. Moreover, some researchers have also proposed the less generic and more application-driven specifications for 3D building models, because standard LoDs specification lacks necessary degree of freedom and flexibility to support applications for specific purposes (Tang et al., 2020; Biljecki et al., 2016). Nevertheless, this thesis concentrates on the LoD3 building models standardized by the CityGML.

In general, simple LoD1 and LoD2 building models cannot satisfy the requirements of many smart city-related applications, since they lack the sufficient information needed for simulation analysis and visualization. Instead, LoD3 building models with richer semantic information and more detailed geometries can create a more realistic virtual world that is intuitive for human brain to perceive and interpret simulation/visualization results from 3D scenes. Various innovative experiments can thus be carried out easily in this digital world modelled at LoD3 by modifying building geometries or semantics, which will greatly reduce the cost of trial and error. This idea is also consistent with the current trend of digital twins. All these advantages and potentials are credited with the advancements in technologies like laser scanning, photogrammetry and 3D computer vision.

Nowadays LoD3 building models with semantic information are being increasingly employed by smart city-related applications, since semantics can provide added value and further

Introduction

enhance their functions, from both the visualization and analysis perspectives. For instance (see Figure 1.1):

Solar energy potential estimation: Solar power distributors need 3D building models to estimate solar energy potential in urban areas (Nelson and Grubestic, 2020), which involves analyzing and computing the geometric structure and attributes (e.g., area, tilt, azimuth, etc.) of each roof plane or building façade. It then will be beneficial to identify the most appropriate places for solar panel installations in cities. From a green and low-carbon perspective, the 3D building models can contribute to achieving the carbon emission goals set by the Paris Agreement (2022) to some degree.

Flood simulation: Emergency management authorities require information on severity of flood damage to buildings and its impact on city areas, so that they can prepare protective measures in advance (Zhi et al., 2020). 3D building models and high-resolution digital terrain model (DTM) together with computational fluid dynamics are a possible solution to simulate flood in a numerical manner, and it can also vividly illustrate the entire dynamic process of flooding the city. With sufficient and comprehensive attribute information of 3D building models, the accuracy of flood simulation can be further enhanced.

Urban positioning accuracy optimization: In urban environment, the positioning accuracy of GNSS (global navigation satellite systems) is usually degraded, which is mainly caused by lots of buildings' flat surfaces as well as other obstacles and thus results in multipath interference and non-line-of-sight (NLOS) reception problems (Groves et al., 2013). Hence, map service providers expect to obtain 3D building models with the highest details, including detailed roof, façade, window and balcony structures, etc. These detailed geometric data enable map service providers to improve the positioning accuracy of GNSS in dense urban areas. Additionally, the usage of 3D building models allows to visually simulate GNSS signal reflections in a 3D scene, which is friendly for researchers to inspect results and debug their proposed algorithms.

Urban planning: As cities continue to grow and global warming worsens, the urban heat island effect poses a significant challenge for urban planning. To tackle this challenge, 3D building models and 3D vegetation objects like trees can play an important role in examining their impacts on daytime temperature control and human thermal comfort (Park et al., 2021). If 3D building models can include façade material data, it becomes possible to assess the relationship between different building materials and indoor energy consumption, particularly during summer and winter (Li et al., 2020). With internal structures embedded in 3D building models, it is even possible to evaluate the energy consumption of a single indoor room. Such analyses can help reduce carbon emissions and save operation cost from the economy and sustainability point of view. These study cases can eventually guide urban planners in constructing greener, low-carbon and sustainable cities.

Post-disaster assessment: Post-disaster assessment usually requires 3D building models with visual appearance/texture, detailed geometry and rich semantics. It enables public safety agencies to better compare damages to buildings before and after disasters, such as earthquake (Redweik et al., 2017) and landslide (Zeng et al., 2021). It can also help quantitatively evaluate the economic losses and damage levels of buildings, and then provide scientific and reasonable guidance on post-disaster reconstruction, including the financial allocation required for housing reconstruction and resettlement of disaster victims.

Business development and tourism: Due to the impact of the global pandemic, cities that rely on tourism as their primary industry have been severely affected economically. To revive the local economy and attract new tourists, 3D visualizations with highly detailed 3D building models, 3D terrain together with other 3D city facilities provides an alternative by offering attractive virtual online tours. A successful example is the digital protection of cultural heritage illustrated by (Ulvi, 2021). Relevant departments should think about expanding the small-scale 3D visualization to a city scale.

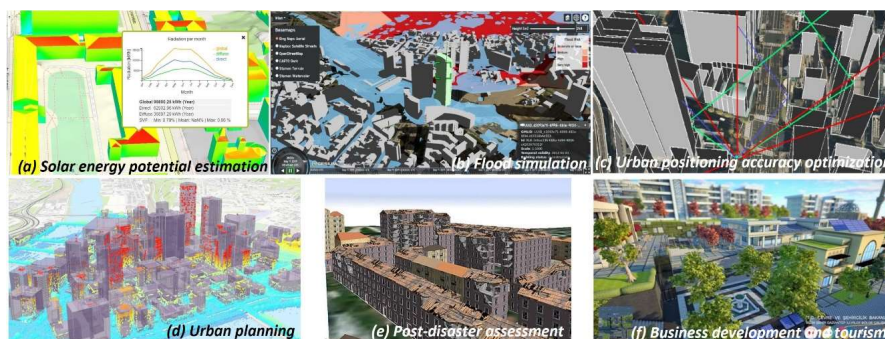


Figure 1.1. Examples of smart city-related applications requiring 3D building models with semantic information. (a) Solar energy potential estimation (Source: solar energy potential in Helsinki, 2023); (b) Flood simulation (Source: Kilsedar et al., 2019); (c) Urban positioning accuracy optimization (Source: Miura et al., 2013); (d) Urban planning (Source: credit from Internet); (e) Post-disaster assessment (Source: Redweik et al., 2017); (f) Business development and tourism (Source: Buyukdemircioglu and Kocaman, 2020).

The applications listed above require a tremendous amount of 3D building models with semantic information. To fulfill this need, in the last decade a number of cities like Berlin and New York have created 3D city models based on the CityGML standard which is freely accessible by the public. These 3D city models consist of buildings, roads and trees as well as parks, water bodies and even digital terrain. They represent the counterpart real-world objects with respect to their geometrical, topological, semantic and appearance properties (Stadler et al., 2009). However, most 3D building models in these open-source datasets are reconstructed in LoD1 or LoD2. LoD3 3D building models unfortunately exist only in some small areas,

Introduction

lacking semantic information, or with semantic information presented as separated attributes rather than being embedded in geometries. There are several reasons. First, existing approaches cannot be directly applied to automatically detect and reconstruct LoD3 building models. Because the input data (like images or point clouds) is heterogeneous and their quality may vary greatly, which lead to the difficulty in developing a general and robust method for object detection from these input data. For example, the classic multi-view stereo algorithm often suffers from painful secondary editing to fix the dense 3D models they produce due to the varying quality of the input images (Furukawa and Ponce, 2009), which is apparently contrary to the original intension of automation. Second, from the modelling point of view, it is challenging to extract the outlines of geometries and adjust the topology among the detected objects at the same time. Third, data collection is expensive, since it needs expensive equipment or sensors as well as plenty of manpower and time. Given the abovementioned challenges, the Nordig Lab has been established as a collaboration between Trondheim municipality and the Norwegian University of Science and Technology (NTNU), in order to generate large-scale LoD3 3D building models with rich semantic information. This is also the primary motivation behind this thesis.

On the other hand, in the last decade of Geographic Information System (GIS) domain, two activities (mapping and geospatial data collection) have been significantly changed from the professional domains to increased involvement of the public by using the power of crowdsourcing. Crowdsourcing is the idea of using human power and wisdom to collect or create data that requires human knowledge not easily performed by computers. The data with geographical locations captured by the means of crowdsourcing is called Volunteered Geographic Information (VGI) (Goodchild, 2007), which has drawn widespread attention since its inception due to its active role of creating, assembling and disseminating geographic data provided voluntarily by individuals (Sangiambut and Sieber, 2016). The ideas of both crowdsourcing and VGI form two rather important concepts of this thesis.

Compared with geographic data obtained through conventional means, VGI data has the characteristics of high timeliness, rapid dissemination, rich information and low cost. It can adapt to the trend of fast information dissemination and high demand for real-time data production, becoming a significant supplement to professional geographic information (Zhou et al., 2018). Since the data acquisition process lacks standardized constraints, which may lead to uneven data quality and excessive data noises, thus numerous studies (Juhász and Hochmair, 2016; Barrington-Leigh and Millard-Ball, 2017; Zhang and Malczewski, 2018; Mohammadi and Sedaghat, 2021) have been carried out to assess the quality of VGI data in terms of positional accuracy, completeness, temporal quality, logical consistency and usability. Their research findings reveal that the quality of VGI data has great inhomogeneity on a global scale, i.e., the data created in developed countries has high quality, while the data quality in developing countries is deficient. There are many reasons for this fact, including project popularity, factors of language, culture and politics, and even Internet infrastructure,

etc. All these reasons may bring uncertainties to the development of VGI projects. But the data quality has been continuously improving over time, as errors or low-quality data can be quickly modified by other contributors. Taking OpenStreetMap (OSM) as an example, the data located in Europe (particularly West Europe) and North America is comparable to authoritative and commercial mapping data in all aspects except for positional accuracy. For points of interest (POI), OSM data is even superior to authoritative mapping data in terms of completeness, accuracy and update frequency. In recent years, VGI data has been increasingly used as important data in research relevant to geography and surveying because of its advantages of wide coverage, free access and relatively high quality. Hence, it is possible to apply VGI data into this thesis for 3D building reconstruction.

There are a couple of widely used VGI data sources nowadays. For instance, 2D OSM (aiming to map the world and distribute geographic data to the public for free) can be assisted in 3D building modelling (Wang and Zipf, 2017); Flickr and Mapillary offer street-view images and they can be employed for urban scene understanding (Neuhold et al., 2017), and in turn, benefiting other applications like autonomous driving (Paz et al., 2020); Twitter data can be utilized for spatiotemporal and semantic analysis (Steiger et al., 2016). However, the majority of existing VGI data is in the form of 2D vectors, images or texts, while 3D VGI data sources are not sufficient.

Regarding the 3D VGI data, Google 3D warehouse (2023) is the most famous project for sharing 3D models by means of crowdsourcing. Although this project offers both geo-referenced 3D buildings and non-geo-referenced objects such as furniture and trees, users must have a certain level of 3D modelling knowledge and skills for data contribution. From this perspective, researchers therefore cannot expect much contribution of 3D building models through data-sharing platforms. Furthermore, several other projects generate 3D building models from OSM data, such as OSM-3D (Goetz and Zipf, 2013), OSM2World (Knerr, 2019) and OSM Buildings (2023). However, most of these building models are only modelled at the coarse level of details (LoD1 or LoD2), and thus lack the detailed façade structures, not to mention the semantic information. The roof geometries of LoD2 models are not accurate as well. Flickr and Mapillary are two other famous VGI image data providers for 3D reconstruction, as smart devices with cameras are currently becoming increasingly powerful and cheaper, in particular for cell phones. Anyone is capable of acting like a sensor to contribute photos via cell phones, digital cameras or even a GoPro. By applying dense image matching algorithm (Agarwal et al., 2011) on a set of street-level images taken around the buildings, it is possible to reconstruct 3D building models from imageries. But this is still at the preliminary experimental stage. The main reason is that images are uncalibrated and have a variety of illumination, resolution and image quality. Additionally, images have unclear positional accuracy and may result in the reconstructed 3D building models having incorrect geographic coordinates. Consequently, all these challenges increase the difficulties in dense image matching and the 3D model reconstruction afterwards.

Introduction

To summarize the major challenges discussed above, they can be categorized into two aspects: (i) 3D buildings reconstruction using existing approaches and (ii) VGI applying for 3D building models reconstruction.

The first aspect involves two issues: (a) difficultly automated 3D reconstruction; (b) high-cost data acquisition. For point (a), as Musialski et al. (2013) pointed out, completely automatic modelling is known to omit user interaction and it is generally accepted that it cannot produce satisfying results in case of erroneous or partially missing data. Hence, inspired by the OpenBuildingModels project (Uden and Zipf, 2013) and the work of Wolberg and Zokai (2018), in which introduced user interaction to overcome geometry incompleteness, it could be a possible solution by applying an interactive method for 3D reconstruction. As for the point (b), the success of Mapillary has proved that the idea of crowdsourcing is a quite good fit for VGI data collection by both experts and laymen. Although it is difficult to guarantee the same quality of VGI data in any geographical regions due to the individual differences among contributors, the convenience of VGI data acquisition and richness of data quantity can compensate for the shortage of data quality to some extent (Zhang et al., 2021), thus resolving the issue of high-cost data acquisition. Therefore, using VGI data by the power of crowdsourcing could be a good alternative solution.

According to above description, the second aspect reflects that most existing 3D VGI building models are reconstructed in LoD1 or LoD2, leading to the façade elements (e.g., windows, doors and balconies) missing problem. To overcome this issue, street-level images can be utilized to provide rich façade information for reconstructing LoD3 3D buildings. Furthermore, in terms of modelling speed and intelligence, deep learning-based methods, especially Convolutional Neural Networks (CNNs), have achieved state-of-the-art results in several image-related tasks, like object detection, semantic segmentation, etc. Hence, it is reasonable to adopt deep learning technique to automatically detect façade elements from VGI street-level images. This is able to shift the main task of volunteers from tedious manual 3D drawing to simple options selection and manual error correction, thus greatly reducing the workload. Each detected façade element can be given a label as semantic information, which is very helpful in generating 3D buildings with semantics. Following this idea, an enhanced YOLOv3 network (Redmon and Farhadi, 2018) for façade elements detection is trained from the scratch on a façade image training dataset — FacadeWHU (Kong and Fan, 2020), which covers 900 street-level images (850 images in Paris, France and 50 images in Trondheim, Norway). By employing this solution, it can not only resolve façade elements missing problem but also ensure that the generated LoD3 building models contain semantic information.

As a result, all these three possible solutions drive the thesis to develop a web-based interactive system (called VGI3D) for LoD3 3D building reconstruction with semantics from VGI street-level images. OpenStreetMap is also embedded in the system to provide building footprints, ensuring that the final reconstructed 3D buildings have a real geographic scale and

real geographic coordinates. The reason for developing a system in the web environment is that it eliminates the need for additional environmental configuration and allows multiple users to perform 3D editing simultaneously online. This stands in contrast to most existing desktop-based 3D building modelling systems, such as Kim and Han's work (2018). The user interface and interaction are designed to be as simple as possible, aiming to reduce the complexity of operation and to make it more user-friendly to ordinary users/non-experts. That can also lower the difficulty of promoting the system to the mass market, which is beneficial for the dissemination and development of the VGI3D. The ultimate goal of VGI3D is to become a VGI platform for collecting 3D building models with semantic information.

Moreover, it is important to note that roof structures are normally invisible in the user uploaded street-level images. In VGI3D, roof models are automatically generated by selecting a pre-defined roof type, but their geometries are usually inaccurate. In some cases, buildings with complex roof structures are unfortunately excluded in the pre-defined options. Although users can manually draw the outlines of rooftops from publicly available bird-view satellite imageries and then generate 3D roof models by incorporating elevation information, the resolution of public satellite imageries is usually not very high. Additionally, the traced rooftop outlines may not be very accurate due to factors such as trees obstruction or other obstacles, thereby affecting the geometric accuracy of reconstructed 3D roof models. To improve the accuracy of reconstructed roof models, airborne laser scanning (ALS) point clouds could be an appropriate data source for accurate roof structure reconstruction because of its high precision on a large scale.

Before reconstructing 3D roofs, the first step is to segment roof planes from ALS point clouds so as to transform a set of points into polygons more easily. Each roof plane then would have unique semantics and attributes (e.g., area, tilt, azimuth, etc.), which are crucial for certain applications like solar energy potential estimation. However, roof plane segmentation is a complex task, since point clouds carry no connection information and do not provide any semantic features of the underlying scanned surfaces (Gilani et al., 2018). The existing methods for roof plane segmentation are mostly based on clustering, model-fitting and region-growing. They either rely on human intervention to select the appropriate input parameters for different datasets or they are not automatic and efficient (Zhang and Fan, 2022). Therefore, the utilization of these conventional approaches is restricted.

Inspired by the part segmentation on ShapeNetPart dataset (Yi et al., 2016) conducted by Mao et al. (2019) based on CNNs, it has shown great potential to apply CNNs to the roof plane segmentation task. Part segmentation aims to segment meaningful parts from an object, for example, a chair mainly consists of three meaningful parts (legs, seat and back). The task of roof plane segmentation is similar to part segmentation, but will further segment a coarse part (e.g., legs) into a couple of separated instances (i.e., independent legs). In general, the results of CNN-based methods rely on the training datasets. Consequently, to implement the

Introduction

task, it is essential to train a CNN on the general and large-scale ALS point cloud dataset with roof plane annotations.

Publicly available ALS point cloud datasets are region dependent and hence roof structures in different regions vary greatly, for instance the Vaihingen 3D Benchmark (V3D) (Rottensteiner et al., 2012), RoofN3D (Wichmann et al., 2018), DublinCity (Zolanvari et al., 2019), the Hessigheim 3D Benchmark (H3D) (Kölle et al., 2021). Amongst them, only RoofN3D includes the labels of roof planes. But the average point density of RoofN3D is only around 4.72 points/m^2 , which is way lower than the standard point density ($10\text{--}12 \text{ points/m}^2$) collected by current mainstream ALS equipment. Additionally, the roof types of RoofN3D are too simple and limited. It does not cover the typical roof types in West Europe (e.g., corner element, cross element and T-element; Kada, 2007), as it was captured in New York, US. That means the network trained on RoofN3D cannot be general enough to segment the typical roof structures of West Europe from the most standard ALS point cloud data.

To close the above research gap, this thesis establishes a new roof dataset (named RoofNTNU) from ALS point clouds to facilitate roof plane segmentation. The initial ALS point clouds were collected by Trondheim Municipality in 2018, with the standard point density. The RoofNTNU contains 930 roofs and covers seven types of typical roof structures in West Europe. Every roof is manually segmented and carefully annotated with distinguishable plane labels by using CloudCompare (2022). Although the training dataset gets ready, existing deep learning-based methods cannot be directly employed to segment roof planes owing to the considerable differences in objects features of indoor scenes. Therefore, this thesis proposes an improved multi-task pointwise network (called RoofNet) based on ASIS network (Wang et al., 2019b) to segment building roof planes. RoofNet is then trained on the RoofNTNU. As a result, detailed façade models combined with accurate roof models can eventually get high-quality 3D building models at LoD3 with semantic information.

As mentioned above, the ambition of this thesis is to make VGI3D become a VGI platform for collecting 3D building models with semantic information, by the power of crowdsourcing. In fact, it is rather challenging to realize this goal, since most existing VGI projects (e.g., Foursquare, Geo-wiki, Moovit, Wikimapia and OpenBuildingModels) have limited influence in the world except OSM and Mapillary, and some of them even have been shut down such as Clickworkers-be a martian and Panoramio. Both the data itself and their contributors are confined to a specific region or nation, far from reaching the scale of global coverage. The main reason is that these platforms and their collected data almost completely overlap with OSM and Mapillary. Users usually have a preconceived idea and are more inclined to contribute data to the most influential and mature platform, thus making it difficult for those similar VGI projects with less influence to draw the attention of new users. Moreover, as the representative of VGI image platform, Mapillary is actually developed on the basis of OSM. In its early years, the main contributors were the same group of people as OSM. They used

the concept of global collaborative mapping to collect street view data as a valuable supplement to OSM data. From this, it can be seen that core users are of great help to the development and promotion of the VGI platform. Therefore, for a new VGI platform to survive, it has to capture geographic information data that is completely different from OSM and Mapillary, such as 3D data. By doing so, project creators can motivate and attract volunteers to participate and increase the reputation, even seizing core users from other existing VGI platforms. While there have been several VGI platforms in recent years that collect 3D buildings and 3D drone data, the number of contributors is scarce. The main constraint on the development of these platforms is the limitation of interactive visualization. The data uploaded by volunteers cannot directly connect with a virtual 3D geographic scene, resulting in a lack of immediate visual satisfaction. This greatly weakens the sense of accomplishment for contributors, thereby losing their motivation to participate in the project and contribute data. Additionally, people's understanding towards 3D building models is still at the stage of 3D visualization, and detailed 3D buildings with semantic information have not been really applied to practical projects. Hence, people are currently satisfied with the triangular meshes with textures of 3D buildings found in Google Earth, and they do not think it is necessary to contribute 3D building data on an unknown platform.

Considering the above reason analysis and motivating volunteers to contribute 3D building models through VGI3D, this thesis develops a web-based 3D visualization platform where digitalizes a real 3D city environment by integrating 3D terrain (DTM), 3D road networks, 3D building models and other road-related 3D objects (e.g., traffic lights & signs and trees). Volunteers are able to interact with the building models not only created by themselves but also by other contributors, and can also inspect the semantic information of 3D objects in the 3D visualization platform. Furthermore, by exploring the areas without 3D buildings in virtual 3D scene, users may feel that they are the pioneers of this virtual world, and every building model they contribute is helping to improve this massive VGI project. That can significantly strengthen the sense of accomplishment for contributors and enhance their psychological comforts at the same time. As a result, the more volunteers participate in the VGI3D project, the higher the popularity and maturity of the project will be. This thesis envisions that in the future, the global coverage of detailed LoD3 3D building models can only be achieved through crowdsourcing, since many impoverished and developing countries and regions do not have the ability and resources to accomplish such a task.

1.2 Research questions and research objectives

The main goal of this thesis is to propose a novel approach for reconstructing 3D buildings with semantic information in a crowdsourcing manner, and to provide volunteers with sufficient motivations to participate in 3D building contributions for the 3D modelling community. For this purpose, the thesis addresses the following original research questions:

Q 1: Can crowdsourcing approaches be utilized for reconstructing 3D building models? How to design the framework of crowdsourcing methods to reconstruct 3D building models with semantics?

A 1: Perform a literature review to study the first subquestion.

From the contributor's perspective, design a new interactive platform for collecting 3D building data to draw the attention of new users, differing from what those existing well-known VGI projects have done. Furthermore, develop this new platform in a web environment to enhance the convenience and interoperability for users when operating and interacting. Meanwhile, connect the reconstructed 3D buildings with a virtual 3D geographic environment for interactive visualization, aiming to give users immediate visual satisfaction as well as the sense of psychological accomplishment after contributions and in turn, to motivate them for further data contributions.

Q 2: How to enable the reconstruction of 3D building models with semantics by means of crowdsourcing?

A 2: Develop a web-based interactive system, VGI3D, for 3D building reconstruction from crowdsourced street-level images based on deep learning technique.

To reconstruct high quality roof models, propose an improved multi-task pointwise network (RoofNet) based on CNN to segment building roofs in point clouds. To train the network, establish a new roof training dataset (RoofNTNU) with seven typical roof types in West Europe and also with the standard point density of ALS point clouds, aiming to provide a general and robust segmentation capability on most ALS cloud data.

Q 2.1: How to ensure that VGI3D has simple interoperability for both expert and non-expert users?

A 2.1: Conduct a usability testing by limited participants who have or have not rich 3D modelling experience and then analyze the data collected from the usability testing in terms of correlation and sensitivity.

Q 2.2: How to analyze the effectiveness of the improved modules of the proposed RoofNet?

A 2.2: Perform a network architecture study including five variant experiments on key improvements.

Q 3: How to motivate volunteers and enhance their psychological comforts for more 3D building contributions using a 3D visualization platform?

A 3: Digitalize a real city environment by integrating base maps, 3D terrain, 3D road networks, georeferenced 3D building models and other road-related 3D objects so as to that placing the 3D buildings created by volunteers into a 3D scene to strengthen their sense of accomplishment and to increase the motivations for more contributions. Furthermore, all these 3D objects except 3D terrain are built on CityGML standard to enhance the interactive capability of the visualization platform.

Q 3.1: How to render massive 3D data rapidly and efficiently in the web environment?

A 3.1: To render high-resolution 3D terrain data efficiently, TIN (Triangulated Irregular Network) and LoD (a scale level defined in a tiling scheme, different from the CityGML's LoD) are adopted as the rendering optimization strategies. 3D Tiles technology is also employed to efficiently render 3D buildings and other 3D data.

Q 3.2: How to render initial 2D road segments on high-resolution 3D terrain in a cartographic manner?

A 3.2: Employ the collision detection to interpolate the vertices intersected with 3D terrain to get 3D polylines. 3D polylines are then expanded to 3D polygons. At the same time, it needs to enable 3D polygons to be as smooth as possible at the bend.

Q 3.3: How to enrich the 3D road-related objects of the visualization platform, such as traffic lights and signs at road intersections?

A 3.3: Present an approach to automatically detect road objects from street-level image sequences and place them to correct locations according to a set of summarized urban rules.

Figure 1.2 illustrates the research questions in context of different boundary levels.

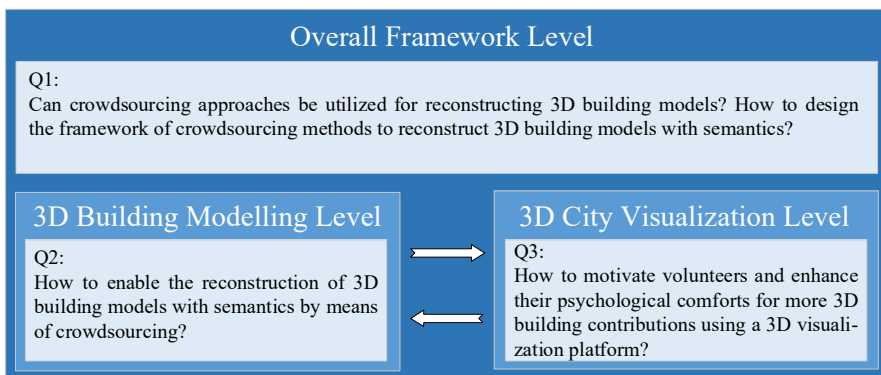


Figure 1.2. Research questions in context of the different boundary levels.

1.3 Thesis organization

The remaining chapters of this thesis are organized as follows:

Chapter 2 reviews the related work about 3D building model representation, reconstruction methods and 3D visualization standards as well as frameworks. Chapter 3 describes the methodologies of the thesis including interactive 3D building modelling algorithms, roof plane segmentation method for the purpose of rooftop reconstruction and web-based 3D visualization. Chapter 4 presents the implementation details of the system or proposed approaches and main experimental results of the papers. Research questions are discussed as well. Chapter 5 summarizes the research and gives directions for future research.

1.4 List of publications

Four journal papers constitute the foundation of this doctoral dissertation. They are presented below together with my contribution of each paper.

Paper 1:

Zhang, C., Fan, H., & Kong, G. (2021). VGI3D: an Interactive and Low-Cost Solution for 3D Building Modelling from Street-Level VGI Images. *Journal of Geovisualization and Spatial Analysis*, 5(2), 1-16.

Contribution: The VGI3D platform was implemented and introduced from the perspective of software engineering in this paper. I implemented all functionalities both for the frontend and backend, designed and conducted the usability testing among expert & non-expert participants in collaboration with Gefei Kong, analysed the testing results and drafted the manuscript. Hongchao Fan proposed the research idea. Revision and editing were done by all co-authors.

Paper 2:

Fan, H., Kong, G., & Zhang, C. (2021). An Interactive platform for low-cost 3D building modeling from VGI data using convolutional neural network. *Big Earth Data*, 5(1), 49-65.

Contribution: The VGI3D platform was introduced from the perspective of geographic information system (GIS) for the first time. Gefei Kong and I made the equal contribution, implemented and evaluated the key algorithms of the VGI3D, wrote the majority of the initial manuscript. Hongchao Fan proposed the research idea. Revision and editing were done by all authors.

Paper 3:

Zhang, C., Fan, H., & Li, W. (2021). Automated detecting and placing road objects from street-level images. *Computational urban science*, 1(1), 1-18.

Contribution: I trained two deep learning models for the tasks of semantic segmentation and object detection, respectively, in collaboration with Wanzhi Li. I summarized and proposed six urban roles to assist the implementation of the methodology. I designed and conducted the experiment alone and visualized the results. Results evaluation and analysis were done together with Hongchao Fan. Finally, I drafted the manuscript. Revision and editing were done by all co-authors.

Paper 4:

Zhang, C., & Fan, H. (2022). An Improved Multi-Task Pointwise Network for Segmentation of Building Roofs in Airborne Laser Scanning Point Clouds. *The Photogrammetric Record*, 37(179), 260-284.

Contribution: I manually established a new roof ALS point cloud dataset (RoofNTNU) with the help of three research assistants. I designed the labels definition for two different segmentation tasks (i.e., instance segmentation & semantic segmentation). I also proposed an improved deep learning model (RoofNet) to segment roof planes, conducted the experiment and visualized the results. The results evaluation and analysis were done together with Hongchao Fan. I wrote the initial draft of the article. Revision and editing were done by all co-authors.

Introduction

Chapter 2

Related work

With the emergence of concepts such as digital twin cities and smart cities, the construction of a comprehensive, high-precision and real-time digital representation of the physical world has become a practical requirement. 3D building reconstruction, as a critical aspect among them, has put forward new demands for both industry and academia, since traditional methods for generating 3D building models can no longer support these new application scenarios and meet new requirements. Hence, additional data sources like VGI data and semantic information need to be added to construct multi-angle representations of 3D building models. In this context, VGI is gaining more attention as an important crowdsourcing means to achieve global 3D buildings coverage. In addition to VGI and 3D building reconstruction, 3D interactive visualization is another significant research aspect of this thesis, in order to directly connect the generated data by volunteers with virtual 3D geographic space, offering immediate visual satisfaction. This can greatly strengthen the sense of accomplishment for volunteers and increase their motivations to participate in global mapping project. The state of the art will be introduced in this chapter through a literature review regarding the VGI, 3D building reconstruction and visualization.

2.1 Volunteered geographic information

Volunteered geographic information reveals differently in geospatial data acquisition by means of crowdsourcing, compared to classic GIS. It refers to geographic information data created by Internet volunteers through interactive web platforms, by directly uploading sensor data or providing digital labor. This behavior is purely voluntary and non-profit-driven. Participants might simply want to make the open-sourced community better and make contributions to the collective cause. The VGI data can be classified into active contribution and passive contribution (e.g., spatial positioning and text data contributed by social media Twitter users), while most of them belong to the active contribution. Therefore, this subsection will only review the active-contribution VGI in terms of existing data platforms and its application in 3D modelling. More relevant works can be found in these published review articles (See et al., 2016; Yan et al., 2020; Lotfian et al., 2020).

2.1.1 VGI data platforms by active contributions

Relying on the crowdsourcing manner, interactive web-based VGI data sharing platforms provide data support and new inspirations for VGI-related researches. With the development

of Internet technologies, these platforms are no longer limited to offer 2D vector map data but to include geotagged street view images, POI data, thematic data such as public transport lines, and so on. An overview of existing VGI platforms is summarized and shown in Table 2.1. From this, it reflects that the VGI data covers the commonly used data in daily use, and demonstrates that these VGI data have good sustainability and practicability. However, apart from a few exceptions like OSM and Mapillary, most VGI data platforms either are not well-known to the general public or are in an inactive operational status due to various reasons, and some have even ceased operations and shut down.

Table 2.1. The existing VGI data platforms by active contributions appeared in VGI-related researches (reproduced from paper (Fan et al., 2022)).

Name	Country	Time	Data Content	Status
OpenStreetMap	UK	2004 – present	Various geospatial map data	Active
Foursquare	US	2004 – present	Geographic locations	Active
Flickr	US	2004 – present	Geotagged photos	Active
Mapillary	Sweden	2013 – present	Street-level imagery	Active
Moovit	Israel	2012 – present	Public transport lines	Active
Panoramio	Spain	2005 – 2016	Geotagged photos	Closed
Be a Martian	US	2009 – 2020	Martian landforms	Closed
OpenSeaMap	UK	2009 – present	Nautical and geospatial data	Active
Wikimapia	Russia	2006 – present	Various geospatial map data	Inactive

2.1.2 VGI as a data source for 3D modelling

Google 3D Warehouse is the first approach for sharing user-generated 3D models, including both geo-referenced objects like 3D buildings, and non-geo-referenced objects such as traffic signs and trees. The non-geo-referenced objects play a significant role in helping improve algorithms of automatic object detection in 3D computer vision. However, volunteers must have a certain level of 3D modelling knowledge and skills for voluntary data contribution (Fan and Zipf, 2016). Thus Google Earth launched Building Maker project for those users who are not experienced in 3D modelling but still want to contribute. This project was designed for generating geo-referenced 3D building models from commercial oblique images in the way of crowdsourcing. Although 3D Warehouse and Building Maker adopt a crowdsourcing manner to collect or create 3D models, the data they used may be commercial/official data, which are usually not free or not easy to access. Hence, several projects (e.g., OSM-3D and OSM Buildings) and study (Bagheri et al., 2019) were conducted for 3D reconstruction from free VGI data, i.e., OSM footprints and remote sensing-derived elevations. Unfortunately, the LoD of the generated building models is at the coarse level and therefore, they do not have detailed façade structures and accurate roof geometries. To

resolve these issues, Uden and Zipf (2013) proposed a web-based platform, OpenBuildingModels, aiming to upload and share complex 3D buildings or other landmarks. However, it was still in the very early stage and many open issues they concluded had not been addressed yet. Additionally, there were no subsequent studies to further follow up. Fan and Zipf (2016) investigated the possibility and presented the preliminary concept of interactively reconstructing 3D building models from OSM footprints and street-level images. Fan et al. (2021) then continued following and implementing the previous concept, by proposing an interactive method for simple LoD3 building reconstruction.

2.2 Current approaches of 3D building model reconstruction

Earth Observation (EO) data (e.g., point clouds, aerial & satellite images) and street-level imageries have been widely employed by traditional and deep learning-based methods for 3D building reconstruction. This sub-section will present the typical conventional and deep learning-based approaches proposed in the last decade and will also discuss their advantages and disadvantages. More relevant work can be found in these published review articles (Musialski et al., 2013; Liu et al., 2021).

2.2.1 Traditional methods

The popular traditional methods for 3D building reconstruction can be classified into two major categories: model-driven, data-driven methods.

Model-driven methods

The model-driven approaches firstly pre-define a set of parametrized roof models to form a catalog and then match the generated roof geometries from this catalog to derive the best fitting roof model. For instance, Henn et al. (2013) proposed a robust and automatic method to reconstruct LoD2 building models from pretty sparse LiDAR point clouds together with 2D building footprints. Boosting by the MSAC (M-Estimator Sample Consensus) and SVM (Support Vector Machines), this method was capable of enabling estimated geometric primitives not only topologically correct but also satisfying symmetry and orthogonality constraints. Xiong et al. (2014) aimed to resolve the problems of small roof faces, low point density and noise occurring in the process of 3D building reconstruction, and presented a flexible graph edit dictionary combined with 15 pre-defined building primitives to automatically identify and fix errors when creating roof topology graphs. As a result, they achieved around 95% of correct reconstructed building models in the test areas. Buyukdemircioglu et al. (2018) even defined a larger catalog containing 32 different building primitives. At the same time, they also developed a semi-automatic system to reconstruct LoD2 building models from stereo aerial imageries. However, these three example methods

Related work

can only obtain promising performance on individual building primitive and cannot handle well with compound buildings with multiple primitives. To address this issue, Li and Shan (2022) presented a two-step multi primitive reconstruction framework based on RANSAC (RANdom SAmple Consensus; Fischler and Bolles, 1981) from point clouds. The proposed approach would firstly segment the combined building into a few simple predefined building primitives and secondly further segment these primitives by applying RANSAC and holistic primitive fitting. Lastly, the 3D Boolean operations were utilized to generate 3D building models, which could keep the consistent topology with their compositional primitives. Consequently, this method makes it possible to reconstruct building models with complex roof structures to some extent.

The 3D building models reconstructed by above approaches are always at LoD2 and hence cannot meet the requirements of some other applications that need the 3D buildings with detailed façade structures (i.e., LoD3 building models). For this purpose, some researchers have explored novel methods on model-driven reconstruction of building facades. Similar to pre-defined building primitives, a façade can also be formulated by using synthesis of templates or grammatical rules (Müller et al., 2007). Amongst them, grammatical rules (also known as façade grammar) are the most widely used for façade reconstruction. Koutsourakis et al. (2009) summarized and proposed a set of shape and parametric rules as generic façade grammar. Then Markov Random Field (MRF) was built for each rule by assigning a probability. In this way, the method could reconstruct 3D building model with detailed façade structures from single street views. However, it is challenging to design a large number of complicated façade grammar to handle with complexity and diversity of building facades. Instead of predefining grammar rules, Dehbi and Plümer (2011) innovatively attempted to learn semantic models and grammar rules of building façades from precise descriptions and noisy observations (e.g., point clouds), in which the learning of topological and geometric constraints was also included. The proposed approach was proven the feasibility through various experiments and achieved reasonable results. Gadde et al. (2016) also adopted the same idea, but learned very general and compact grammar from a set of images by conducting a clustering algorithm during the grammar learning. As a result, they obtained better façade parsing results compared to other grammar-learned methods as well as handcrafted grammar rules.

In summary, model-driven methods can ensure the reconstructed LoD2 building models are topologically correct (mainly refers to the roof parts), which are generated from such as ALS point clouds or aerial images. But problems may occur if there is no candidate for the roof shape in the predefined catalog (Buyukdemircioglu et al., 2022). Regarding the LoD3 building models that are generated from street view imageries, terrestrial laser scanning (TLS) or mobile laser scanning (MLS) point clouds by means of façade grammar, as Wang et al. (2020) pointed out, the established grammar rules usually focus on the arrangement of elements on the facade and have a weak robustness to deal with occlusions, shadows and

specular reflections. In addition, model-driven approaches employ a restricted pre-defined building primitives or façade grammar, which may lead to the accuracy reduction and may not be able to reconstruct complex roof structures.

Data-driven methods

The data-driven approaches assume that a building is an aggregation of several segmented roof planes (Tarsha-Kurdi et al., 2007), and usually begin with a segmentation of the building ALS point clouds into individual roof planes and then assemble the roof planes to comprise polygonal building models (Li and Shan, 2022). Therefore, most of data-driven methods are involved in the building roof segmentation from ALS point clouds or unmanned aerial vehicle (UAV) photogrammetric point clouds, and they can be categorized as clustering-based, model fitting-based and region growing-based methods.

Clustering-based methods are basically considered as unsupervised learning where points are classified into distinguishable primitives based on pre-computed local surface features or properties. Kong et al. (2013) proposed a novel combination of the K-plane and K-means algorithms aiming to produce high-precision segmentation of roof structures. Additionally, an improved initialization method was used to acquire the better initial clustering centres for the K-means algorithm. Wang et al. (2019a) presented a new roof plane segmentation algorithm based on the DBSCAN density clustering technique. The optimal searching radius ϵ of DBSCAN could be automatically estimated through the intrinsic properties of the point cloud data. Despite the popularity of clustering-based approaches for segmentation, they are sensitive to noise and outliers, and have difficulty in neighborhood selection as well as being computationally expensive for multidimensional features in large datasets.

Model fitting-based methods consist of two common used algorithms: the Hough Transform (HT) (Ballard, 1981) and the RANSAC. In the 3D HT for plane detection, all LiDAR points are first mapped to parameter space, in which each point corresponds to a roof plane in object space. Voting accumulators are then applied to count the points falling into the corresponding planes by searching for the local maximum values in the parameter space. These accumulators are the detected shapes in the object space. Nevertheless, the HT is sensitive to the selection of parameter values and inefficient for computational time (Nguyen and Le, 2013). The standard RANSAC method typically fits a best mathematical plane with the most inliers from the LiDAR points and then considers this mathematical plane as the detected planar shape. Malihi et al. (2018) presented a two-level segmentation approach for 3D building reconstruction from UAV point clouds by combining RANSAC with contextual as well as conceptual knowledge. Since the initial UAV imageries were oblique, their work could reconstruct LoD3 3D building models with an acceptable geometric accuracy. Although RANSAC is robust on datasets with a large amount of noise and outliers, the issue of the spurious planes cannot be completely solved.

Region growing-based methods start with a chosen seed region/point and iteratively expand to the seed's neighboring points until the growing meets the criteria. For instance, Nurunnabi et al. (2012) selected the seed points with the least curvatures in local surfaces, and then employed angle differences between normal vectors, the distance of points to planes, and distance between two points as the criteria for the growing. To improve the computational efficiency and robustness of region growing, Xu et al. (2017) presented a voxel-based region growing method with robust principal component analysis for building roofs segmentation. Both qualitative and quantitative results surpassed the representative algorithms. However, many studies (Li et al., 2020; Shao et al., 2021) have revealed that compared with clustering-based methods, although region growing-based methods are more robust to outliers and noise, it is challenging to accurately determine the boundaries between adjacent planes and may tend to over- or under-segment.

In Summary, the main issue of data-driven methods is that the segmented planar patches may not be intersected properly and thus resulting in topological or geometrical errors (Buyukdemircioglu et al., 2022). Moreover, data-driven methods either depend on human intervention and prior knowledge to select the appropriate input parameters for different datasets, or they are not automatic and efficient enough during the process of segmentation.

2.2.2 Deep learning-based methods

Benefitting from the advances in deep learning technique as well as the availability of 3D shapes training datasets, the 3D building reconstruction has been revolutionized through several specific tasks such as object detection, segmentation, classification, etc. Deep learning-based methods, in particular Convolutional Neural Networks (CNNs), can greatly break through the bottlenecks of traditional methods by automatically learning features of 3D shapes from aerial/satellite imageries or point clouds.

CNNs were applied into 3D building reconstruction for the first time by Wang and Frahm (2017). By learning joint features from satellites images, vector map and LiDAR data during the training phase, their approach was able to reconstruct parametric building models in LoD1. To generate LoD2 building models, apart from training on orthophotos, Alidoost et al. (2019 & 2020) also fused digital surface models (DSM) as well as 2D rooflines into training and ensured that the reconstructed 3D buildings had simple estimated roof structures, but with low geometric accuracy. To improve the geometric accuracy of building models, Gui and Qin (2021) further optimized existing work and proposed an automate CNN-based approach for LoD2 models reconstruction from orthophotos and high-resolution satellite-derived DSM, following by a “decomposition-optimization-fitting” paradigm. This method was evaluated on several testing datasets with different urban patterns and gained the high-quality reconstruction results.

RoofN3D as the first public ALS point cloud training dataset, it provides not only distinct classes for buildings but also plane labels of each roof, for the purpose of 3D building reconstruction using deep learning technique (Wichmann et al., 2018). Since it was released, researchers started attempting to directly learn features from point clouds and then generate 3D building models without the assistant of other data. Thus, Zhang et al. (2021) trained the well-known PointNet++ (Qi et al., 2017) on RoofN3D for building primitive recognition, and then utilized a holistic primitive fitting approach to generate LoD2 building models from point clouds in a model-driven way. However, the point density of RoofN3D is too low, only around 4.72 points/m², and hence the proposed method may not have a general capability to be applied to other point clouds with standard point density. In a recent study, an end-to-end network, Point2Roof, was proposed by Li et al. (2022). It could directly reconstruct building roofs from ALS point clouds by identifying the corner points of roof planes using the learned deep features. Although this method is more concise and efficient than some existing approaches, the Point2Roof is trained on a synthetic dataset containing 16 different roof types handcrafted by authors themselves, which is different from the real datasets. Therefore, more evaluations on real datasets are necessary.

However, automatic generation of high detailed LoD3 models using CNNs is still a challenging topic for researchers. To overcome this issue, some researchers take the interactive modelling into account, so that reconstructing LoD3 buildings with the help of CNNs. Thus an interactive system was developed by Nishida et al. (2018), with which users required to outline the boundary of building façade from a single building image. A façade grammar then would be automatically generated for procedural modelling of buildings using CNNs. Additionally, Liu et al. (2022) presented a translational symmetry-aware façade parsing method and achieved rather photorealistic 3D building models from one façade image. They automatically parsed the façade elements by combining semantic segmentation and instance detection and then interactively generated 3D models through procedural modelling. However, users were asked to repeatedly interact with the system, and gradually sketched and added floors, ledges, roof, windows, doors and balconies, respectively. Frankly speaking, it is not user-friendly to non-experts, since they may feel the interaction process tedious and thus may lose their patience.

2.3 Visualization of 3D building models

CityGML is mainly designed as a conceptual model and exchange format for the representation, storage and exchange of virtual 3D city models, even though it can be a supplement to the visualization standards/formats such as 3D Tiles (2023), glTF (2023) or COLLADA (Barnes et al., 2008). Therefore, dedicated 3D visualization technologies are needed to visualize 3D building models in an efficient manner.

2.3.1 Standards for 3D visualization

3D computer video games have become the most successful implementation of 3D visualization by high-definition object modelling and texture mapping, the realistic simulation of lights and shadows, as well as excellent rendering optimization. This kind of visualization is directly implemented on the hardware through calling a series of sophisticated graphic APIs (Application Programming Interface) provided by game engines. The space size of a game engine together with its all essential dependencies and material assets is commonly up to dozens of gigabytes (GB) and they also have high demands for computing performance of client devices, particularly powerful GPU (graphics processing unit). The better GPU, the more expensive. Hence, for 3D building models that need to be visualized by anyone from anywhere on any devices, it is not a smart choice to implement 3D visualization using a game engine, despite its impressive rendering capability.

With the enhancement of data transmission through Internet and the development of Web Graphics Library (WebGL), it becomes possible to visualize 3D models online and hence several popular 3D standards for online visualization are proposed. They provide sufficient APIs for 3D scenes rendering and user interaction. As a result, the visualization of 3D building models would have no limitations of platforms, but only needs a browser with WebGL. Currently, there are a couple of popular international 3D visualization standards, for example:

- *glTF*. It is an open standard that supports the efficient transmission and loading of 3D scenes and models by engines and applications, like games, virtual reality (VR) or web applications. *glTF* can minimize the size of 3D assets and the runtime processing needed to unpack and use them.
- *KML* (2023). It is designed and presented by Google used for displaying geographic data and visualization on existing or future web-based online and mobile maps (2D, e.g., Google Maps) in and earth browsers (3D, e.g., Google Earth).
- *3D Tiles*. It is designed by Cesium for streaming, sharing, visualizing, fusing, and interacting with massive heterogeneous 3D geospatial content (e.g., 3D buildings, point clouds) across desktop, web, and mobile applications. Built on *glTF* and other 3D data types, 3D Tiles has proven its strong ability on dealing with today's ever-growing geospatial data and also reveals its huge potential of overcoming more complex challenges in the future.
- *CityJSON* (Ledoux et al., 2019). It is a JSON-based encoding for storing 3D city models, aiming to offer a compact and developer-friendly standard so that files can be easily visualized, manipulated, and edited on the web-based applications.
- *COLLADA*. It is an open standard XML-based schema for interactive 3D applications, enabling it easier to transport 3D digital assets between different graphics applications without loss of information. *COLLADA* is also compatible across multiple platforms.

2.3.2 Web-based 3D visualization frameworks

Online 3D building model visualization belongs to the category of 3D WebGIS. Now there are two main technical routes to implement 3D WebGIS. The first one is to add plugins into the browser or to develop some special components in the operating system in order to facilitate the interactive visualization, and the second one is to rely on the latest WebGL rendering mechanism of the browser for upper-layer development (Xu et al., 2020). The whole framework is divided into frontend (client side) and backend (server side). The frontend is used for displaying geospatial data and providing user interaction while the backend is responsible for processing the request events, communicating with databases and doing relevant logical computation. Moreover, the network communication protocol as a bridge is utilized to connect frontend and backend. The 3D WebGIS frameworks employing browser's built-in WebGL are currently popular ones because they do not need users to install additional plugins, greatly simplifying the environment configuration process. The convenience and applicability of 3D WebGIS with WebGL has been confirmed by practical application of real-time urban energy simulation in the Hammarby district of Stockholm, Sweden (Mao et al., 2020).

When displaying large amount of 3D data online, web-based 3D visualization frameworks often face the problems such as slow speed of data loading or browser crash due to inherent limitations in browser memory and network bandwidth. Thus some previous studies fixed these problems from the perspective of 3D building generalization (Zhao et al., 2012; Baig and Rahman, 2013), and then visualized the simplified 3D buildings on the web side using existing rendering technologies. These solutions are indeed able to address the problems to a certain degree and improve the visualization performance, but at the expense of the geometric or semantic integrity of 3D building models. In other words, these solutions do not fundamentally solve the problems.

To render massive 3D building models while remaining their geometric or semantic integrity, Cesium presented the 3D Tiles technology for the first time in 2015 and became an international standard afterwards. After that, the combination of 3D Tiles and WebGL has constituted a promising solution for efficiently rendering 3D geospatial data, not just 3D buildings. Song and Li (2018) proposed an improved 3D Tiles dynamic loading and scheduling strategy based on terminal memory and screen space error in order to rapidly and efficiently visualize massive 3D city models. Kulawiak et al. (2019) also applied 3D Tiles and other open-source technologies to present a framework for remote processing, integration, storage, visualization, and dissemination of 3D LiDAR in a web environment. Observing the challenges of visualizing BIM models in the WebGIS environment, Xu et al. (2020) explored a method by which BIM models can be transformed from Industry Foundation Classes (IFC) into 3D Tiles while ensuring their complete semantic attributes. Then a 3D WebGIS framework was developed for BIM model visualization based on 3D Tiles. Recently, Lu et al. (2021) even extended the application to meteorology field and created a new framework to

Related work

perform real-time 3D visual analytics of large-scale weather radar composing data taking advantages of 3D Tiles and WebGIS.

3D Tiles shows the great advantages over other standards because of its powerful capability on rapidly rendering massive 3D geospatial data and being able to reduce memory consumption. Therefore, this thesis also adopts 3D Tiles standard to realize the fusion and visualization of multiple 3D georeferenced datasets in a web environment.

Chapter 3

Methodology

This chapter begins by briefly explaining the overall research framework of the thesis with three main modules. Then detailed description of each module regarding the methodology is introduced in Section 3.2 – 3.4, respectively.

3.1 Research framework

Reconstruction of 3D building models with high LoD and semantic information is a complex task which usually involves several steps including façade elements detection, roof plane segmentation, topology adjustment of detected/segmented objects, extraction of key points & edges, and georeferencing. These steps are primarily employed in the process of detailed façade and accurate roof modelling. To attract more volunteers to interactively create 3D buildings by means of crowdsourcing, it is necessary to simplify the process of interactive operations from user side. In addition, it is also of great help to integrate the generated 3D buildings into a virtual 3D city environment with other 3D city objects for better user experience and interaction, and in turn, the 3D visualization platform enables to increase their self-satisfaction and to stimulate more motivations to contribute 3D data.

In this thesis, novel 3D reconstruction methods on façade level and rooftop level are studied thoroughly and are jointly utilized to generate high-quality 3D building models with semantics in LoD3. On the façade level, VGI3D, an interactive and low-cost solution, is presented for 3D building modelling (particularly detailed façade reconstruction such as windows, doors and balconies) from VGI street-level images using CNN. The VGI3D is web-based and its user interaction is designed as simple as possible. It also offers an interactive updating function if façade elements are incorrectly detected due to complex scenes of images such as distortion, blur, occlusion or bad illumination. On the rooftop level, an improved deep learning-based network, RoofNet, is proposed to segment roof planes from ALS point clouds. Meanwhile, a new roof dataset with seven typical roof types in West Europe, RoofNTNU, is established by taking standard ALS point clouds as training data for automatic and more general segmentation. All generated 3D building models are georeferenced with the real scales and correct geospatial coordinates. Following the CityGML2.0 standard, 3D buildings together with their semantics are then represented and stored as CityGML format, which would be convenient for data exchange/transformation on 3D visualization platform afterwards.

To verify and access the effectiveness of the proposed methods, the 3D modelling results are then evaluated by visual comparison and several commonly used quantitative metrics in deep

Methodology

learning, respectively. The former is to manually compare real buildings with reconstructed ones in terms of completeness, visual plausibility and geographic location. The latter is to automatically test the accuracy/precision in the process of object detection or segmentation, since they are the critical steps of the 3D reconstruction. If the errors produced by them are less, the overall accumulated errors of the generated models would also be less accordingly. In addition, usability testing and user survey in the form of questionnaire are also conducted on VGI3D by limited non-expert/expert participants from the perspective of software engineering, which is intended to verify the usefulness and simple interoperability of VGI3D and to better optimize the system and user experience at the same time. The data collected from the questionnaire is then further discussed through data and correlation analysis. Both analyses aim to deeply mine the hidden findings in data answered by participants with different backgrounds.

Last but not the least, a web-based 3D visualization platform integrated with multiple 3D city models is implemented, aiming to allow volunteers to interact with their generated 3D buildings in a virtual city environment and in turn, to increase their motivations for more contributions of 3D buildings using VGI3D. The 3D visualization platform is integrated with high-resolution DTM, 3D buildings coming from three different data sources, 3D road networks and road-related objects, and applies 3D Tiles technology as the rendering strategy for a rapidly display of massive 3D data. Because it is difficult for the browser to render and load a large amount of 3D data at one time. All of 3D city models except 3D terrain (stored as quantized-mesh format) are converted from CityGML into b3dm (Batched 3D Model) tiles and are visualized in the user browser through an asynchronous JavaScript library (CesiumJS). Besides, to improve the content richness of the 3D visualization platform, a case study focusing on road-related objects, specifically traffic lights and signs at the intersections, is conducted. The study aims to explore automated method for detecting and localizing these objects from street-level image sequences based on urban rules. The overall research framework of this thesis is illustrated in Figure 3.1. The rest of this chapter will introduce each part of proposed research framework in details. More specifically, the thesis introduces three main components, namely, detailed façade modelling, accurate roof modelling and web-based 3D visualization platform. In the component of accurate roof modelling, it focuses on roof plane segmentation as an essential step in the process of roof reconstruction.

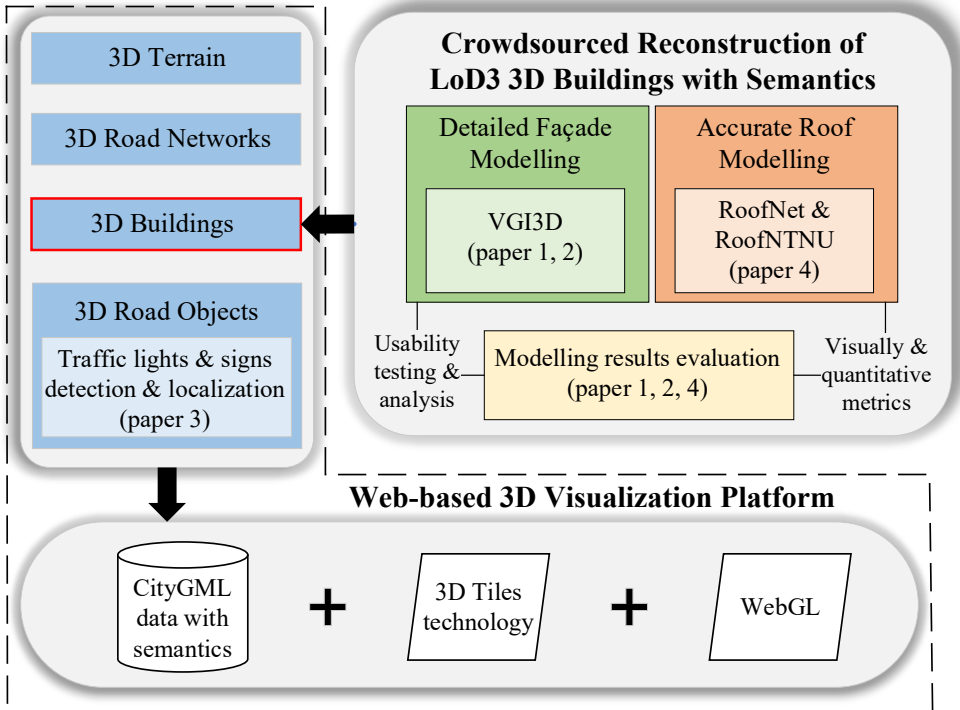


Figure 3.1. The overall research framework of the thesis.

3.2 Detailed façade modelling

The algorithms embedded in VGI3D for reconstructing LoD3 3D buildings with semantic information will be explained and described in this section.

3.2.1 Workflow

A typical workflow of the VGI3D is as follows:

- (1) Users upload all images belonging to the same building but with different façade orientations.
- (2) For each image, users draw the façade boundary, select its corresponding footprint edge from OpenStreetMap, pick a predefined roof type, enter the number of floors to estimate the building's height and then click "Save" button; repeat this process until no more images need to be handled.
- (3) Images are transferred into the module of façade elements extraction to automatically obtain the locations of windows, doors and balconies under a 2D image coordinate system, and assign a semantic label to each façade element.

Methodology

- (4) These locations obtained in (3) are transformed from 2D into the 3D coordinate system to construct 3D building model and display it on the Web side in real time.
- (5) The generated 3D building model can be exported in CityGML/OBJ format by clicking “Download” button.

Furthermore, some façade elements may be incorrectly detected because of low-quality input images or occlusion. Hence, interactively updating 3D models is necessary. For this purpose, VGI3D allows users to delete the wrongly reconstructed façade element, outline the boundary of incorrect façade element upon the image and select a corresponding element type for it. After that, the updating algorithm would automatically fuse the new façade element into the original 3D building and adjust element’s location to make the entire model look coordinated and harmonious. The abstract overview of the workflow is shown in Figure 3.2.

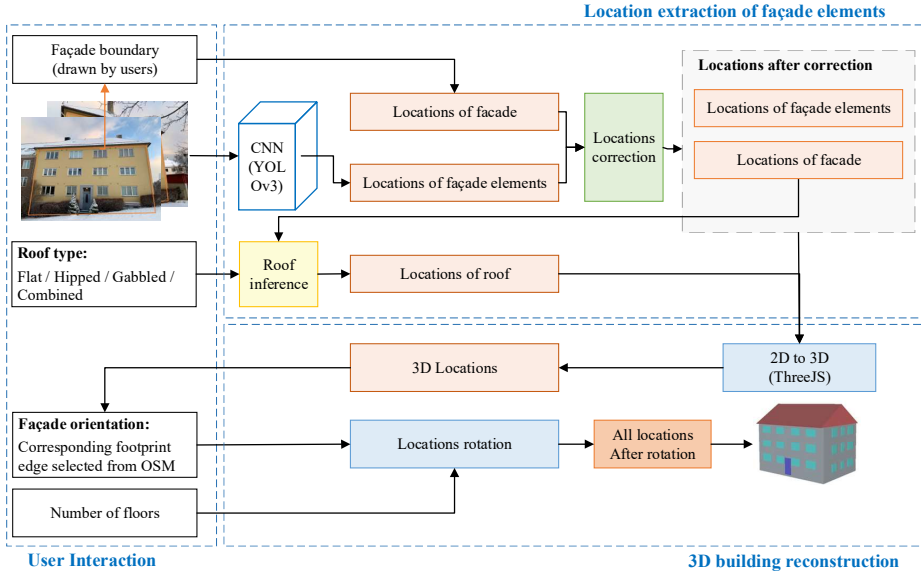


Figure 3.2. A typical workflow of the VGI3D. In the user interaction module, users provide façade boundaries, roof type, façade orientation and number of floors. Then, images are inputted into façade elements extraction module to automatically obtain the locations of the windows, doors and balconies. In the 3D building modelling module, the locations are transformed from 2D to 3D space and are then rotated according to the façade orientation. Finally, the 3D building model is reconstructed and displayed to users.

3.2.2 User interaction

In the user interaction module, users have the flexibility to upload multiple unique images of a building from different façade orientations, denoted as N_f . There are no special

requirements for image overlap or camera type. But it is worth noting that users should make sure each image capturing a complete façade structure of the building. Users then outline the corners (X_F, Y_F) of the façade on an image to define façade boundary, where $X_F = \{x_{F1}, x_{F2}, \dots, x_{Fn}\}$ and $Y_F = \{y_{F1}, y_{F2}, \dots, y_{Fn}\}$, $(X_F, Y_F) \in \{(X_F, Y_F)_{ni} \mid ni=1, \dots, N_I\}$. These corners serve as key points for reconstructing 3D façade models and rectifying the locations of façade elements. Additionally, users select the roof type and the corresponding OSM footprint edge for each façade image, which helps in generating the roof model and providing real geographic scale and coordinates. The number of storeys should also be entered to estimate the building model's height in case the real building height is not available in the database. All these information is then utilized by façade elements extraction method to prepare for the 3D building reconstruction.

3.2.3 Location extraction of façade elements

Façade elements detection

Since the VGI street-level images uploaded by users usually have a variety of complex scenes such as occlusion or bad illumination, traditional approaches are not able to deal with these scenes well. The existing approaches of interactive reconstruction need users to outline each single façade element one by one, which is labor-intensive and time-consuming. The completeness and accuracy of façade elements cannot be guaranteed when users are drawing. Therefore, it should be a good alternative solution to utilize CNNs for façade elements extraction from images because of their proven capability. Then YOLOv3 is chosen to detect façade elements due to its reliable and excellent performance. It can also make an impressive balance between detection accuracy and speed. Following the work of Kong and Fan (2020), the Darknet53 is applied as the backbone of YOLOv3. The whole network is retrained on FaçadeWHU dataset from scratch to detect façade elements (windows, doors and balconies) directly upon the uploaded images. Then, each location of façade elements, also called *bounding box*, is organized and formulated as $(C_{cm1}, C_{cm2}, categorycode) = ((x_{cm1}, y_{cm1}), (x_{cm2}, y_{cm2}), categorycode)$ for the location's correction afterwards. $(C_{cm1}, C_{cm2}, categorycode) \in \{(C_{cm1}, C_{cm2}, categorycode)\} = \{(C_{cm1}, C_{cm2}, categorycode)_{ni}^{k,i} \mid k=1, \dots, N_c; i=1, \dots, n_k; ni=1, \dots, N_I\}$, where where N_c is the number of categories, n_k is the number of façade elements of a façade in each category, and N_I is the number of input images.

Perspective distortion correction and inference of façade elements

Perspective distortion often occurs in initial VGI images by distorting the shape or layout of the façade elements, resulting in the incorrect locations of the façade and its façade elements. Hence, it is essential to correct the perspective distortion before 3D building reconstruction. The following steps are listed to resolve this issue:

- (1) The aspect ratio $r_F = h_F / w_F$ of every façade bounding box is firstly calculated, where h_F and w_F are the height and width of the façade bounding box drawn by the user. Meanwhile, the façade height and width after perspective distortion correction are defined as h_p and $w_p = h_p / r_F$, respectively. As a result, the corrected bounding box of the façade is $((0, 0), (w_p, h_p))$, and the corrected location of façade can be denoted as $(X_{pF}, Y_{pF}) = ((0, 0, w_p, w_p), (0, h_p, h_p, 0))$.
- (2) A homography matrix, M , is then computed by façade's bounding box before and after perspective distortion correction using Equations (3-1) - (3-3). This homography matrix is regarded as the perspective transformation matrix and is used to correct perspective distortion.

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = M \begin{bmatrix} x_F \\ y_F \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x_F \\ y_F \\ 1 \end{bmatrix} \quad (3-1)$$

$$x_p = \frac{x'}{w'} = \frac{m_{11}x_F + m_{12}y_F + m_{13}}{m_{31}x_F + m_{32}y_F + m_{33}} \quad (3-2)$$

$$y_p = \frac{y'}{w'} = \frac{m_{21}x_F + m_{22}y_F + m_{23}}{m_{31}x_F + m_{32}y_F + m_{33}} \quad (3-3)$$

where (x_F, y_F) is the initial façade location obtained from user's drawing, and (x_p, y_p) is the new façade location after perspective correction.

- (3) All the locations after applying perspective distortion correction can be denoted as (X_{pF}, Y_{pF}) for façade, and $\{(C_{p1}, C_{p2}, categorycode)_{ni}\}$ for the façade elements. Specifically, the location of each façade element is formulated as $(C_{p1}, C_{p2}, categorycode)_{ni} = ((x_{p1}, y_{p1}), (x_{p2}, y_{p2}), categorycode)_{ni}$. $(C_{p1}, C_{p2}, categorycode)_{ni} \in \{(C_{p1}, C_{p2}, categorycode)_{ni}\}$.

Now all the locations caused by perspective distortion have been corrected, but the layout of façade elements is still misaligned and will influence the 3D building reconstruction. Therefore, a layout correction approach for façade elements is introduced here:

- (a) First of all. for each façade, the locations of façade elements after perspective correction from xy - xy to xy - wh are reorganized and formulated as $\{(x_{pc}, y_{pc}), (wep, hep), categorycode)_{ni}\}$, where (x_{pc}, y_{pc}) is the center of each façade element, and (wep, hep) is the width and height of every façade element. The locations are then grouped by $categorycode$ and outputted as $\{(x_{pc}^k, y_{pc}^k), (wep^k, hep^k), categorycode^k)_{ni}\}$, where $k = 1, 2, \dots, N_c$, and N_c is the number of categories.
- (b) For each category of façade elements, the new xy - wh locations are sorted by y_{pc} . Then the height difference between two adjacent locations is represented as $(h_{diff}_p^k, categorycode^k)$,

$$h_{diff_p}^{k,i} = y_{pc}^{k,i+1} - y_{pc}^{k,i} \quad (3-4)$$

$$k = 1, 2, \dots, N_c, \quad i = 1, 2, \dots, (n_k - 1)$$

where N_c is the number of categories, and n_k is the number of façade elements in each category of a façade.

- (c) A clustering algorithm, *K-means++* (Arthur and Vassilvitskii., 2007), is employed to classify height difference into two clusters. One cluster G_s has small height difference and represents that façade elements are at the same storey. Large height difference is composed of the other cluster G_l , which represents the façade elements are at different storeys.
- (d) The façade elements of each category are classified into N_f groups, where N_f is the number of floors entered by the users. Every group corresponds to a floor of the building, and then computes the mean *y-center* coordinate y_{ac}^{kj} as well as mean height he_a^{kj} of the façade elements, $j = 1, 2, \dots, N_f$.
- (e) Repeat steps (b) - (d) to correct the *x-center* coordinates of the façade elements, but the height difference, *y-center* coordinates and mean height should be accordingly replaced with width difference, *x-center* coordinates and mean width. As a result, the layout of all façade elements can be eventually rectified.

The locations of façade elements after layout correction are formulated as $\{(x_{ac}, y_{ac}), (we_a, he_a), categorycode\}_{ni}$. However, this *xy-wh* locations is not suitable for 3D building reconstruction and thus they need to be changed to *xy-xy* locations. Then the location of each façade element is denoted as $(C_{l1}, C_{l2}, categorycode)_{ni} = ((x_{l1}, y_{l1}), (x_{l2}, y_{l2}), categorycode)_{ni}$. Repeating above steps (1) - (3) and (a) - (e) will be able to obtain all the locations of inputted façades as well as corresponding locations of their façade elements.

After that, the roof's location can be easily inferred according to locations of the façade, i.e., the highest façade's coordinate on the front view is usually the lowest roof's coordinate. There are four predefined roof types in VGI3D and the roof model will be automatically reconstructed based on user's selected roof type. The roof location is denoted as (X_R, Y_R) .

3.2.4 3D building reconstruction

Before the 3D modelling, it is necessary to normalize the locations obtained from subsection 3.2.2 because their values are in a various range due to different sizes of input images. Thus locations of each image ni are then normalized from range $(0, h_p)$ to range $(0, h_{th})$ in height, and from range $(0, w_p)$ to $(0, h_{th}/r_F)$ in width using Equation (3-5), where h_{th} is a constant to make sure the same height of all images. The location of each façade element after normalization is formulated as $(C_{nr1}, C_{nr2}, categorycode)_{ni} = ((x_{nr1}, y_{nr1}), (x_{nr2}, y_{nr2}), categorycode)_{ni} \in \{(C_{nr1}, C_{nr2}, categorycode)_{ni}\}$. Additionally, the center of all locations is moved to coordinate $(0, 0)$ for better visualization.

Methodology

$$x_{nr}^{k,i} = x_l^{k,i}/w_p \times h_{th}/r_F - \frac{1}{2}(h_{th}/r_F), \quad y_{nr}^{k,i} = y_l^{k,i}/h_p \times h_{th} - \frac{1}{2}h_{th} \quad (3-5)$$

$$k = 1, 2, \dots, N_c, \quad i = 1, 2, \dots, n_k$$

Now the normalized locations of façade and its façade elements are still under a 2D reference coordinate system, and they need to be transformed into 3D space for the purpose of 3D modelling. Therefore, a lightweight JavaScript 3D library, Three.js (Dirksen, 2015), is used to reconstruct 3D shapes from 2D polygons by applying a useful modelling function, *ExtrudeGeometry*. The 3D locations after coordinate transformation are formulated as $((X_{3D}, Y_{3D}, Z_{3D}), categorycode)_{ni}$, where $X_{3D} = \{X_{3D}^0, X_{3D}^1, \dots, X_{3D}^{N_{bp}}\}$, and $X_{3D}^i = (x_{3D}^{i0}, x_{3D}^{i1}, \dots, x_{3D}^{i(co-1)})$. Y_{3D}, Z_{3D} are the same as X_{3D} . N_{bp} refers to the number of all elements and can be calculated by $N_{bp} = 1 + \sum_{k=1}^{N_c} n_k$. co is the number of corners of the façade as well as each façade element.

By repeating above steps for every input image, it is possible to figure out N_I 3D location groups of a building formulated as $\{((X_{3D}, Y_{3D}, Z_{3D}), categorycode)\} = \{((X_{3D}, Y_{3D}, Z_{3D}), categorycode)_{ni} \mid ni=1, \dots, N_I\}$. If $N_I > 1$, that means users input more than one image with different façade orientations belonging to a building.

Next step is to rotate 3D locations of every façade and its façade elements to their right positions in a 3D space. The right spatial positions are the corresponding footprint edges selected by users in the user interaction module. Please note that the Three.js utilizes a right-hand rule for reference coordinate system and hence the y -axis is face-up. The rotation matrix M_r can be formulated as shown in Equation (3-6), where θ is the rotation angle between façade derived from image and corresponding footprint edge (i.e., real geographic façade position).

$$M_r = \begin{bmatrix} \cos \theta & 0 & \sin \theta \\ 0 & 1 & 0 \\ -\sin \theta & 0 & \cos \theta \end{bmatrix} \quad (3-6)$$

The rotated 3D location groups can be obtained by applying Equation (3-7) and they are denoted as $\{((X_{3Dr}, Y_{3Dr}, Z_{3Dr}), categorycode)\}$.

$$[X_{3Dr}^g \ Y_{3Dr}^g \ Z_{3Dr}^g]^T = M_r [X_{3D}^g \ Y_{3D}^g \ Z_{3D}^g]^T \quad (3-7)$$

$$g = 1, \dots, N_I$$

Finally, the 3D building model is reconstructed based on rotated 3D locations. The semantic information of all building parts is derived from category code of each element and they are visualized in different colors according to a predefined color scheme.

3.3 Roof plane segmentation

Though the main goal of Section 3.2 is to reconstruct detailed façade structures, the roof model is also automatically inferred and generated by selecting a predefined roof type, for the purpose of integrity of the reconstruction results. However, the geometries of 3D roof models are usually inaccurate by using this way. Hence, to reconstruct the 3D roof models as accurately as possible, it is a promising solution to firstly segment roof planes from ALS point clouds and then reconstruct the segmented roof planes into 3D polygons by detecting the corner/key points. In this way, the final 3D roof model would not only have accurate geometric structure but also have semantic information for every single roof plane. This section will elaborate a deep learning-based method of roof plane segmentation in details.

The proposed method can be regarded as an improvement of the ASIS network (Wang et al., 2019b) with an incorporated feature fusion (FF) module and a modified joint features learning (JFL) module. With its novel network architecture and efficient learning strategy of mutual features, the ASIS network has delivered remarkable outcomes for both instance and semantic segmentation tasks on the S3DIS (Armeni et al., 2016) indoor point cloud dataset. Nevertheless, the ASIS network neglected the fused features generated from the decoder of backbone network (PointNet++), which have the potential to create more distinctive features and improve prediction accuracy. To resolve the issues highlighted earlier in ASIS, an FF module has been added into the proposed method, and certain modifications have been made to the JFL module to enhance the overall segmentation results.

3.3.1 Network architecture

Three main components of the network are depicted in Figure 3.3, i.e., the PointNet++ serving as the backbone network, the FF module, and the JFL module. The backbone network is composed of a shared encoder and two parallel decoders, where one decoder branch is responsible for extracting point-level semantic features for semantic predictions (i.e., groups of roof planes with similar geometric shapes) and the other decoder branch performs instance segmentation (i.e., separated roof planes). The FF module is integrated immediately after the semantic decoder to fuse high- and low-level features extracted from the semantic decoder. The JFL module promotes the learning of both semantic and instance features.

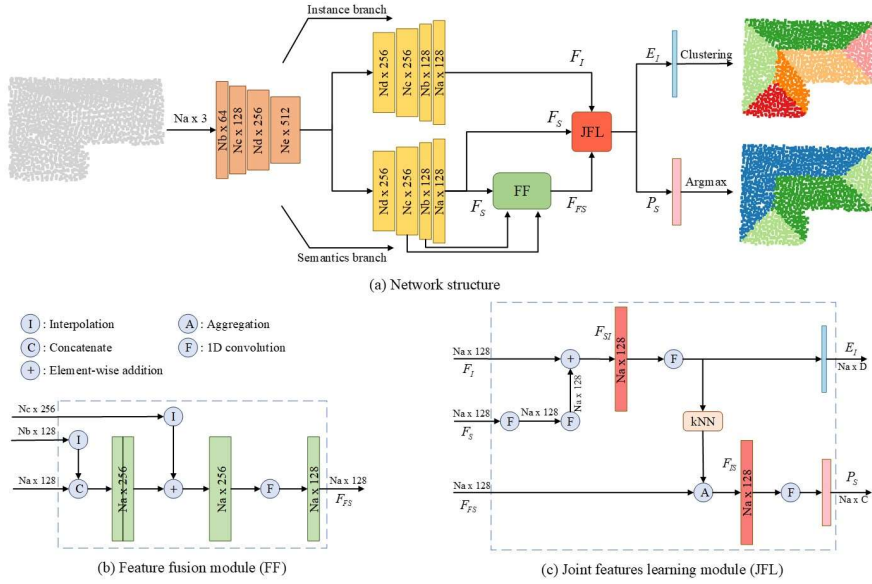


Figure 3.3. (a) The architecture of our RoofNet with two proposed modules: (b) the feature fusion (FF) module; (c) the joint features learning (JFL) module.

The proposed approach starts with inputting a point cloud of size N_a , which is then encoded by the shared encoder to produce a $N_e \times 512$ feature matrix. This feature matrix is then decoded separately by two parallel decoders into an $N_a \times 128$ shaped feature matrix. The output of the semantic decoder is represented by F_S , while the output of the instance decoder is represented by F_I . Subsequently, the semantics decoder generates two parallel branches with the same F_S as input. One of the parallel branches fuses the F_S with the features from different layers of the semantic decoder via an FF module, thereby producing a new fused semantic feature matrix F_{FS} of shape $N_a \times 128$. Finally, JFL module receives three types of features, namely F_I , F_S and F_{FS} , and produces two matrices (E_I and P_S) simultaneously. The matrix E_I with shape $N_a \times D$ denotes the point-level instance embeddings, where D represents the dimension of the embeddings. The clustering algorithm later utilizes E_I to predict the instance label for each point. Given that points belonging to the same instance are closely situated in embedding space, while those belonging to different instances are far apart (Wang et al., 2019b). Hence, embeddings from the network are output instead of directly outputting instance predictions. The other matrix P_S with shape $N_a \times C$ represents the final semantic predictions, where the most likely class is outputted from C types of candidate classes.

During the training phase, RoofNet is supervised by a hybrid loss function ℓ_{all} comprising a standard softmax cross entropy loss ℓ_{sem} for learning per point semantics and a

discriminative loss function ℓ_{ins_emb} consisting of three terms to learn instance embeddings, which is inspired by instance segmentation in 2D images (De Brabandere et al., 2017). The hybrid loss function ℓ_{all} is defined as follows:

$$\begin{aligned}\ell_{all} &= \ell_{sem} + \ell_{ins_emb} \\ &= \ell_{sem} + (\alpha \cdot \ell_{close} + \beta \cdot \ell_{apart} + \gamma \cdot \ell_{reg})\end{aligned}\quad (3-8)$$

$$\ell_{close} = \frac{1}{K} \sum_{k=1}^K \frac{1}{N_k} \sum_t^{N_k} [\|\mu_k - e_t\|_1 - \delta_v]_+^2 \quad (3-9)$$

$$\ell_{apart} = \frac{1}{K(K-1)} \sum_{k_A=1}^K \sum_{k_B=1}^K [2\delta_d - \|\mu_{k_A} - \mu_{k_B}\|_1]_+^2, \quad k_A \neq k_B \quad (3-10)$$

$$\ell_{reg} = \frac{1}{K} \sum_{k=1}^K \|\mu_k\|_1 \quad (3-11)$$

where ℓ_{close} applies a pulling force that aims to draw embeddings towards the center of the instance (i.e., mean embedding of the instance), while ℓ_{apart} applies a pushing force that separates different instances from each other. The regularisation term ℓ_{reg} pulls all instances towards the origin to form a boundary for embedding values. In addition, K is defined as the number of instances; N_k stands for the number of points in k -th instance; μ_k refers to the center of k -th instance (the mean embedding); the embedding of a point is represented by e_t ; $\|\cdot\|_1$ is used to calculate L_1 distance; δ_v and δ_d are the margins for the loss ℓ_{close} and ℓ_{apart} respectively; $[x]_+ = \max(0, x)$. The experiments follow the guidance of De Brabandere et al. (2017), where set $\alpha = \beta = 1$, $\gamma = 0.001$ are set.

Last but not the least, during the testing phase, a mean-shift clustering algorithm (Comaniciu and Meer, 2002) with *bandwith* = 0.6 is used to generate instance labels on embeddings E_I produced from the instance segmentation branch. For semantic labels, an argmax operation is performed on semantic predictions P_S to obtain the semantic labels.

3.3.2 Feature fusion (FF) module

It is widely accepted that the low-level layers of a network tend to learn more local and detailed features such as edges and curves, while the higher-level layers tend to learn more global and semantic features such as object shapes. This insight has been obtained through the visualization of each layer's output (Qin et al., 2018). Many researchers have attempted to fuse features from different layers to improve the accuracy of semantic segmentation. Experimental results have shown that such fused features have a positive impact on learning and result in better segmentation results, both for 2D images (Chen et al., 2018) and 3D point clouds (Hu et al., 2020).

Drawing from previous research successes, the feature fusion is also adopted in this thesis, and an FF module is introduced (as shown in Figure 3.3b) to enhance the network's predictions. Due to computational efficiency and GPU memory consumption, only the last three layers of the decoder are fused. These three features are denoted as F_c , F_b and F_a , with corresponding shapes of $N_c \times 256$, $N_b \times 128$ and $N_a \times 128$, respectively. In the FF module, an upsampling operation, involving interpolation, is first conducted on F_b and F_c to achieve the same point number for all features. This results in F_b and F_c being upsampled and denoted as F'_b and F'_c , respectively. Next, F_a and F'_b are concatenated, producing $F_{ab'}$. $F_{ab'}$ is then added to F'_c using element-wise addition, generating the feature $F_{ab'c'}$ with a shape of $N_a \times 256$. Lastly, $F_{ab'c'}$ is passed through a 1D convolution (Conv1D) with batch normalization and ReLU non-linearity to produce the final fused feature F_{FS} with a shape of $N_a \times 128$. The upsampling operation is carried out using the inverse distance weighted average based on the three nearest neighbors, following the approach of Qi et al. (2017).

3.3.3 Joint features learning (JFL) module

Instance segmentation is generally an extension of semantic segmentation, since the latter has already grouped points with the same semantic label together, while the former aims to separate each distinguishable instance from the groups. Therefore, the quality of the semantic features has a significant impact on instance segmentation results. While the ASIS network has shown that joint learning of the two tasks can lead to a win-win situation (Wang et al., 2019b), it remains unclear whether fused features can better facilitate the learning of semantic features and, in turn, enhance the learning of instance embeddings. To address this issue, this thesis proposes modifications to the ASIS network by adding an FF module and ultimately forming a JFL module (Figure 3.3c).

To incorporate the semantic features into instance features, the initial semantic feature matrix F_S is transformed into an instance feature space using two sequential Conv1Ds with batch normalization and ReLU non-linearity, resulting in F'_S . Then, F'_S is added to the instance feature F_I element-wisely to obtain the semantic-aware instance features F_{SI} . Finally, the instance embeddings E_I , which have a shape of $N_a \times D$, are produced from the semantic-aware instance features F_{SI} through a Conv1D operation followed by dropout (*Drop*). The complete integration procedure can be outlined as follows:

$$F_{SI} = \text{Conv1D}(\text{Conv1D}(F'_S)) + F_I \quad (3-12)$$

$$E_I = \text{Conv1D}(\text{Drop}(F_{SI})) \quad (3-13)$$

It is expected that the instance embeddings can enhance the learning of semantic features and improve the distinction of boundaries between two groups of semantic classes, since in the instance embedding space, points of the same instance are brought closer together while different instances are pushed apart. Hence, following the ASIS, the k nearest neighbors (kNN) of each point in the instance embedding space are found, which forms an index matrix with a

shape of $N_a \times M$. Then the fused semantic matrix F_{FS} (*getGroupFeatures*) is used to generate groups of fused semantic features denoted as G_{FS} with a shape of $N_a \times M \times 128$. Each group represents a local region in the instance embedding space that is close to its centroid point. After that, the fused semantic features of each group G_{FS} and the original semantic matrix F_{FS} are aggregated through a max aggregation operation (*Aggregation*) to produce the instance-aware semantic feature matrix F_{IS} . Finally, the instance-aware semantic features F_{IS} are fed into a Conv1D layer with dropout to generate the final semantic predictions P_S with a shape of $N_a \times C$. The entire process can be described as follows:

$$G_{FS} = \text{getGroupFeatures}(kNN(E_I), F_{FS}) \quad (3-14)$$

$$F_{IS} = \text{Aggregation}(G_{FS}, F_{FS}) \quad (3-15)$$

$$P_S = \text{Conv1D}(\text{Drop}(F_{IS})) \quad (3-16)$$

3.3.4 Roof plane ALS point cloud dataset

As mentioned earlier, currently available public ALS point cloud datasets have several limitations such as the lack of roof plane annotations, insufficient coverage of roof types, or point densities that deviate significantly from standard point clouds. To address these research gaps, a new point cloud dataset called RoofNTNU is manually established based on raw ALS point cloud data with a standard point density of 12-20 points/m², captured by the mapping authority of Trondheim Municipality. The typical roof structures are then selected from residential areas based on the roof types concluded by Kada (2007), with certain modifications as some roof types are not suitable for Norway. Additionally, complex roof structures that are a combination of two or three “sub-roof structures” are defined as a new roof type, combination, in this thesis. Overall, seven typical roof structures, labeled from 1 to 7 (*flat*, *hipped*, *gabled*, *corner element*, *T-element*, *cross element*, and *combination*) are identified and presented in dataset RoofNTNU, as shown in Figure 3.4.

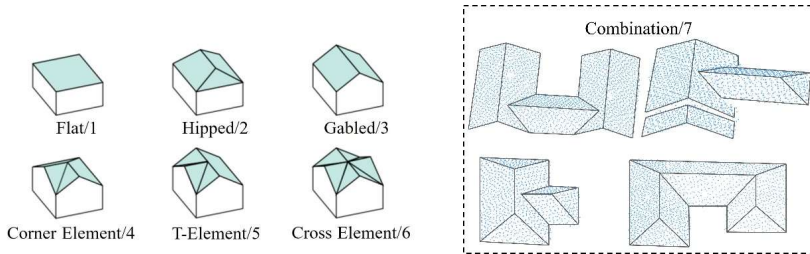


Figure 3.4. Visualization of defined roof types in the dataset RoofNTNU. The left figure is the abstract representation of roof types 1-6, while the right figure shows some practical examples of roof type combination/7.

The recognition of the geometric shapes that comprise the roof structures is crucial for instance labeling of roof planes as it teaches the neural network to learn the fundamental geometric shapes from point cloud data. These geometric shapes have been summarized into five categories: rectangle, isosceles trapezoid, triangle, parallelogram and ladder-shaped. Each category has been given two or four labels based on the proposed roof types in Figure 3.4. The label digits range from 0 to 11 and have been assigned a unique and distinguishable color for visualization purpose, as shown in Figure 3.5. Moreover, semantic labels for semantic features learning have also been defined to benefit the learning of instance embeddings. Label digits ranging from 0 to 4 have been assigned to represent groups of roof planes with the same geometric shapes. For example, hipped roof can be decomposed into two groups: a pair of isosceles trapezoids and a pair of triangles. A specific color map has also been assigned to semantic labels. Please note that even though some colors are the same between semantic and instance labels, they belong to two different tasks, and the colors of semantic labels should not be confused with the colors of instance labels. The structure of the ground truth has been denoted as $[x, y, z, \textit{roof-type}, \textit{instance-label}, \textit{semantic-label}]$, where the coordinate $[x, y, z]$ has been normalized, and the correspondence between two sorts of labels with plane geometries and roof types can be found in Table 3.1.

Table 3.1. Overview of correspondence among semantic and instance labels, plane geometries and roof types in the ground truth. Please note that do not confuse semantic label color with instance label color, as they belong to two different tasks.

<i>Semantic labels</i>	<i>Instance labels</i>	<i>Plane geometries</i>	<i>Roof types</i>
0	0	Rectangle	Flat, Gabled, T-Element, Cross Element, Combination
	1		
1	2	Isosceles trapezoid	Hipped, Corner Element, Combination
	3		
2	4	Triangle	Hipped, Corner Element, Combination
	5		
3	6	Parallelogram	Corner Element, Combination
	7		
4	8	Ladder-shaped	T-Element, Cross Element, Combination
	9		
	10		
	11		

As a result, the dataset RoofNTNU is established by manually annotating and segmenting a total of 930 different roofs with 3498 planes, comprising over 2.2 million points. The dataset includes seven typical roof structures found in Western Europe and has a standard point density. Thus it can provide general usability for roof plane segmentation from most ALS point cloud data. The distribution of the roof types in RoofNTNU is shown in Figure 3.5. For the purpose of neural network training, each roof type is randomly divided into three subsets

with an 8:1:1 ratio, namely training dataset (744), validation dataset (93), and testing dataset (93).

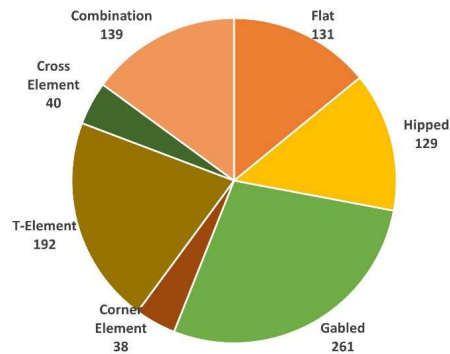


Figure 3.5. The distribution of the roof types in RoofNTNU dataset.

3.4 Web-based 3D visualization

Although the generated LoD3 3D building models can be visualized in a local 3D viewer of VGI3D, they are unable to connect to a virtual 3D city environment and perform interactive visualization over there, resulting in a lack of immediate visual satisfaction. If integrating the 3D buildings into a virtual 3D city environment, it would not only address the shortcomings mentioned above but also strengthen the sense of accomplishment for users, since they would realize that every tiny contribution is of great help in improving and enriching the whole massive VGI project. Moreover, such a vivid 3D geographic scene is likely to increase users' motivations to participate in the project and contribute more 3D building data. Therefore, a web-based 3D visualization with WebGL is a great solution to meet above requirements when taking convenience and accessibility into consideration. A similar and existing implementation is Google Earth (Gorelick et al., 2017), but it cannot accept CityGML as input format to integrate other new 3D data. For this reason, a new 3D visualization platform is developed based on Cesium, which is a JavaScript library based on WebGL with interfaces for various data sources for base maps, vector layers, etc (Schilling et al., 2016). 3D Tiles technology is also utilized for rapid rendering of massive 3D data over the web.

The entire 3D visualization platform consists of the following main components:

Base map

A layer with geographic information that serves as a background and provides context for additional layers that are overlaid on top of the base map.

3D terrain

3D terrain data is tiled as a set of high-resolution tiles in modern Quantized Mesh format that are derived from georeferenced DTM data with one-meter resolution. The terrain tileset covers the whole Norway and has 18 zoom levels for users to observe the terrain by zoom in/out.

3D building models

There are three sorts of data sources comprising the 3D building models with different LoDs. The first one is LoD2 building models but without semantic information generated from ALS point clouds and OSM footprints using the method proposed by Huang et al. (2022). Second, LoD3 building models with semantics generated by volunteers via VGI3D. The third one is a group of crowdsourcing SketchUp building models which cover all campus of NTNU in Trondheim. Yet, the data format conversion (i.e., .skp to CityGML) and georeferencing of them are necessary, because the format .skp is specially designed for SketchUp under a local coordinate system and hence, it cannot be directly visualized on web-based applications. Thus, the tasks of format conversion and georeferencing are implemented with the help of OSM footprints and an open-sourced SketchUp plugin, GEORES (2023).

3D road networks

3D road networks are a series of coherent road segments represented by 3D polygons and are generated from initial 2D polylines with no elevations. To visualize them on high-resolution 3D terrain, it is essential to interpolate the additional vertices of road segments according to the ups and downs of the actual terrain, as shown in Figure 3.6, where the red points are the desired additional vertices. This thesis applies collision detection algorithm to find these additional vertices out by creating an axis aligned bounding box (AABB) tree (Hart and Rimoli, 2020), and then employs Catmull-Rom spline algorithm (Twigg, 2003) to expand the 3D road polylines to 3D polygons, which can make curves keep as smooth as possible.

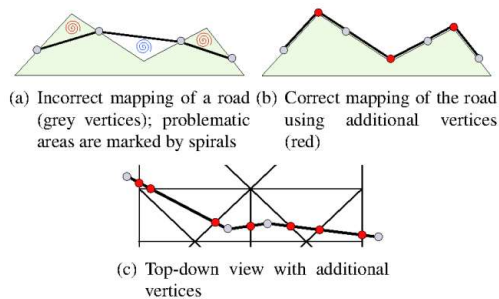


Figure 3.6. The sketch of 2D road polylines mapping onto terrain meshes (borrowed from the work of Vaaraniemi et al. (2011)).

3D road-related objects

To improve the content richness of the 3D visualization platform, this thesis studies the automated detection and localization of road objects at road intersections from street-level image sequences. The whole workflow is depicted in Figure 3.7 and consists of three modules: (1) data preprocessing and cleaning module; (2) object segmentation and recognition module; (3) localization module.

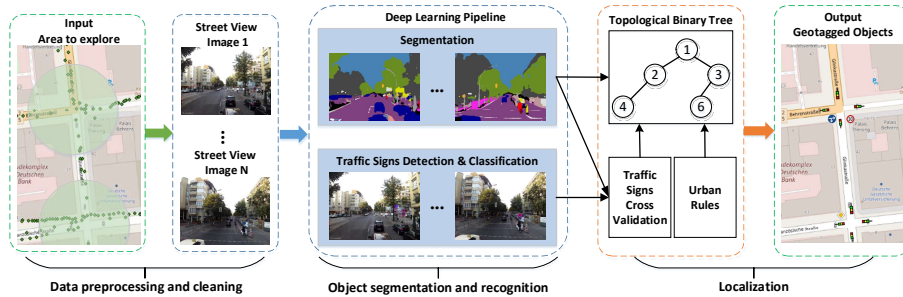


Figure 3.7. The workflow of the automated detection and localization of road objects from street-level images.

The workflow starts with the data preprocessing and cleaning module for (1) image sequences identification within intersection buffers and (2) correction of GPS positions of image sequences. The latter issue may be caused by the tall buildings obstructing the GPS signal or the low accuracy of GPS receiver built into the camera, and can be addressed by structure from motion (SfM) algorithm (Snavely et al., 2008).

In the second module, road-related objects such as sidewalks, traffic signs and lights are extracted from images using a CNN-based method, PSPNet (Zhao et al., 2017) trained on Mapillary Vistas dataset (Neuhold et al., 2017). Nevertheless, the quality of VGI images varies greatly, making it challenging to ensure accurate segmentation for all images. Thus an approach (YOLOv3) based on object detection is also adopted to address this issue together.

In the third module, since VGI images usually lack camera intrinsics in their EXIF information, it is not possible to apply photogrammetry methods for the localization purpose. Therefore, an alternative solution is figured out by creating an attributed topological binary tree (ATBT) according to urban rules. The ATBT can depict the coherent relations of topologies and attributes of the extracted road objects. Summarized urban rules are listed in Table 3.2. With the image's shooting location approaching to the intersection, ATBT is able to update itself until it no longer changes. Furthermore, the scene depth information is also taken into account when creating and updating the ATBT. For instance, in Figure 3.8 two low traffic lights (labeled by No. 9 and 19 in the third image) are identified as a pair according to *Rule 4*. Their height should be the same in reality, however, the No. 19's height appears to be

Table 3.2. Summarized urban rules used in attributed topological binary trees (ATBT).

Rule No.	Description of the urban rules
<i>Rule1</i>	1) Traffic light is surrounded by sky; 2) distance between the traffic light and road surface is far more than twice the height of the tallest pedestrian. Conclusion: high traffic lights
<i>Rule2</i>	1) Traffic light is surrounded by buildings; 2) distance between the traffic light and road surface is less than or equal to twice the height of the tallest pedestrian. Conclusion: low traffic lights
<i>Rule3</i>	The sidewalks on the same side are connected.
<i>Rule4</i>	Low traffic lights on the sidewalks tend to appear in pairs.
<i>Rule5</i>	Traffic signs either appear alone, or are usually close to the low traffic light above or both up and down, or arrange together.
<i>Rule6</i>	The minimum width of an ordinary sidewalk is 2~3 meters.

lower than No. 9. Because photography follows the law that the object appears larger when it is closer and smaller when it is farther away. To overcome this challenge, a combination of scene depth information and *Rule 4* is used to infer that these two traffic lights are indeed a pair and should be located on the sidewalk, leading to a better performance of the ATBT.

Finally, the approximate positions of road objects can be determined by matching ATBT with OSM footprints as well as the usage of urban rules. The proposed method is carried out on two object categories, traffic lights and signs, as a specific case study. Furthermore, a comprehensive explanation about how this method works can be found in Paper 3.

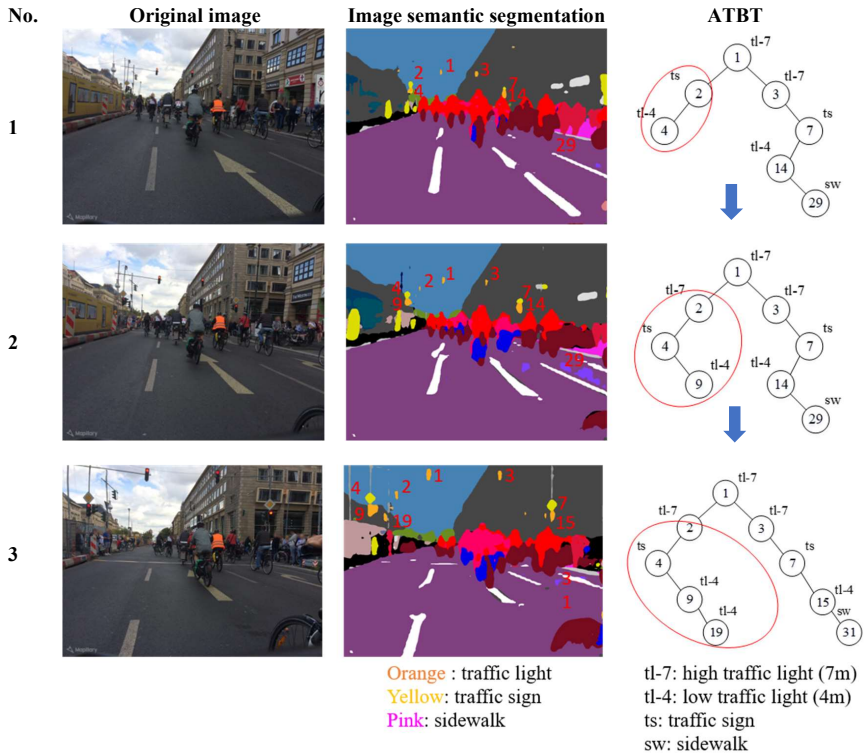


Figure 3.8. ATBT generation and its self-updating considering the scene depth information. The image numbered 1-3 are gradually approaching to the intersection.

Methodology

Chapter 4

Implementation, results and discussion

The existing related studies mentioned in Section 2.1 have proven the possibility and have shown great potential for reconstructing 3D building models with semantics using crowdsourcing approaches, even though this topic is still a quite young and innovative research field with many open questions. Differing from the 2D vectors (e.g., OSM) and images (e.g., Mapillary or Flickr), a new type of VGI data with an additional third dimension is very likely to draw much attention of users from both academia and industry. Besides, conventional 3D geospatial data production methods can hardly meet the needs of emerging and rapidly developing digital twin cities. They require to add additional 3D VGI data (e.g., detailed 3D building models with semantic information) as one of significant data sources to construct multi-source and multi-angle representations of geographic entities. For this purpose, a new 3D VGI data collection platform should be designed from the perspective of contributors. The platform should be developed in a web-based environment to ensure user convenience and interoperability during operation and interaction. Furthermore, it is also crucial to connect the reconstructed 3D buildings with a virtual 3D geographic environment to enable interactive visualization. This connection aims to provide users with immediate visual satisfaction and a sense of psychological fulfillment upon making contributions, thereby fostering motivations for further data contributions. Hence, all these measures can answer the proposed research question Q1.

The rest of this chapter will introduce the detailed framework implementation of Q1 and present the key results published in 4 journal articles (Paper 1, 2, 3, and 4). Research questions Q2 and Q3 and their subquestions will be also addressed in respective sections.

Section 4.1 shows the results related to the proposed interactive 3D building reconstruction system, VGI3D, and answers research questions Q2 to Q2.3. Specifically, Paper 1 and Paper 2 describe the methodology implementation for reconstructing 3D building models in a crowdsourcing manner, as presented in Section 4.2.1. Section 4.2.2 displays the results of Paper 4 concentrating on visual and quantitative results for both roof instance and semantic segmentation tasks on RoofNTNU testing dataset, for the purpose of roof model reconstruction. Furthermore, an architecture study of the proposed network is also conducted.

Section 4.2 covers the results about the web-based 3D visualization platform including its basic framework and overall visualization result integrated with all mentioned components. Research questions Q3 to Q3.3 are answered, among which Paper 3 addresses the Q3.3.

4.1 3D building reconstruction using crowdsourcing method

The crowdsourcing method for reconstructing LoD3 building models with semantic information takes three aspects into consideration. They are the simple interoperability, low cost, generality and robustness. Simple interoperability is also called ease of use, aiming to make the whole modelling process as simple as possible. Low cost refers to reduce the costs of labor/workload, processing time and equipment investment during the generation of 3D buildings. Generality and robustness enable the method to perform well on most of cases as much as possible, which can better promote the method to the mass market and attract ordinary users to use. Besides, simple interoperability is a necessary condition for achieving the latter two aspects. Therefore, the proposed method complies with this logic, and it is possible to realize the ambition of becoming a VGI platform to collect 3D building models with semantics.

Q 2: How to enable the reconstruction of 3D building models with semantics by means of crowdsourcing?

A web-based interactive system, VGI3D, for 3D building reconstruction from VGI street-level images based on deep learning technique is proposed, which is mainly utilized for generating detailed façade models including doors, windows and balconies upon them. VGI3D adopts the widely used software design pattern known as MVC (model-view-controller). An overview of the VGI3D architecture applying MVC is introduced in Section 4.2.1 from a software engineering perspective.

As mentioned in Section 3.3, since roof structures are usually invisible in street-level images, thus roof models are inferred by VGI3D, thereby resulting in the inaccurate roof geometries. To restructure high quality roof models, roof plane segmentation is a critical step in the process of 3D roof reconstruction. Then outlines of segmented roof planes can be extracted more easily, enabling the rooftop reconstruction. Section 4.2.2 illustrates the visual and quantitative result of roof plane segmentation task performing on RoofNTNU testing dataset.

4.1.1 VGI3D

VGI3D is implemented by a popular software design pattern, MVC (model-view-controller). In general, the controller interprets mouse and keyboard inputs from the user interaction, sends a request to the database to retrieve data, processes the data, and updates the view accordingly. An abstract overview of the system architecture is shown in Figure 4.1.

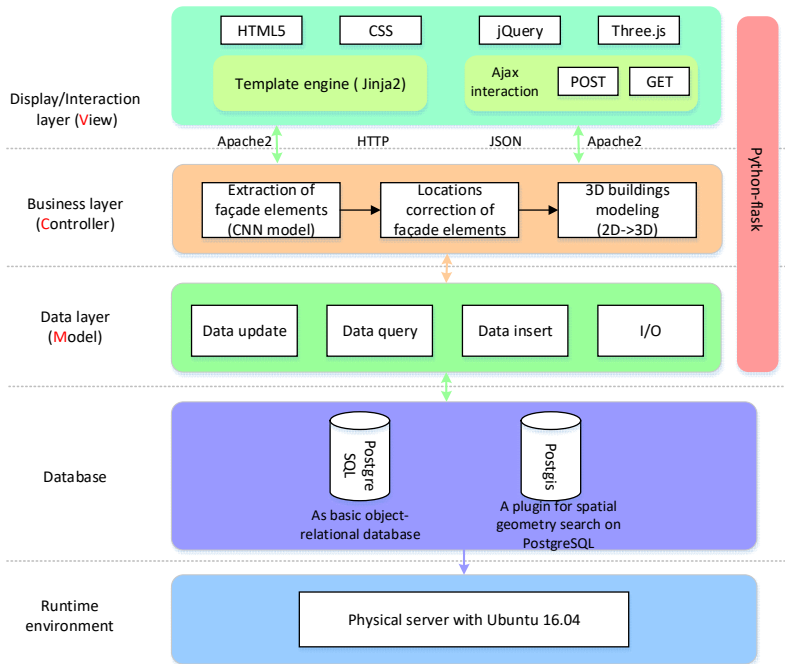


Figure 4.1. Abstract overview of the VGI3D architecture.

In this design pattern, the “M” stands for “model” and corresponds to the data layer, where handles all interactions with the database. A PostgreSQL relational database with PostGIS plugin is thus established for spatial geometry search. The data layer also stores all building footprints and partial buildings’ height in Norway, along with the images uploaded by users and the generated 3D building models. Footprints are used in interaction layer for edge selection and supplying geographic information of buildings. With this geographic information, the real ratio of building facades can be calculated, ensuring that generated 3D building models have the accurate geographic coordinates and the same ground boundaries as the real ones,

The “V” refers to the view or interaction layer, through which users can operate the system via a graphical user interface. Since one of the design principles of the VGI3D is to prioritize simplicity and user-friendliness, so a drawing idea is borrowed from Mapbox (2023) SDK (software development kit) to simplify drawing the outline of the façade boundary. Leaflet (2023) is also utilized to select the edges of the footprints from an embedded OpenstreetMap and transmitted them to the backend for calculation through GeoJSON. Furthermore, the view layer incorporates other tools for user interface rendering and interaction. HTML5, cascading style sheets (CSS), and Bootstrap are employed for

rendering user graphics, while Three.js provides 3D-rendering support. User interaction is handled through the use of jQuery, which listens to response events. To enable communication between the frontend and backend, an HTTP connection, Ajax, and Jinja2 template engine are intermittently utilized. The data flow between the frontend and backend is transmitted in the JSON format over the Internet.

The “C” represents the “controller”, which refers to the business layer and is the backbone of the system. Most key modules are placed there to handle the user's actions and perform the 3D building reconstruction. As another design principle of the VGI3D is the generality and robustness, so the modelling method should not be constrained to a limited number of façade styles and input image quality. To achieve this, a three-step pipeline is presented. First, it employs YOLOv3 to automatically detect façade elements such as windows, doors, and balconies. Utilizing a CNN model not only fulfills the goal of generality and robustness, but also significantly reduces the time required for the entire workflow, since extracting façade elements is the most time-consuming step reported by Nishida et al. (2016). Second, it then corrects the locations of façade, detected façade elements, and roof to ensure they are under the same coordinate system. Third, it converts 2D elements to 3D and obtains the final 3D building model.

Furthermore, the VGI3D is currently deployed on a physical Dell workstation with an Ubuntu 16.04 as operating system, and can be accessed at <https://vgi.ibm.ntnu.no:5002/facade/interact/>.

Q 2.1: How to ensure that VGI3D has simple interoperability for both expert and non-expert users?

The interoperability of the VGI3D for both expert and non-expert users is evaluated by conducting a small-scale usability testing at the NTNU and Wuhan University, China, with a total of 30 participants aged between 25-43 years, including 7 women and 23 men. Among them, 6 are experts with 3D modelling experience in fields such as 3D city modelling and photogrammetry, while the remaining participants are non-experts in fields such as computer science, urban planning, 3D visualization, etc. Before the testing, all participants are provided with a tutorial video and a slide presentation to familiarize themselves with the system. Each participant is randomly assigned a folder containing building images to use during the testing. After completing the testing, the participants are asked to fill out a user feedback form which includes questions about, for example, their demographic information, experience with 3D modelling, satisfaction with the interaction operation, and any suggestions or comments they have.

After the usability testing, 93.3% (28/30) of the participants report that the user interface is easy and clear to understand, which is really encouraging and positive feedback. Regarding

the 3D modelling, out of the 30 participants, 20 (66.7%) are able to reconstruct 3D models successfully, with the majority being satisfied with the speed and plausibility of the results. Some participants even express that VGI3D was highly beneficial to their own research. However, there are also negative feedback, including issues with the tips being difficult to notice and a bug that made it hard to delete façade elements. Moreover, some participants, particularly those experienced in BIM, find that the reconstructed models are less photorealistic than they were used to.

In addition to the qualitative feedback, quantitative evaluation and analysis of the questionnaire data are also performed from a statistical perspective. The statistical results about raw data have been summarized and presented in Table 4.1 (raw data can be found in Appendix B of Paper 1). Outliers are identified as single points in box plots as shown in Figure 4.2.

Table 4.1. Statistics about raw data collected from participants. Mean(μ), standard deviation(σ), lower quartile(Q1), middle quartile(Q2), upper quartile(Q3), interquartile range (IQR). 3DBMs: 3D Building Models; 3DBMing: 3D Building Modelling; 3DMing: 3D Modelling.

	Involved in 3DBMs apps?	Experienced using browser?	Experienced with 3DMing?	VGI3D easy to use?	Modeling result looks plausible?	Modeling result is completed?	VGI3D looks useful for 3BMing?	3D viewing smooth?	Fast modeling speed?
μ	3.20	4.20	2.40	4.00	3.83	3.57	3.90	4.53	4.20
σ	1.30	0.60	1.33	0.52	0.58	0.56	0.60	0.50	0.40
Q1	2.00	4.00	1.00	4.00	4.00	3.00	4.00	4.00	4.00
Q2	3.00	4.00	2.00	4.00	4.00	4.00	4.00	5.00	4.00
Q3	4.00	5.00	3.00	4.00	4.00	4.00	4.00	5.00	4.00
IQR	2.0	1.0	2.0	0	0	1.0	0	1.0	0
Skewness	-0.19	-0.11	0.51	-1.45	-0.98	-0.84	-0.90	-0.13	1.50

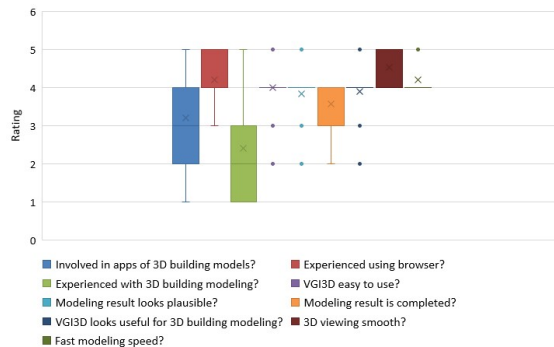


Figure 4.2. Box plot of data for illustrating skewness.

Correlation analysis

To study the potential relations between participants’ responses (i.e., the answers to questions defined in questionnaire), some possible pairs are illustrated in the form of scatter plot. Since the sample size of participants is relatively small, the scatter plots sometimes appear sparse and nonlinear, but still contain bound data internally (as seen in Figure 4.3). Hence, to accurately identify associations between participant responses, Spearman's rank correlation coefficient with full covariance has been computed between all possible question pairs (see Table 4.2), which is suitable for dealing with linear, nonlinear, and skewed relationships.

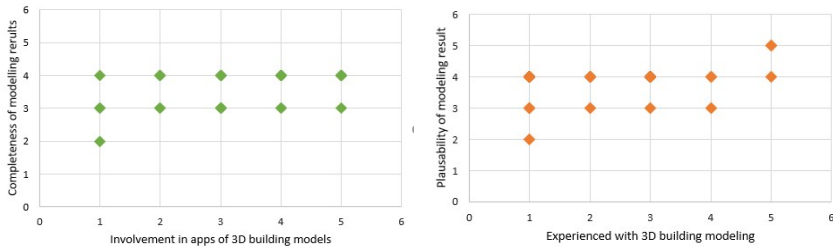


Figure 4.3. Examples of sparse scatter plots with nonlinearity in appearance but including bound data points internally.

Table 4.2. Correlation table showing Spearman’s rho and significance values for parameters. Values in red present strong positive correlation; values in blue present moderate positive correlation; other values show weak correlation. 3DBMs: 3D Building Models; 3DBMing: 3D Building Modelling; 3DMing: 3D Modelling.

	Involved in 3BMs apps?	Experienced using browser?	Experienced with 3DMing?	VGI3D easy to use?	Modeling result looks plausible?	Modeling result is completed?	VGI3D looks useful for 3BMing?	3D viewing smooth?	Fast modeling speed?
Involved in 3BMs apps?	1								
Experienced using browser?	-0.22 (p=0.25)	1							
Experienced with 3DMing?	0.76 (p=0.00)	-0.10 (p=0.60)	1						
VGI3D easy to use?	0.54 (p=0.00)	-0.17 (p=0.36)	0.50 (p=0.00)	1					
Modeling result looks plausible?	0.51 (p=0.00)	-0.28 (p=0.13)	0.33 (p=0.08)	0.68 (p=0.00)	1				
Modeling result is completed?	0.36 (p=0.05)	-0.01 (p=0.95)	0.14 (p=0.45)	0.45 (p=0.01)	0.64 (p=0.00)	1			
VGI3D looks useful for 3BMing?	0.47 (p=0.00)	-0.08 (p=0.66)	0.50 (p=0.00)	0.82 (p=0.00)	0.63 (p=0.00)	0.34 (p=0.06)	1		
3D viewing smooth?	0.50 (p=0.00)	0.05 (p=0.78)	0.31 (p=0.09)	0.40 (p=0.03)	0.56 (p=0.00)	0.60 (p=0.00)	0.27 (p=0.14)	1	
Fast modeling speed?	0.44 (p=0.01)	0.10 (p=0.60)	0.72 (p=0.00)	0.37 (p=0.04)	0.28 (p=0.13)	0.08 (p=0.68)	0.38 (p=0.04)	0.30 (p=0.11)	1

First of all, I investigate whether participants' involvement in 3D building model apps or experience with browsers or 3D building modelling is associated with their opinions on VGI3D's ease of use/understanding, plausible and completed modelling results, and usefulness for the 3D building modelling community. The findings, as presented in Table 4.2, indicate weak positive correlations between involvement in 3D building model apps and the completeness of modelling results, but with nearly 100% confidence, and moderate positive correlations with ease of use, modelling results plausibility, and VGI3D usefulness. Although the correlation is weak, it is still supported by participants as shown in Figure 4.3(left). However, given the small sample size, it needs more data to confirm this trend. Surprisingly, experience with 3D building modelling shows weak positive correlations with the plausibility and completeness of modelling results and smooth 3D viewing capability, which is contrary to my expectations. Upon closer examination of the raw data, it can be found that non-expert participants, who have little 3D modelling experience, reduce the overall correlations among responses. However, the relationship between experience with 3D building modelling and the plausibility of modelling results in Figure 4.3(right) displays a different trend from the correlation. Higher ratings reveal that participants indeed think the modelling results look plausible, regardless of their level of experience in 3D modelling.

Second, I seek to determine whether the usefulness of VGI3D for the 3D building modelling community is linked to any aspect of VGI3D, such as its ease of use/understanding, the plausibility or completeness of modelling results. Based on Table 4.2, strong correlations with almost 100% confidence are found between usefulness for the 3D building modelling community and both the ease of use/understanding of VGI3D and the plausibility of modelling results. However, it cannot be observed a strong correlation between the completeness of modelling results and usefulness.

The final observation is regarding the relationship between the ease of use/understanding of VGI3D and the plausibility of modelling results. A clear positive correlation can be discovered between these two variables, with a value of 0.68, which is in line with my expectations. It is worth noting that all of the findings and correlations are derived from a restricted sample of data. Although I can only make assumptions about the meaning behind them, they still provide with valuable insights into which aspects of the system are critical in shaping users' perceptions. This information will be useful in the future efforts to optimize the system.

Sensitivity analysis

To verify the low sensitivity of the VGI3D, five buildings with different façade complexities are selected, ranging from simple to complex (as shown in Table 4.3). Building A has the simplest façade structure with only 7 windows, and it is reconstructed using only 2 input images within 48 seconds. Due to its simple facades, no model updating is needed. To export

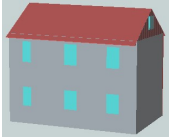

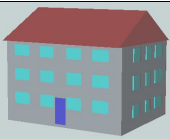


the model as CityGML format, the system takes an additional 19 seconds, primarily to perform rotation, translation, and scaling based on the building's true size and orientation in the real world, as well as coordinate transformation.

Building B has a bit more complex facade structure than Building A, with 16 windows and a door. However, the door is incorrectly detected as a window, likely due to similarities in their features. Hence, manually updating the door is required, which takes approximately 15 seconds. Despite its increased complexity, Building B is reconstructed by only one image, resulting in a shorter export time.

Building C has a larger number of windows than building B and uses 2 images for reconstruction. The testing results indicate that the semi-auto time is similar to building A's, with a similar scenario occurring in building D's case. As a matter of fact, more than half of the time is consumed by the user's interaction, but the detection of facade elements usually takes a similar amount of time regardless of facade complexity. This suggests that reducing the overall time can be achieved by improving the user's interaction proficiency. Since two facade elements are incorrectly detected, the user spends an additional 32 seconds on updating. Additionally, the exporting time (21 seconds) is similar to that of building A.

The appearance of balconies in building D and E makes their facade structures more complex, but the semi-auto time and exporting time remain stable compared to the previous cases. However, the updating time is affected by the quality of input images and the proficiency of user interaction. When doors or windows are obscured by vehicles, they cannot be detected correctly, which would increase the time and workload of manual updates. It is also difficult to measure the proficiency of user interaction because it depends on the users. Besides, the only noticeable change is the size of the outputted 3D model, which increases with the increment of facade complexity. But for the most complex building E, its output size is still less than 240KB.

Table 4.3. Summary of buildings with various sizes and façade complexities for sensitive analysis. The fifth column shows the interactive modelling time (semi-auto), time of interactively updating incorrect façade elements (update), exporting 3D model time (export) and updated façade elements in (·). Blue is for door, cyan is for window and green is for balcony.

No.	Num of imgs	Building size (L×W×H) m	Façade complexity	Modelling time (s) (semi-auto + update + export)	Output 3D model size (KB)	3D viewing
A	2	11×4.7×8.7	7 windows	48 + 0 + 19	87.9	
B	1	21.4×8.3×12.4	16 windows 1 door	24 + 15 + 11 (1 door)	112.7	
C	2	16×12.1×14.6	21 windows 1 door	49 + 32 + 21 (1 door + 1 window)	150.9	
D	2	11.7×10.4×20.7	17 windows 1 door 5 balconies	47 + 59 + 20 (1 door + 1 balcony + 2 windows)	162.5	
E	1	30.3×15.1×20.6	28 windows 4 doors 6 balconies	25 + 13 + 11 (1 balcony)	238.5	

On the other hand, the 3D viewing capability of VGI3D provides users with a smooth visualization experience, without any lagging. This is confirmed by the positive feedback received from participants in response to question 11, which asks if they agree that the 3D viewing capability of the model is smooth. All participants either select "Tend to agree" or "Strongly agree".

In a word to sum up, VGI3D exhibits low sensitivity to different building sizes and complexities, such as modelling time, 3D viewing capability, and output size of 3D models.

4.1.2 Roof plane segmentation in ALS point clouds

To segment roof plane in ALS point clouds, RoofNet network is proposed and implemented according to the description in Section 3.3.1 by using an open-source deep learning framework Pytorch (Paszke et al., 2019). RoofNTNU as a new roof dataset with standard

point density is established for training the RoofNet, and it is also intended to provide a general and robust segmentation capability on most ALS cloud data. Experiments on the RoofNTNU testing dataset show that the proposed method achieves promising segmentation results for both instance and semantic segmentation tasks.

Visual results of instance segmentation

The visual results produced from the RoofNet are compared with two other multi-task networks: ASIS and JSNet (Zhao and Tao, 2020). Since PointNet++ (used as backbone network) cannot perform instance segmentation, so it will not be compared here. Figure 4.4 shows the visual comparison results between ground truth and different methods. Roof type combination/7 is not included in the visual comparison, because it has been decomposed as the data augmentation strategy. More details about data augmentation can be found in Paper 4. Overall, the proposed method outperforms the other two methods. The first three columns of Figure 4.4 represent the most common and simplest roof structures in Norway and other countries. Due to their simple geometric structures, all three methods perform well on them. On the other hand, the last three columns correspond to the roof types specific to Norway and Western European countries, which are more complex. Although ASIS and JSNet segment T-element roofs well, segmentation errors occur at some boundaries between adjacent roof planes (as marked by the red circles in Figure 4.4). Some incorrectly classified points appear at the boundaries or the planes are over/under-segmented. Similarly, corner element and cross element roofs have similar errors. These errors are more noticeable due to their geometric complexity. In contrast, the RoofNet produces great and distinguishable segmentation results on all roof types due to the semantic-aware instance features, which help separate them in the instance embedding space. More instance segmentation results on the RoofNTNU testing dataset can be available in Appendix B of Paper 4.

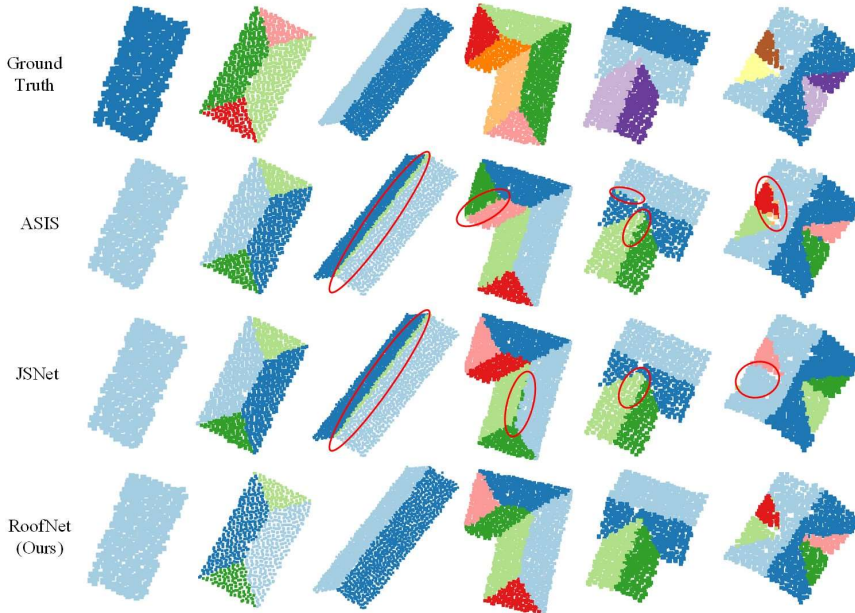


Figure 4.4. Visual comparison results of ASIS, JSNet and the proposed method on the instance segmentation task. Different colors stand for different instances (that is, roof planes). The colors of the same instance in ground truth and prediction are not necessarily the same.

Visual results of semantic segmentation

Visual semantic segmentation results are displayed in Figure 4.5 and compared with the baseline PointNet++, RoofNet, ASIS and JSNet. Once again, the method outperforms the other three methods overall. For the most common roof type, hipped, the other three methods struggle to segment the boundaries between different pairs of geometric shapes. A similar problem is observed on roof type corner element, but to a greater extent. The visualization reveals that JSNet produces over/under-segmentations at the boundaries and incorrectly classified points (row 4, column 4), including the result generated by the RoofNet. However, the proposed method still yields the best result with clear boundaries. This demonstrates that the improved result indeed benefits from the aggregation of instance embeddings, which enhances the distinguishability and accuracy of boundaries. In terms of incorrectly classified points, this problem occurs frequently in the results generated from ASIS and JSNet. ASIS fails to fuse semantic features adequately, while JSNet’s network structure is overly complex, leading to overly abstract features that do not aid semantic segmentation.

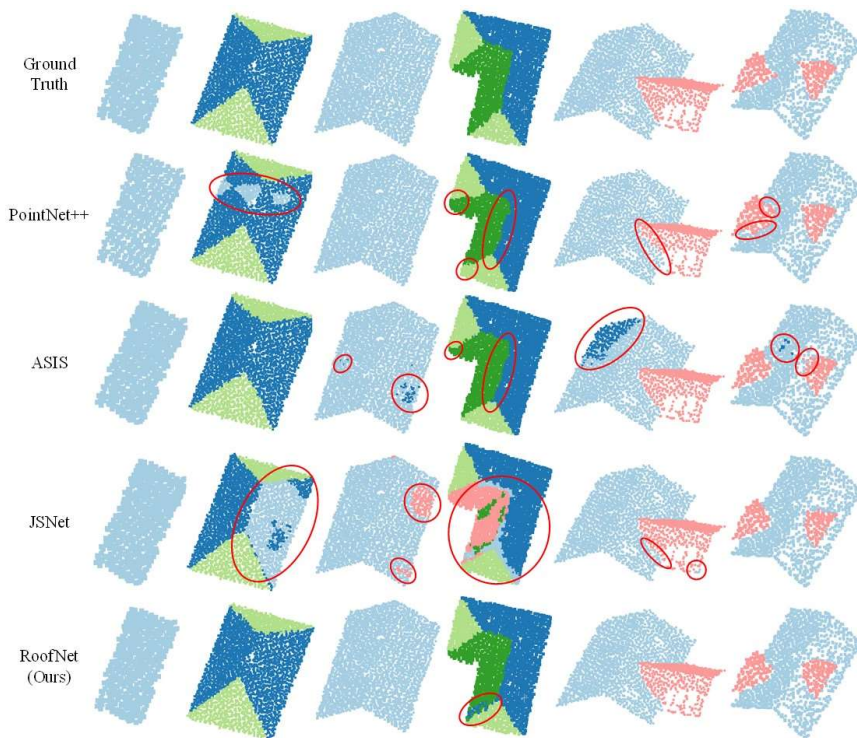


Figure 4.5. Visual comparison results of PointNet++, ASIS, JSNet and the proposed method on the semantic segmentation task. Different colors represent different pairs of roof planes with the same geometric shapes.

To quantitatively access the quality of roof segmentation results on RoofNTNU testing dataset, several evaluation metrics are employed. For instance segmentation evaluation, four metrics are computed and they are mean coverage (mCov), mean weighted coverage (mWCov) (Ren and Zemel, 2017), mean recall (mRec) with IoU threshold of 0.5, and mean precision (mPrec). Mean accuracy (mAcc), overall accuracy (oAcc), and mean intersection over union (mIoU) are computed for evaluating the semantic segmentation performance. The Cov score is used to measure the instance-wise IoU of the prediction matched with the ground truth, and it is then further weighted with the size of the ground truth instances to obtain WCov (Wang et al., 2019b). The equations for Cov and WCov with respect to ground truth regions GT and predicted regions P are given as follows:

$$Cov(GT, P) = \sum_{a=1}^{|GT|} \frac{1}{|GT|} \max_b IoU(r_a^{GT}, r_b^P) \quad (4-1)$$

$$WCov(GT, P) = \sum_{a=1}^{|GT|} w_a \max_b IoU(r_a^{GT}, r_b^P) \quad (4-2)$$

$$w_a = \frac{|r_a^{GT}|}{\sum_k |r_k^{GT}|} \quad (4-3)$$

Quantitative assessment of instance segmentation results

Table 4.4 presents the quantitative results of various methods on the instance segmentation task. The proposed method shows a slightly better performance than ASIS in terms of metrics mCov and mWCov, and a significant improvement of 1.4% in mPrec when evaluated on the testing dataset. This suggests that the FF module and modified JFL module have helped optimize the network structure, and the instance branch benefits more from the semantics branch, which aligns with the initial hypothesis. In contrast, JSNet is significantly inferior to the proposed method on all four metrics except for mPrec, which is still 2.3% lower than the proposed method. This may be because JSNet is primarily designed for indoor datasets (e.g., S3DIS; Armeni et al., 2016), where indoor scenes are more complex than roof structures. Therefore, JSNet’s architecture is designed with many convolution layers and other tensor operations. This reflects that complex and deep network structures may not necessarily improve feature learning on datasets with relatively simple scenes. Furthermore, the quantitative results also validate that the JFL module is appropriately designed and capable of enhancing the performance of instance segmentation.

Table 4.4. Quantitative results of instance segmentation on RoofNTNU dataset.

<i>Method</i>	<i>mCov (%)</i>	<i>mWCov (%)</i>	<i>mPrec (%)</i>	<i>mRec (%)</i>
ASIS	85.0	85.0	94.8	91.7
JSNet	66.7	66.5	93.9	71.6
RoofNet (Ours)	85.3	85.2	96.2	91.7

Quantitative assessment of semantic segmentation results

Table 4.5 shows the quantitative results of different methods on the semantic segmentation task. The proposed method achieves an mIoU of 86.7%, which significantly outperforms the PointNet++ baseline by 2.7%. In addition, RoofNet also considerably outperforms ASIS and JSNet in terms of mIoU. Regarding the metric mAcc, RoofNet shows a slight improvement compared to the baseline, but it demonstrates significant improvement when compared to ASIS and JSNet, with an improvement of 1.3% and 5.1% respectively. Similarly, there is an observable improvement in the oAcc metric, with the RoofNet surpassing the other three methods by 1.0%, 2.2%, and 5.2%, respectively. The proposed method employs PointNet++ as its backbone, and its superior performance indicates the ability to extract high-quality

features. The fused features produced by FF module further enhance the learning of semantic features, which are advantageous for the final semantic segmentation results.

Furthermore, in order to present a more objective evaluation of the performance of RoofNet and eliminate any bias due to data imbalance on two roof types (corner element/4 and cross element/6), the scores for each roof type have been calculated separately, and the detailed results can be found in Appendix A of Paper 4.

Table 4.5. Quantitative results of semantic segmentation on RoofNTNU dataset.

<i>Method</i>	<i>mAcc (%)</i>	<i>oAcc (%)</i>	<i>mIoU (%)</i>
PointNet++ (baseline)	93.8	95.4	84.0
ASIS	93.1	94.2	81.1
JSNet	89.3	91.2	66.4
RoofNet (Ours)	94.4	96.4	86.7

Q 2.2: How to analyze the effectiveness of the improved modules of the proposed RoofNet?

To assess the effectiveness of the improvements, a network architecture study involved in five variant experiments on key improvements is performed and discussed. This study only focuses on the improved components compared to ASIS, as ASIS has already demonstrated its advantages in terms of semantic awareness and instance aggregation. The essential part of the ASIS baseline and the details of the five experiment groups are shown in Figure 4.6, where their encoders and decoders are the same as RoofNet. In each group, the upper yellow bar represents the feature generated by the instance branch decoder, while the lower yellow bar refers to the feature generated by the semantic branch decoder.

- ASIS baseline. The ASIS network is regarded as the baseline (Figure 4.6a) and all other groups will compare the results with the baseline.
- Add FF module to both the semantic and instance branches immediately after the decoders; eliminate the upper parallel feature F_S from the semantic branch and transfer the fused semantic features into the instance feature space (see Figure 4.6b).
- Eliminate the upper parallel feature F_S from the semantic branch and transfer the fused semantic features to the instance feature space (see Figure 4.6c).
- Keep two parallel features F_S of the semantics branch, but incorporate an FF module after the upper parallel feature F_S ; then transfer the fused semantic features into instance feature space (see Figure 4.6d).
- Remain the semantic branch and semantic awareness consistent with RoofNet; deepen the instance branch to learn more high-level instance features (see Figure 4.6e), which is inspired by JSNet.

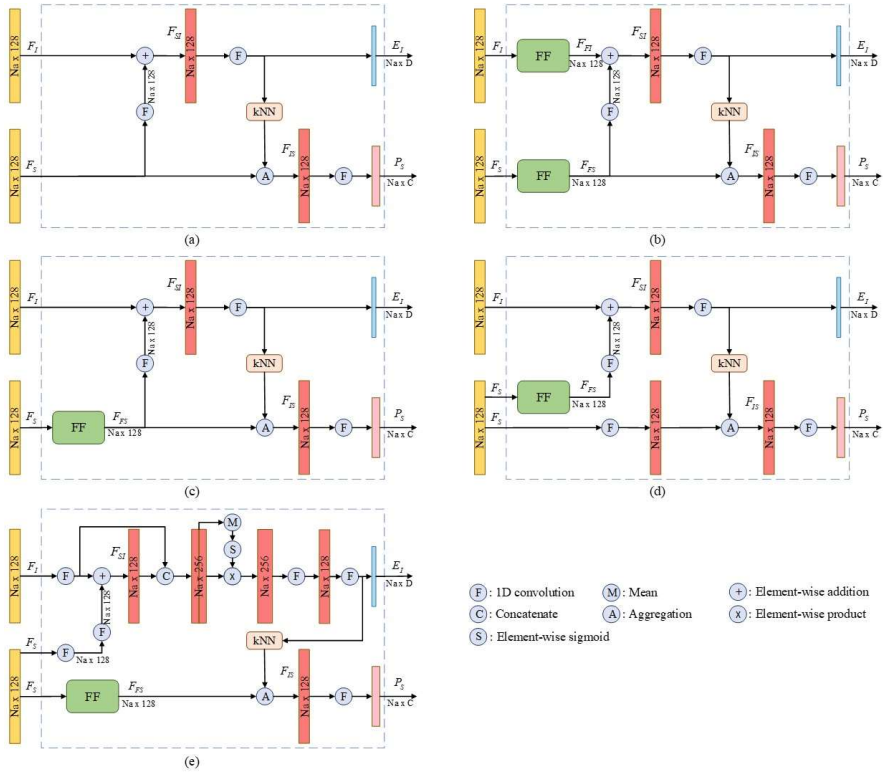


Figure 4.6. Key part of baseline ASIS (a) and four groups of experiments (b–e). Their encoders and decoders are consistent with RoofNet.

Results analysis

The experiments focus on answering three questions: (i) whether the instance branch requires the FF module (groups 1 and 2); (ii) where to add the FF module to the semantics branch (groups 3–6); (iii) whether it is beneficial to increase the depth of the instance branch (group 5).

Table 4.6 presents the results of all architecture study experiments. When comparing group 2 with the baseline, it shows that even though the FF modules do enhance semantic segmentation, they weakened the instance segmentation at the same time. This is because integrating fused semantic features into fused instance features may cause confusion in the network, particularly in the instance branch, about which features it should learn. Hence, the FF module was not added to the instance branch.

When comparing with group 2, the performance of group 3 is improved when the FF module is applied only to the semantic branch. This enhancement benefits both segmentation tasks by

providing more explicit information to the network about what features to learn for each branch. Specifically, it penalizes over-separated semantic features on the semantic branch and facilitates the learning of semantic-aware instance embeddings.

When comparing with group 3, it is less effective for group 4 by adding the FF module as a parallel semantic sub-branch and integrating it into the instance branch. The primary reason is that the aggregation operation (*Aggregation*) is directly applied to non-fused semantic features, which limits the learning of refined semantic features for each point. Hence, the structure of RoofNet (group 6 in Table 4.6) is designed to enable the semantic the semantics branch to learn refined semantic features for each point after the aggregation operation. In addition, thanks to the weight updating during backpropagation, adding non-fused semantic features to the instance branch benefits the entire instance branch from the FF module.

The deeper network architecture tested in group 5 is found to be less effective for roof plane segmentation. This is because the complexity of roof planes’ features is much lower compared to object features found in indoor/outdoor scenes. Applying a deeper and more complex network may result in overfitting and negatively affect the segmentation performance.

In summary, although RoofNet did not outperform all other groups on both segmentation tasks, the ultimate objective of it is to accurately segment roof planes or instances. Therefore, a compromise was made, and the network that demonstrated the best performance on instance segmentation was selected, i.e., RoofNet.

Table 4.6. Instance and semantic segmentation results of all experiments of architecture study on RoofNTNU.

Group	Instance Segmentation		Semantic Segmentation	
	<i>mWCov</i> (%)	<i>mPrec</i> (%)	<i>mAcc</i> (%)	<i>mIoU</i> (%)
(1) Baseline (ASIS)	85.0	94.8	93.1	81.1
(2)	83.6	94.8	94.8	88.3
(3)	84.1	95.9	94.9	88.8
(4)	83.0	94.9	93.0	85.1
(5)	82.8	92.5	93.4	85.9
(6) RoofNet (Ours)	85.2	96.2	94.4	86.7

4.2 Web-based 3D visualization platform

The results on the 3D platform’s implementation, rendering optimization, multi-source data integration and visualization are presented in this section. Paper 3 discusses a new approach for extracting and locating road objects to enhance the content richness of the 3D visualization platform and therefore, Paper 3 resolves research question 3.4.

Q 3: How to motivate volunteers and enhance their psychological comforts for more 3D building model contributions using a 3D visualization platform?

As a visualization platform aimed at motivating volunteers and enhancing their psychological comforts to contribute more 3D buildings through VGI3D, it should be able to digitalize a city environment by integrating different 3D data sources, to enable interaction with 3D objects and to have a fluid user experience. Additionally, as a research prototype, this platform should have good scalability by integrating other research methods to support 3D urban analysis, and should be easily implemented as well. Therefore, through this 3D visualization platform, on one hand, it can enhance the popularity of VGI3D and attract more volunteers to contribute 3D buildings; on the other hand, it can also provide convenience for relevant researchers, without requiring them to regenerate such large-scale 3D data and rebuild a new visualization platform for displaying research results.

The framework of 3D visualization platform is shown in Figure 4.7, which is basically an extension of the Cesium virtual globe. Most data sources (e.g., 3D buildings, 3D road networks and 3D road objects) are in CityGML format and are converted into b3dm format to represent 3D city objects on the web environment. High-resolution DTMs are tiled and transformed as Quantized Mesh format to represent 3D terrain. 2D imagery data is also included as base map. Most of these 3D data contain both geometry and semantic information, and they are then parsed and rendered in a web browser through Cesium engine. Users are able to interact with the 3D city models, such as inspecting semantic information on mouse-click, hiding and displaying desired 3D objects, and observing the 3D models from different view perspectives. Moreover, the combination of 3D terrain and base map can hugely increase the realism of the 3D visualization platform and improve the user experience as well.

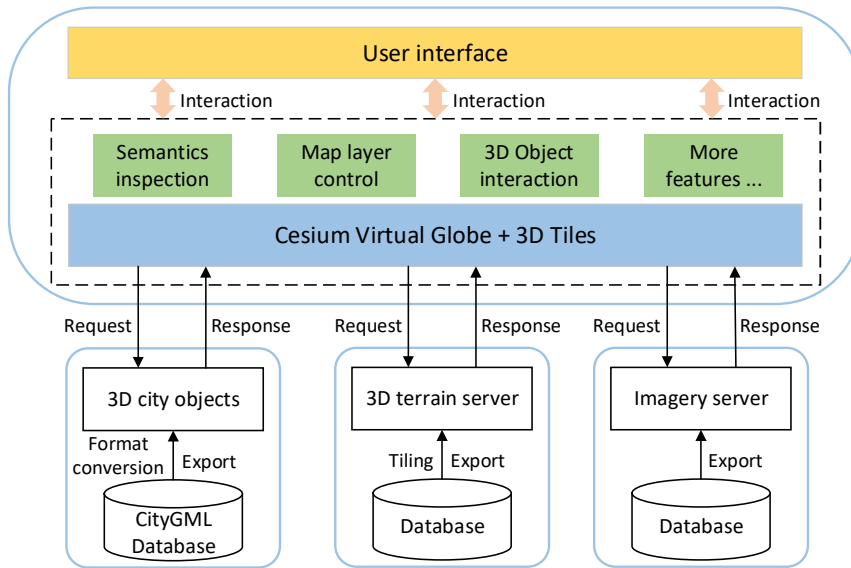
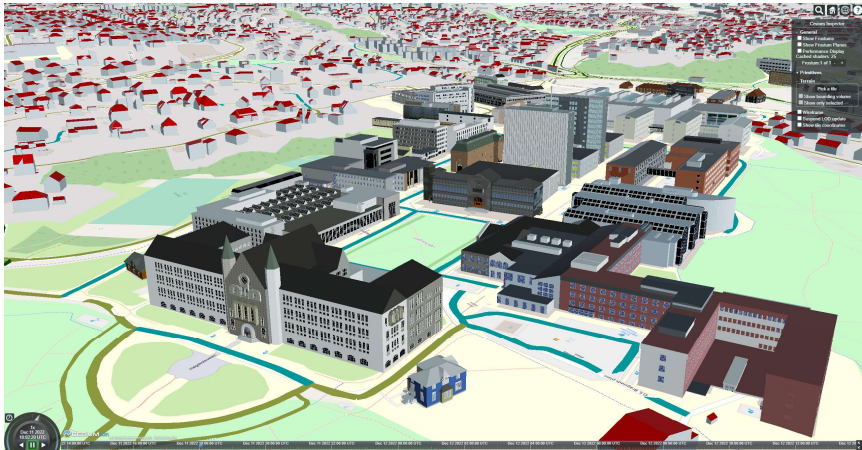
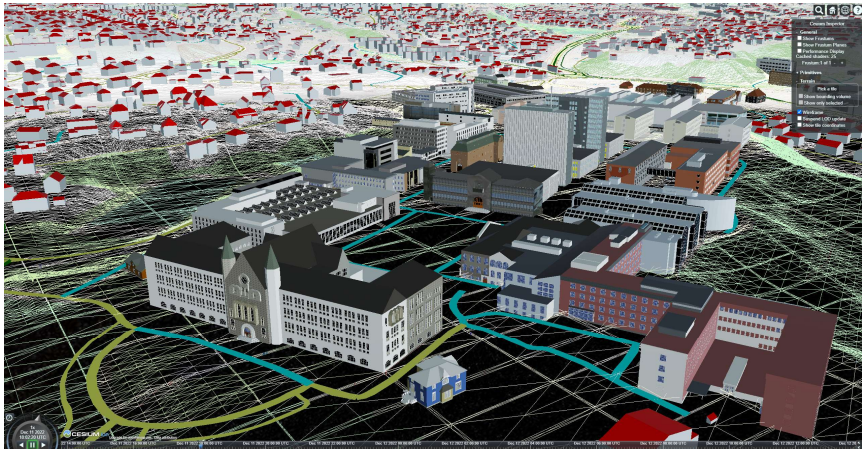


Figure 4.7. The framework of the web-based 3D visualization platform.

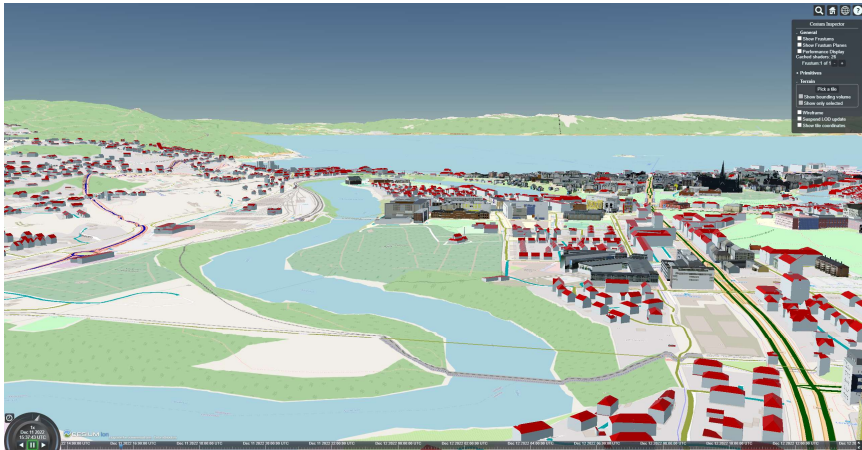
Furthermore, the 3D visualization platform is currently deployed on a physical Dell workstation, and can be accessed at <http://vgi.ibm.ntnu.no:8082/Apps/terrain/TRD3D.html/>. Visualization examples of the 3D platform with integrated 3D city objects are illustrated in Figure 4.8. For instance, Figure 4.8a displays a close-up view of Gløshaugen campus, NTNU, showcasing different elements of the 3D visualization platform. The 3D buildings with red roofs are represented in LoD2 and have semantics, thus being able to be interactive. Detailed building models with textures are derived and transformed from SketchUp models but they lack semantic information. The LoD3 building models are generated from VG13D and contain the semantic information. The road networks are represented with different colors indicating different road categories, but some segments of the roads are missing owing to 3D terrain's occlusion. This reflects a deficiency of the interpolation method for creating 3D road networks, because the elevation values of the 3D road networks are fixed and derived from the level 16 (18 levels in total) of the 3D terrain. Consequently, the 3D road networks cannot dynamically adjust their elevations as users zoom in or out. Finally, 3D terrain is represented as TIN and can be visually viewed in Figure 4.8b after switching to wireframe mode. Both Figure 4.8c and Figure 4.8d present a good perspective to better visualize and perceive the 3D terrain from a global view.



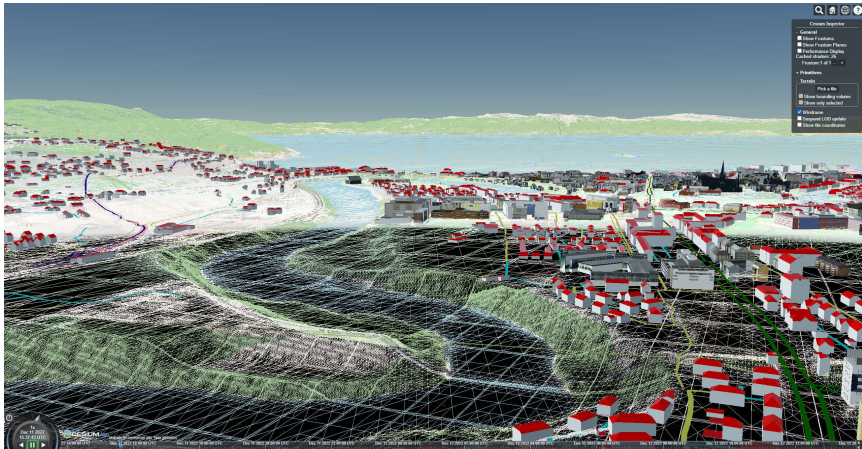
(a)



(b)



(c)



(d)

Figure 4.8. (a)-(b) provide a close-up view of Gløshaugen, NTNU, showcasing different elements of the 3D visualization platform, while (c)-(d) present a good perspective for better visualizing and perceiving the 3D terrain.

Q 3.1: How to render massive 3D data rapidly and efficiently in the web environment?

Rendering performance is one of the most important factors to consider during visualization. Faster loading time provides users with a more fluid user experience.

To render high-resolution 3D terrain data efficiently, a popular optimization strategy, TIN (Triangulated Irregular Network), is applied in this thesis. This technique involves the creation of a surface model using a set of non-overlapping triangles to represent the terrain

surface. The triangles are formed by connecting adjacent points on the terrain surface, and the shape of the triangles is determined by the shape of the terrain. For example, for the sloped terrain surfaces, it can use dense and small triangles to keep high resolution, while for the flat terrain surfaces, using sparse and sparse and large triangles to reduce the complexity. Each triangle is assigned an elevation value based on the average elevation of the points that define the triangle. In this way, a large amount of memory space and computing resources have been saved, making the TIN models more efficient at representing terrain surfaces with variable resolution, as the density of triangles can be adjusted based on the complexity of the terrain. In addition to the TIN optimization strategy, another rendering optimization mechanism, LoD, is also employed during runtime. It is aware that the concept of this LoD a scale level defined in a tiling scheme and hence it is different from the one defined by CityGML standard. Based on the distance between the user's viewpoint and the 3D terrain tiles, the high-resolution terrain tiles that are close to the viewpoint would be rendered, while only rendering low-resolution terrain tiles that are away from the viewpoint. As a result, it can allow for faster rendering and improved performance. An example of displaying 3D terrain data on the 3D visualization platform is shown in Figure 4.9.

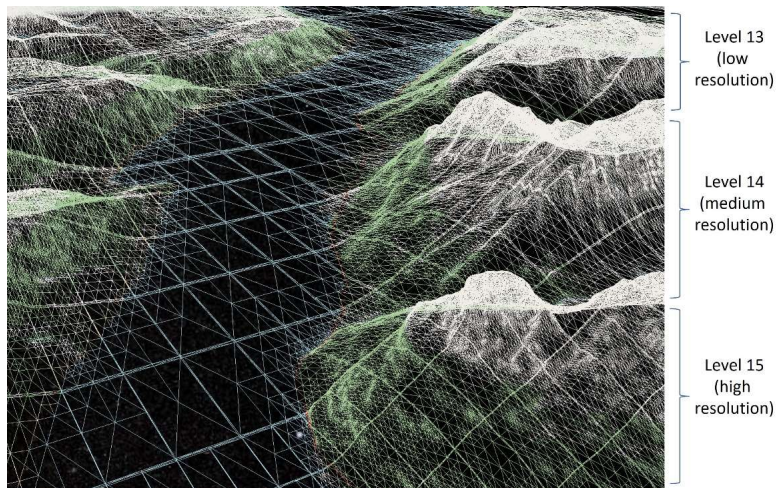


Figure 4.9. Example of 3D terrain visualization using two rendering optimization strategies.

To rapidly render large amount of 3D city models such as 3D buildings and 3D road networks, 3D Tiles technology is utilized due to its excellent performance for visualization. 3D models are first divided into a series of hierarchical tiles based on geometric errors and 3D models' distribution, each tile representing a smaller portion of the overall 3D models. Geometric errors are calculated based on the distance between tiles and viewpoint. This means that when the user's viewpoint moves closer to the 3D models, the tiles with higher detailed tiles are loaded and rendered on-the-fly, while lower detailed tiles are culled to

improve performance and reduce the network bandwidth. The tiles are usually compressed for faster data transmission over the network. Furthermore, 3D Tiles would not affect the semantic information or attributes embedded in the 3D city models and support interactive selection and inspection.

Q 3.2: How to render initial 2D road segments on high-resolution 3D terrain in a cartographic manner?

Roads in GIS are usually stored as a collection of 2D polylines. To render them on high-resolution 3D terrain in a cartographic way, a geometry-based approach is developed in this thesis. One of the core parts of this approach is to find out the intersected/additional vertices against the 3D terrain quickly and accurately, as mentioned in Figure 3.6 of Section 3.4. For this purpose, the 3D terrain tiles near the road segments are firstly identified, and then all triangles comprising the selected 3D terrain tiles are projected to ground (xy plane). Next, by taking the 2D triangles as input, an AABB tree is constructed to insert additional vertices at each intersection between road segments and triangles, which is so-called collision detection algorithm applied in this thesis. As long as the actual elevation values are reassigned to the vertices, the 3D road polylines can be eventually obtained.

Moreover, to render cartographic 3D roads, it has to expand the 3D polylines to the 3D polygons and at the same time, to remain the curves as smooth as possible. To achieve this, Catmull-Rom spline algorithm is employed, making the original sharp curves become smooth enough. Figure 4.10 shows the final cartographic roads on high-resolution 3D terrain.

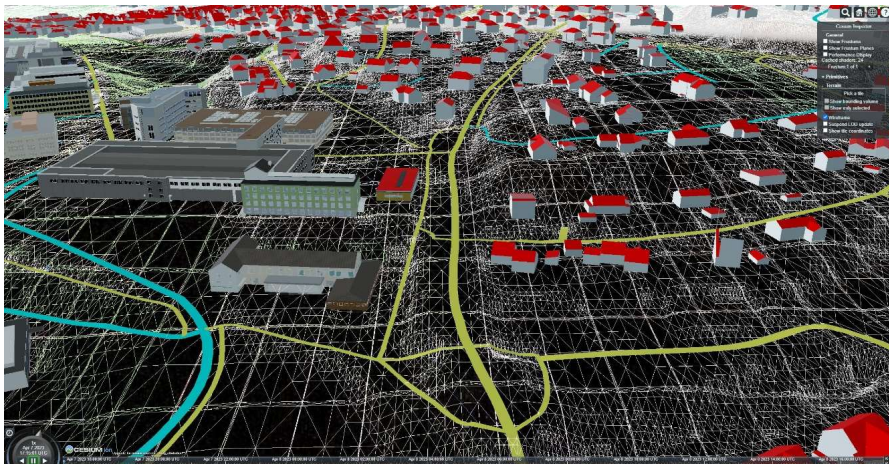


Figure 4.10. Cartographic rendering of roads on high-resolution 3D terrain in wireframe mode, in Trondheim, Norway. Different colors represent the different road categories.

Q 3.3: How to enrich the 3D road-related objects of the visualization platform, such as traffic lights and signs at road intersections?

A case study is carried out to automatically detect and localize the traffic lights and signs at road intersections by using the method introduced in Section 3.4. To evaluate the performance of the proposed method, the result of semantic segmentation is firstly assessed because the ATBT is created based on the semantic segmentation results. In other words, the quality of semantic segmentation results would significantly affect the establishment of ATBT, and then further influence the localization of desired road objects. Therefore, a quantitative comparison between PSPNet (the trained network in this thesis) and a competitive network, DeepLabv3+ (Chen et al., 2018), as illustrated in Table 4.7. It shows that the trained PSPNet network outperforms the DeepLabv3+ in terms of both evaluation metrics mean IoU and pixel accuracy (acc).

Table 4.7. Quantitative comparison of mIoU and pixel accuracy between PSPNet and DeepLabv3+.

Method	Mean IoU (%)	Pixel Acc. (%)
PSPNet (ours)	34.17	91.3
DeepLabv3+	33.97	90.2

Regarding the evaluation of localization task, two elements of spatial data quality need to be assessed: completeness and positional accuracy. Although positional accuracy is the most established indicator of mapping accuracy, official ground truth position data for traffic signs and lights are unavailable. As a result, I manually collect the locations of traffic signs/lights from Google Maps through visual observation, and make them as “reference data” to evaluate the completeness and positional accuracy of my localization results. 100 intersections featuring either four or three branches are tested in the experiment, containing more than 350 image sequences and over 3400 images. In terms of completeness level, over 97% of completeness is achieved in all 100 testing intersections. Please note that only when all traffic lights and signs are detected and their predicted locations are not far away from the actual locations at an intersection, then it is considered as a complete and correct case. Figure 4.11 shows the comparative localization results for two examples of one crossroad and one T-junction, where the first column represents the visual results generated by the proposed method and the second column is the manually collected “reference data”. Red dot 1 in the right figure of (a) contains three signs, and each of the red dots 1,2,3 in the right figure of (b) contains two signs because they are overlapped. As seen in the figures, both examples demonstrate approximate positional accuracy compared to the annotated “reference data”.

Implementation, results and discussion

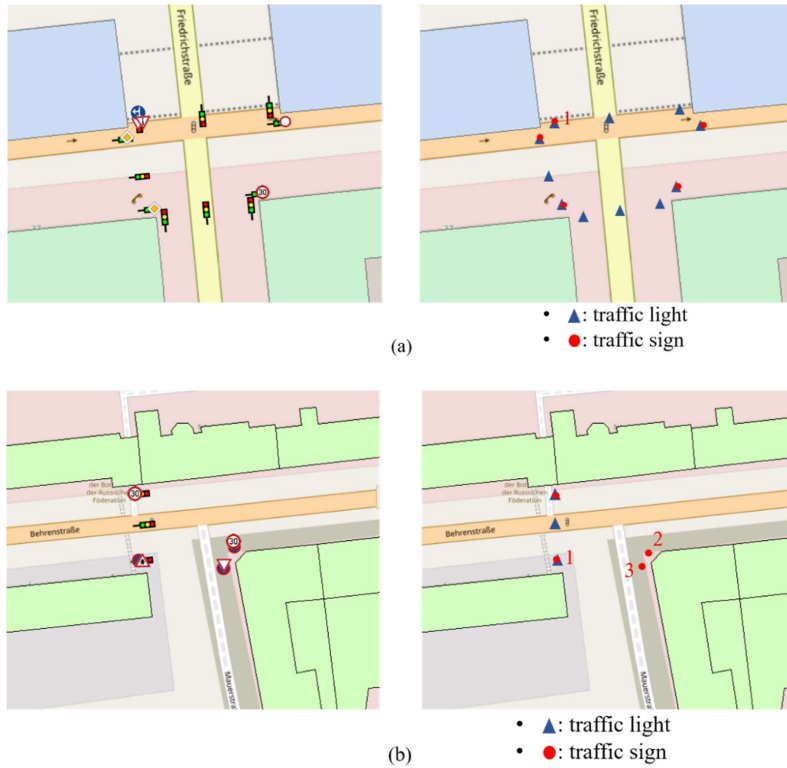


Figure 4.11. Visual comparison of traffic lights and signs localization results. The first column represents the results generated by our proposed algorithm. The second column refers to the manually collected “reference data” from Google Maps.

Chapter 5

Conclusion and future research

5.1 Conclusion

This thesis concentrates on the reconstruction of LoD3 3D building models with semantic information by means of crowdsourcing. The 3D building modelling at LoD3 can be mainly divided into two tasks: detailed façade reconstruction and high accuracy roof reconstruction. To achieve these, novel 3D reconstruction approaches on façade level and roof level are studied respectively, and then their modelling results are combined to eventually generate LoD3 building models with semantics.

On the façade modelling level, an interactive and web-based solution, VGI3D, is proposed for 3D building modelling from street-level images, particularly focusing on reconstructing detailed façade models. With the ambition of becoming a VGI platform to collect 3D building models with semantics, it takes simple interoperability, low cost, generality and robustness into consideration. All these efforts make the VGI3D require less user input and operations and meanwhile, make it be able to perform well on most of modelling cases as much as possible. It is appropriate for rapid modelling while still retaining a high level of detail (i.e., LoD3). Furthermore, functional and usability testing are also carried out with a limited number of participants among experts and non-experts. This aims to evaluate the usefulness of the VGI3D for the 3D building modelling community and to better optimize the system and user experience as well. The findings indicate that most of participants believe it holds promising value for the 3D building modelling community, regardless of whether they have rich 3D modelling experience or not. That is exciting feedback for this work, since it will be able to have the potential to motivate volunteers for more data contribution and to promote the VGI3D to the mass market, leading to increased contributions of 3D building models for smart city-related applications.

On the roof modelling level, although VGI3D is able to automatically generate the roof model by selecting a predefined roof type, its geometric accuracy cannot be guaranteed. By contrast, ALS point clouds can provide the accurate geometric information about roof structures for roof modelling. On the other hand, since roof plane segmentation is an essential step in the process of 3D roof reconstruction, so an improved deep learning-based network, RoofNet, is proposed for better segmenting roof planes in ALS point clouds. The network consists of a shared encoder and two parallel decoders for rich features extraction, an FF module for semantic feature fusion, and a JFL module for simultaneous learning of semantic and instance features. Moreover, a new roof plane training dataset, RoofNTNU, is manually created from standard ALS point clouds for training the proposed network. The dataset contains 930 roof samples covering seven typical roof structures in Trondheim, Norway. The

Conclusions and future research

proposed network achieves good performance on both instance (i.e., separated roof planes) and semantic (i.e., groups of roof planes with similar geometric shapes) segmentation tasks, as demonstrated by the results of the architecture study. Experiments on the RoofNTNU dataset show that the proposed method obtains a mean precision (mPrec) of 96.2% for instance segmentation and a mean accuracy (mAcc) of 94.4% for semantic segmentation. These promising results demonstrate the potential of the approach to accurately segment roof planes in residential regions of Norway and some other West European countries by using standard ALS point clouds.

Furthermore, to motivate volunteers and enhance their psychological comforts for more 3D building contributions via VGI3D, a web-based 3D visualization platform is developed by digitalizing a city environment and integrating multiple 3D city models, such as 3D terrain, 3D building models, 3D road networks and 3D road objects. Based on this platform, it enables to provide the immediate visual satisfaction for volunteers when interacting with their made 3D buildings and other 3D objects in a 3D scene, and in turn, facilitating volunteers to strengthen the sense of accomplishment and to increase their motivations to participate in the VGI3D project and contribute 3D data. Besides, this thesis also would like to build the 3D visualization platform as a general research platform, where the ideas or methods proposed by other researchers can be easily implemented, validated and extended on this platform to support their research work, for instance 3D urban analysis and applications. Hence, through this 3D visualization platform, it can not only enhance the popularity of VGI3D but also offer convenience for relevant researchers, with no need to regenerate such a large-scale 3D data and rebuild a new platform for visualizing the research outcomes.

Finally, the deeper development of smart cities in the future will show a huge demand for detailed 3D building models, benefiting various aspects of people's daily life. The global coverage of detailed 3D buildings can only be achieved through crowdsourcing, because many poor and developing countries and regions lack the capability and resources to accomplish this demanding task. Both will undoubtedly provide immense motivations for global volunteers to contribute data.

5.2 Limitations and future research

Regarding the VGI3D, it still has much room for improvement and optimization as the future research:

- It is necessary to further enhance the CNN model used for façade elements detection to get more accurate detection results on most input images, so that it can reduce the costs of user interaction for 3D model's error correction.
- The number of participants involving in usability testing is relatively small. Their responses and comments may not represent the views of all professionals working in

the field of 3D building modelling. Therefore, usability testing with larger samples is needed especially conducting on related practitioners in the future, as their feedbacks are rather valuable and crucial.

- Except for visually comparing the reconstructed building models with the real buildings, it is difficult to quantitatively evaluate how good the reconstruction is, because there are no existing high-quality 3D building models as ground truth. So the quantitative assessment of the reconstructed models can be a direction of future research.
- The building heights are not available except for in Trondheim, Norway. As a result, 3D buildings reconstructed elsewhere have inaccurate heights, because their heights are roughly estimated by entering the number of floors during the user interaction. It may be addressed to some degree by collecting and importing the public governmental data associated with the buildings into VGI3D's database.
- Building footprints from countries other than Norway, Sweden, and Italy are currently unavailable. It has to enlarge the database of building footprints by adding footprints from for example the rest of European nations, so as to promote VGI3D more extensively.

Regarding the RoofNet for roof plane segmentation, several aspects will be improved in the future:

- A place worthy of improvement is the joint features learning (JFL) module, as the current use of kNN has limitations in the selection of k -value and distance metric. In the future study, the network will be further optimized to overcome this problem and to obtain more accurate segmentation results.
- The amount of two roof types (cross element and corner element) in the RoofNTNU dataset is too small compared to other roof types, leading to the data imbalance issue. Consequently, the thesis will also plan to enrich the dataset on these two roof types.
- The current RoofNet cannot directly segment combination roofs (roof type 7) due to the complexity and diversity of their geometric structures. It is necessary to increase the general ability of the method to handle with hard cases in the future study.
- Furthermore, the key/corner points of each roof plane will be extracted and the topological relations among them will be adjusted to form the geometric surfaces, implementing the final goal of roof reconstruction.

Regarding the 3D visualization platform,

- The main objective of developing a 3D visualization platform is to enhance VGI3D users' immediate visual satisfaction and psychological comforts through a virtual 3D city environment, thus giving them motivations for more data contributions. However, whether visual perception and psychological comforts are improved have not been

Conclusions and future research

verified yet. Thus, research from the perspectives of visibility engineering psychology and cognitive science will be conducted to analyze users' visual perception and psychological comforts towards 3D visualization platform. For instance, a combination of virtual reality (VR) technique and questionnaire is a potential method to apply.

- The 3D terrain tilesets consist of 18 zoom levels, and currently, the elevation values of the 3D road networks are fixed and derived from the level 16 of the 3D terrain. They cannot be dynamically adjusted as users zoom in or out. Therefore, as the 3D terrain zooms in or out, dynamically adjusting and rendering 3D road networks will be a future optimization direction.
- Currently, 3D building models generated by VGI3D cannot be immediately delivered and visualized in 3D visualization platform for some technical reasons caused by Cesium. Instead, they have to be uploaded to the visualization platform in batches. Hence, that would be great in the future to implement a seamlessly linking mechanism between VGI3D and 3D visualization platform for “what you see is what you modelled”, which means volunteers will be able to see and interact with their 3D buildings right after the modelling in a 3D geographic scene.

Last but not the least, the VGI3D will be relocated to Trondheim municipality's website and shared as a citizen participation project. The announcement of the host change will be publicly released when the new website is ready to access.

Bibliography

- Nelson, J. R., & Grubestic, T. H., 2020. The use of LiDAR versus unmanned aerial systems (UAS) to assess rooftop solar energy potential. *Sustainable Cities and Society*, 61, 102353.
- The Paris Agreement, 2022. <https://unfccc.int/process-and-meetings/the-paris-agreement/the-paris-agreement> (Accessed 23rd August 2022).
- Zhi, G., Liao, Z., Tian, W., & Wu, J., 2020. Urban flood risk assessment and analysis with a 3D visualization method coupling the PP-PSO algorithm and building data. *Journal of environmental management*, 268, 110521.
- Groves, P. D., Jiang, Z., Rudi, M., & Strode, P., 2013. A portfolio approach to NLOS and multipath mitigation in dense urban areas. *The Institute of Navigation*.
- Park, Y., Guldmann, J. M., & Liu, D., 2021. Impacts of tree and building shades on the urban heat island: Combining remote sensing, 3D digital city and spatial regression approaches. *Computers, Environment and Urban Systems*, 88, 101655.
- Li, Z., Lin, B., Zheng, S., Liu, Y., Wang, Z., & Dai, J., 2020. A review of operational energy consumption calculation method for urban buildings. In *Building Simulation* (Vol. 13, No. 4, pp. 739-751). Tsinghua University Press.
- Redweik, P., Teves-Costa, P., Vilas-Boas, I., & Santos, T., 2017. 3D city models as a visual support tool for the analysis of buildings seismic vulnerability: The case of Lisbon. *International Journal of Disaster Risk Science*, 8(3), 308-325.
- Zeng, P., Sun, X., Xu, Q., Li, T., & Zhang, T., 2021. 3D probabilistic landslide run-out hazard evaluation for quantitative risk assessment purposes. *Engineering Geology*, 293, 106303.
- Ulvi, A., 2021. Documentation, Three-Dimensional (3D) Modelling and visualization of cultural heritage by using Unmanned Aerial Vehicle (UAV) photogrammetry and terrestrial laser scanners. *International Journal of Remote Sensing*, 42(6), 1994-2021.
- Solar energy potential in Helsinki, 2023. <https://kartta.hel.fi/3d/solar/#/> (Accessed 23rd January 2023).
- Kilsedar, C. E., Fissore, F., Pirotti, F., & Brovelli, M. A. (2019). 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, 669-673.
- Miura, S., Hisaka, S., & Kamijo, S., 2013. GPS multipath detection and rectification using 3D maps. In *16th International IEEE Conference on Intelligent Transportation Systems* (pp. 1528-1534). IEEE.
- Buyukdemircioglu, M., & Kocaman, S. (2020). Reconstruction and efficient visualization of heterogeneous 3D city models. *Remote Sensing*, 12(13), 2128.
- Sangiambut, S., & Sieber, R. (2016). The V in VGI: Citizens or civic data sources. *Urban Planning*, 1(2), 141-154.
- Wang, Z., & Zipf, A., 2017. Using openstreetmap data to generate building models with their inner structures for 3d maps. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 4, 411.

Bibliography

- Neuhold, G., Ollmann, T., Rota Bulò, S., & Kotschieder, P., 2017. The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision* (pp. 4990-4999).
- Paz, D., Zhang, H., Li, Q., Xiang, H., & Christensen, H. I., 2020. Probabilistic semantic mapping for urban autonomous driving applications. In *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 2059-2064). IEEE.
- Steiger, E., Resch, B., & Zipf, A., 2016. Exploration of spatiotemporal and semantic clusters of Twitter data using unsupervised neural networks. *International Journal of Geographical Information Science*, 30(9), 1694-1716.
- Google 3D Warehouse, 2023. <https://3dwarehouse.sketchup.com/> (Accessed on 13rd January 2023).
- Goetz, M., & Zipf, A. (2013). The evolution of geo-crowdsourcing: bringing volunteered geographic information to the third dimension. In *Crowdsourcing geographic knowledge* (pp. 139-159). Springer, Dordrecht.
- Knerr, T. (2019). OSM2World Create 3D models from OpenStreetMap. Available: <http://osm2world.org/>
- OSM Buildings, 2023. <http://osmbuildings.org/> (Accessed 13rd January 2023).
- Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., & Szeliski, R. (2011). Building rome in a day. *Communications of the ACM*, 54(10), 105-112.
- Wolberg, G., & Zokai, S. (2018). PhotoSketch: a photocentric urban 3D modeling system. *The Visual Computer*, 34(5), 605-616.
- Zhang, C., Fan, H., & Kong, G. (2021). VGI3D: an interactive and low-cost solution for 3D building modelling from street-level VGI images. *Journal of Geovisualization and Spatial Analysis*, 5, 1-16.
- Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- Kong, G., & Fan, H. (2020). Enhanced facade parsing for street-level images using convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 59(12), 10519-10531.
- Kim, H., & Han, S. (2018). Interactive 3D building modeling method using panoramic image sequences and digital map. *Multimedia tools and applications*, 77, 27387-27404.
- Gilani, S. A. N., Awrangjeb, M., & Lu, G. (2018). Segmentation of airborne point cloud data for automatic building roof extraction. *GIScience & remote sensing*, 55(1), 63-89.
- Zhang, C., & Fan, H. (2022). An Improved Multi-Task Pointwise Network for Segmentation of Building Roofs in Airborne Laser Scanning Point Clouds. *The Photogrammetric Record*, 37(179), 260-284.
- Yi, L., Kim, V. G., Ceylan, D., Shen, I. C., Yan, M., Su, H., Lu, C., Huang, Q., Sheffer, A., & Guibas, L. (2016). A scalable active framework for region annotation in 3d shape collections. *ACM Transactions on Graphics (ToG)*, 35(6), 1-12.
- Mao, J., Wang, X., & Li, H. (2019). Interpolated convolutional networks for 3d point cloud understanding. In *Proceedings of the IEEE/CVF international conference on computer vision* (pp. 1578-1587).

- Rottensteiner, F., Sohn, G., Jung, J., Gerke, M., Baillard, C., Benitez, S., & Breitkopf, U. (2012). The ISPRS benchmark on urban object classification and 3D building reconstruction. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences 1-3, Nr. 1, 1(1)*, 293-298.
- Zolanvari, S. M., Ruano, S., Rana, A., Cummins, A., da Silva, R. E., Rahbar, M., & Smolic, A. (2019). DublinCity: Annotated LiDAR point cloud and its applications. *arXiv preprint arXiv:1909.03613*.
- Kölle, M., Laupheimer, D., Schmohl, S., Haala, N., Rottensteiner, F., Wegner, J. D., & Ledoux, H. (2021). The Hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and Multi-View-Stereo. *ISPRS Open Journal of Photogrammetry and Remote Sensing, 1*, 100001.
- Kada, M. (2007). Scale-dependent simplification of 3D building models based on cell decomposition and primitive instancing. *International Conference on on Spatial Information Theory*, Springer, Berlin. Heidelberg. 222-237.
- Wang, X., Liu, S., Shen, X., Shen, C., & Jia, J. (2019b). Associatively segmenting instances and semantics in point clouds. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition* (pp. 4096-4105).
- CloudCompare. 2022. CloudCompare-Open Source project. <http://www.cloudcompare.org/> (Accessed 20th December 2022).
- Gröger, G., & Plümer, L. (2012). CityGML–Interoperable semantic 3D city models. *ISPRS Journal of Photogrammetry and Remote Sensing, 71*, 12-33.
- Tang, L., Ying, S., Li, L., Biljecki, F., Zhu, H., Zhu, Y., Yang, F., & Su, F. (2020). An application-driven LOD modelling paradigm for 3D building models. *ISPRS Journal of Photogrammetry and Remote Sensing, 161*, 194-207.
- Biljecki, F., Ledoux, H., & Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems, 59*, 25-37.
- Stadler, A., Nagel, C., König, G., & Kolbe, T. H. (2009). Making interoperability persistent: A 3D geo database based on CityGML. In *3D Geo-information sciences* (pp. 175-192). Springer, Berlin, Heidelberg.
- Furukawa, Y., & Ponce, J. (2009). Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence, 32(8)*, 1362-1376.
- Goodchild, M. F. (2007). Citizens as sensors: the world of volunteered geography. *GeoJournal, 69*, 211-221.
- Zhou, X., Zhao, Y., Li, G., & Zhang, P. (2018). Crowdsourcing Spatio-Temporal Data Model Considering Reputation. *Geomatics and Information Science of Wuhan University, 43(1)*, 10-16.
- Juhász, L., & Hochmair, H. H. (2016). User contribution patterns and completeness evaluation of Mapillary, a crowdsourced street level photo service. *Transactions in GIS, 20(6)*, 925-947.
- Barrington-Leigh, C., & Millard-Ball, A. (2017). The world's user-generated road map is more than 80% complete. *PloS one, 12(8)*, e0180698.
- Zhang, H., & Malczewski, J. (2018). Accuracy evaluation of the Canadian OpenStreetMap road networks. *International Journal of Geospatial and Environmental Research, 5(2)*.
- Mohammadi, N., & Sedaghat, A. (2021). A framework for classification of volunteered geographic data based on user's need. *Geocarto International, 36(11)*, 1276-1291.

Bibliography

- See, L., Mooney, P., Foody, G., Bastin, L., Comber, A., Estima, J., ... & Rutzinger, M. (2016). Crowdsourcing, citizen science or volunteered geographic information? The current state of crowdsourced geographic information. *ISPRS International Journal of Geo-Information*, 5(5), 55.
- Lotfian, M., Ingensand, J., & Brovelli, M. A. (2020). A framework for classifying participant motivation that considers the typology of citizen science projects. *ISPRS International Journal of Geo-Information*, 9(12), 704.
- Yan, Y., Feng, C. C., Huang, W., Fan, H., Wang, Y. C., & Zipf, A. (2020). Volunteered geographic information research in the first decade: A narrative review of selected journal articles in GIScience. *International Journal of Geographical Information Science*, 34(9), 1765-1791.
- Fan, H., Kong, G., & Yang, A. (2022). Current status and prospects of research for volunteered geographic information. *Acta Geodaetica et Cartographica Sinica*, 51(7), 1653.
- Girres, J. F., & Touya, G. (2010). Quality assessment of the French OpenStreetMap dataset. *Transactions in GIS*, 14(4), 435-459.
- Forghani, M., & Delavar, M. R. (2014). A quality study of the OpenStreetMap dataset for Tehran. *ISPRS International Journal of Geo-Information*, 3(2), 750-763.
- Wu, H., Lin, A., Clarke, K. C., Shi, W., Cardenas-Tristan, A., & Tu, Z. (2021). A comprehensive quality assessment framework for linear features from Volunteered Geographic Information. *International Journal of Geographical Information Science*, 35(9), 1826-1847.
- Balducci, F. (2021). Is OpenStreetMap a good source of information for cultural statistics? The case of Italian museums. *Environment and Planning B: Urban Analytics and City Science*, 48(3), 503-520.
- Brovelli, M. A., & Zamboni, G. (2018). A new method for the assessment of spatial accuracy and completeness of OpenStreetMap building footprints. *ISPRS International Journal of Geo-Information*, 7(8), 289.
- Xu, Y., Chen, Z., Xie, Z., & Wu, L. (2017). Quality assessment of building footprint data using a deep autoencoder network. *International Journal of Geographical Information Science*, 31(10), 1929-1951.
- Zhou, Q. (2018). Exploring the relationship between density and completeness of urban building data in OpenStreetMap for quality estimation. *International Journal of Geographical Information Science*, 32(2), 257-281.
- Zacharopoulou, D., Skopeliti, A., & Nakos, B. (2021). Assessment and Visualization of OSM Consistency for European Cities. *ISPRS International Journal of Geo-Information*, 10(6), 361.
- Muttaqien, B. I., Ostermann, F. O., & Lemmens, R. L. (2018). Modeling aggregated expertise of user contributions to assess the credibility of OpenStreetMap features. *Transactions in GIS*, 22(3), 823-841.
- Jacobs, K. T., & Mitchell, S. W. (2020). OpenStreetMap quality assessment using unsupervised machine learning methods. *Transactions in GIS*, 24(5), 1280-1298.
- Zhang, D., Ge, Y., Stein, A., & Zhang, W. B. (2021). Ranking of VGI contributor reputation using an evaluation-based weighted pagerank. *Transactions in GIS*, 25(3), 1439-1459.
- Zielstra, D., & Hochmair, H. H. (2013). Positional accuracy analysis of Flickr and Panoramio images for selected world regions. *Journal of Spatial Science*, 58(2), 251-273.
- Senaratne, H., Bröring, A., & Schreck, T. (2013). Using reverse viewshed analysis to assess the location correctness of visually generated VGI. *Transactions in GIS*, 17(3), 369-386.

- Bagheri, H., Schmitt, M., & Zhu, X. (2019). Fusion of multi-sensor-derived heights and OSM-derived building footprints for urban 3D reconstruction. *ISPRS International Journal of Geo-Information*, 8(4), 193.
- Uden, M., & Zipf, A. (2013). Open building models: Towards a platform for crowdsourcing virtual 3D cities. *Progress and new trends in 3D geoinformation sciences*, 299-314.
- Fan, H., & Zipf, A. (2016). Modelling the world in 3D from VGI/Crowdsourced data. *European handbook of crowdsourced geographic information*, 435.
- Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., Van Gool, L., & Purgathofer, W. (2013). A survey of urban reconstruction. In *Computer graphics forum* (Vol. 32, No. 6, pp. 146-177).
- Liu, C., Kong, D., Wang, S., Wang, Z., Li, J., & Yin, B. (2021). Deep3D reconstruction: Methods, data, and challenges. *Frontiers of Information Technology & Electronic Engineering*, 22(5), 652-672.
- Henn, A., Gröger, G., Stroh, V., & Plümer, L. (2013). Model driven reconstruction of roofs from sparse LIDAR point clouds. *ISPRS Journal of photogrammetry and remote sensing*, 76, 17-29.
- Xiong, B., Elberink, S. O., & Vosselman, G. (2014). A graph edit dictionary for correcting errors in roof topology graphs reconstructed from point clouds. *ISPRS Journal of photogrammetry and remote sensing*, 93, 227-242.
- Buyukdemircioglu, M., Kocaman, S., & Isikdag, U. (2018). Semi-automatic 3D city model generation from large-format aerial images. *ISPRS International Journal of Geo-Information*, 7(9), 339.
- Li, Z., & Shan, J. (2022). RANSAC-based multi primitive building reconstruction from 3D point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 185, 247-260.
- Fischler, M. A., & Bolles, R. C. (1981). Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6), 381-395.
- Müller, P., Gang, Z., Wonka, P., & Gool, L. J. V. (2007). Image-based procedural modeling of facades. *ACM Transactions on Graphics*, 26(3), 85.
- Koutsourakis, P., Simon, L., Teboul, O., Tziritas, G., & Paragios, N. (2009). Single view reconstruction using shape grammars for urban environments. In *2009 IEEE 12th international conference on computer vision* (pp. 1795-1802). IEEE.
- Dehbi, Y., & Plümer, L. (2011). Learning grammar rules of building parts from precise models and noisy observations. *ISPRS journal of photogrammetry and remote sensing*, 66(2), 166-176.
- Gasde, R., Marlet, R., & Paragios, N. (2016). Learning grammars for architecture-specific facade parsing. *International Journal of Computer Vision*, 117, 290-316.
- Buyukdemircioglu, M., Kocaman, S., & Kada, M. (2022). Deep learning for 3D building reconstruction: A review. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 359-366.
- Wang, Y., Fan, H., & Zhou, G. (2020). Reconstructing facade semantic models using hierarchical topological graphs. *Transactions in GIS*, 24(4), 1073-1097.
- Tarsha-Kurdi, F., Landes, T., Grussenmeyer, P., & Koehl, M. (2007). Model-driven and data-driven approaches using LIDAR data: Analysis and comparison. In *ISPRS workshop, photogrammetric image analysis (PIA07)* (pp. 87-92).

Bibliography

- Kong, D., Xu, L., Li, X., & Li, S. (2013). K-plane-based classification of airborne LiDAR data for accurate building roof measurement. *IEEE Transactions on Instrumentation and Measurement*, 63(5), 1200-1214.
- Wang, C., Ji, M., Wang, J., Wen, W., Li, T., & Sun, Y. (2019a). An improved DBSCAN method for LiDAR data segmentation with automatic Eps estimation. *Sensors*, 19(1), 172.
- Ballard, D. H. (1981). Generalizing the Hough transform to detect arbitrary shapes. *Pattern recognition*, 13(2), 111-122.
- Nguyen, A., & Le, B. (2013). 3D point cloud segmentation: A survey. In *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)* (pp. 225-230). IEEE.
- Malihi, S., Valadan Zoej, M. J., & Hahn, M. (2018). Large-scale accurate reconstruction of buildings employing point clouds generated from UAV imagery. *Remote Sensing*, 10(7), 1148.
- Nurunnabi, A., Belton, D., & West, G. (2012). Robust segmentation in laser scanning 3D point cloud data. In *2012 International Conference on Digital Image Computing Techniques and Applications (DICTA)* (pp. 1-8). IEEE.
- Xu, Y., Yao, W., Hoegner, L., & Stilla, U. (2017). Segmentation of building roofs from airborne LiDAR point clouds using robust voxel-based region growing. *Remote Sensing Letters*, 8(11), 1062-1071.
- Li, L., Yao, J., Tu, J., Liu, X., Li, Y., & Guo, L. (2020). Roof plane segmentation from airborne LiDAR data using hierarchical clustering and boundary relabeling. *Remote Sensing*, 12(9), 1363.
- Shao, J., Zhang, W., Shen, A., Mellado, N., Cai, S., Luo, L., Wang, N., Yan, G., & Zhou, G. (2021). Seed point set-based building roof extraction from airborne LiDAR point clouds using a top-down strategy. *Automation in Construction*, 126, 103660.
- Wang, K., & Frahm, J. M. (2017). Single view parametric building reconstruction from satellite imagery. In *2017 International Conference on 3D Vision (3DV)* (pp. 603-611). IEEE.
- Alidoost, F., Arefi, H., & Tombari, F. (2019). 2D image-to-3D model: Knowledge-based 3D building reconstruction (3DBR) using single aerial images and convolutional neural networks (CNNs). *Remote Sensing*, 11(19), 2219.
- Alidoost, F., Arefi, H., & Hahn, M. (2020). 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)
- Gui, S., & Qin, R. (2021). Automated LoD-2 model reconstruction from very-high-resolution satellite-derived digital surface model and orthophoto. *ISPRS Journal of Photogrammetry and Remote Sensing*, 181, 1-19.
- Wichmann, A., Agoub, A., & Kada, M., 2018. RoofN3D: Deep Learning Training Data for 3D Building Reconstruction. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(2)
- Zhang, W., Li, Z., & Shan, J. (2021). Optimal model fitting for building reconstruction from point clouds. *IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing*, 14, 9636-9650.
- Qi, C. R., Yi, L., Su, H., & Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in neural information processing systems*, 5105-5114.

- Li, L., Song, N., Sun, F., Liu, X., Wang, R., Yao, J., & Cao, S. (2022). Point2Roof: End-to-end 3D building roof modeling from airborne LiDAR point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 193, 17-28.
- Nishida, G., Bousseau, A., & Aliaga, D. G. (2018). Procedural modeling of a building from a single image. In *Computer Graphics Forum* (Vol. 37, No. 2, pp. 415-429).
- Liu, H., Li, W., & Zhu, J. (2022). Translational Symmetry-Aware Facade Parsing for 3-D Building Reconstruction. *IEEE MultiMedia*, 29(4), 38-47.
- 3D Tiles, (2023). <https://www.ogc.org/standard/3dtiles/> (accessed on 28 February 2023)
- glTF, (2023). <https://www.khronos.org/glTF/> (accessed on 1 March 2023)
- Barnes, M., Finch, E.L., Sony Computer Entertainment Inc, (2008). COLLADA – Digital Asset Schema Release 1.5.0
- KML, (2023). <https://www.ogc.org/standard/kml/> (accessed on 1 March 2023)
- Ledoux, H., Arroyo Otori, K., Kumar, K., Dukai, B., Labetski, A., & Vitalis, S. (2019). CityJSON: A compact and easy-to-use encoding of the CityGML data model. *Open Geospatial Data, Software and Standards*, 4(1), 1-12.
- Mao, B., Ban, Y., & Laumert, B. (2020). Dynamic online 3D visualization framework for real-time energy simulation based on 3D tiles. *ISPRS International Journal of Geo-Information*, 9(3), 166.
- Xu, Z., Zhang, L., Li, H., Lin, Y. H., & Yin, S. (2020). Combining IFC and 3D tiles to create 3D visualization for building information modeling. *Automation in Construction*, 109, 102995.
- Zhao, J., Zhu, Q., Du, Z., Feng, T., & Zhang, Y. (2012). Mathematical morphology-based generalization of complex 3D building models incorporating semantic relationships. *ISPRS Journal of Photogrammetry and Remote Sensing*, 68, 95-111.
- Baig, S. U., & Rahman, A. A. (2013). A three-step strategy for generalization of 3D building models based on CityGML specifications. *GeoJournal*, 78, 1013-1020.
- Song, Z., & Li, J. (2018). A dynamic tiles loading and scheduling strategy for massive oblique photogrammetry models. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)* (pp. 648-652). IEEE.
- Kulawiak, M., Kulawiak, M., & Lubniewski, Z. (2019). Integration, processing and dissemination of LiDAR data in a 3D web-GIS. *ISPRS International Journal of Geo-Information*, 8(3), 144.
- Lu, M., Wang, X., Liu, X., Chen, M., Bi, S., Zhang, Y., & Lao, T. (2021). Web-based real-time visualization of large-scale weather radar data using 3D tiles. *Transactions in GIS*, 25(1), 25-43.
- Arthur, D., & Vassilvitskii, S. (2007). K-means++ the advantages of careful seeding. In *Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms* (pp. 1027-1035).
- Dirksen, J. (2015). Learning Three.js—the JavaScript 3D Library for WebGL. Packt Publishing Ltd.
- Armeni, I., Sener, O., Zamir, A. R., Jiang, H., Brilakis, I., Fischer, M., & Savarese, S. (2016). 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 1534-1543).
- De Brabandere, B., Neven, D., & Van Gool, L. (2017). Semantic instance segmentation with a discriminative loss function. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 478-480.

Bibliography

- Comaniciu, D., & Meer, P. (2002). Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on pattern analysis and machine intelligence*, 24(5), 603-619.
- Qin, Z., Yu, F., Liu, C., & Chen, X. (2018). How convolutional neural networks see the world – A survey of convolutional neural network visualization methods. *Mathematical Foundations of Computing*, 1(2): 149.
- Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801-818).
- Hu, Q., Yang, B., Xie, L., Rosa, S., Guo, Y., Wang, Z., Trigoni, N. & Markham, A. (2020). RANet: Efficient semantic segmentation of large-scale point clouds. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 11108-11117).
- Gorelick, N., Hancher, M., Dixon, M., Ilyushchenko, S., Thau, D., & Moore, R. (2017). Google Earth Engine: Planetary-scale geospatial analysis for everyone. *Remote sensing of Environment*, 202, 18-27.
- Schilling, A., Bolling, J., & Nagel, C. (2016). Using glTF for streaming CityGML 3D city models. In *Proceedings of the 21st International Conference on Web3D Technology* (pp. 109-116).
- Huang, J., Stoter, J., Peters, R., & Nan, L. (2022). City3D: Large-scale building reconstruction from airborne LiDAR point clouds. *Remote Sensing*, 14(9), 2254.
- GEORES (2023). <https://www.geoplex.de/geores/>. (Accessed 1 April, 2023).
- Vaaranemi, M., Treib, M., & Westermann, R. (2011). High-quality cartographic roads on high-resolution DEMs, *Journal of WSCG*, 12/2, pp. 41–48.
- Twigg, C. (2003). Catmull-rom splines. *Computer*, 41(6), 4-6.
- Snavely, N., Seitz, S. M., & Szeliski, R. (2008). Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2), 189–210.
- Hart, K. A., & Rimoli, J. J. (2020). Generation of statistically representative microstructures with direct grain geometry control. *Computer Methods in Applied Mechanics and Engineering*, 370, 113242.
- Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2881-2890).
- Neuhold, G., Ollmann, T., Rota Bulò, S., & Kotschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision* (pp. 4990–4999).
- Mapbox (2023). <https://www.mapbox.com/>. (Accessed 1 April, 2023).
- Leaflet (2023). <https://leafletjs.com/>. (Accessed 1 April, 2023).
- Nishida, G., Garcia-Dorado, I., Aliaga, D.G., Benes, B. and Bousseau, A. (2016). Interactive sketching of urban procedural models. *ACM Transactions on Graphics (TOG)*, 35(4), p.130.
- Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L. & Desmaison, A. (2019). Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32: 8026–8037.
- Zhao, L. and Tao, W. (2020). JSNet: Joint instance and semantic segmentation of 3D point clouds. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(7): 12951–12958.

Ren, M. and Zemel, R. S. (2017). End-to-end instance segmentation with recurrent attention. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6656–6664.

Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801–818).

Bibliography

Research publications

This part includes the complete research outcomes that are communicated through the following international publications.

Research publications

PAPER 1

**VGI3D: an interactive and low-cost solution for 3D building modelling
from street-level VGI images**

Chaoquan Zhang^a, Hongchao Fan^a, Gefei Kong^b

^aDepartment of Civil and Environmental Engineering, Norwegian University of Science and
Technology, Trondheim, Norway

^bSchool of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China

This paper is published in *Journal of Geovisualization and Spatial Analysis*, September
2021; 5:1-16

Research publications



VGI3D: an Interactive and Low-Cost Solution for 3D Building Modelling from Street-Level VGI Images

Chaoquan Zhang¹ · Hongchao Fan¹ · Gefei Kong²

Accepted: 10 August 2021
© The Author(s) 2021

Abstract

Applications in smart cities are inseparable from the usage of three-dimensional (3D) building models. However, the cost of generating and constructing 3D building models with semantic information is high both in time and in labour. To solve this problem, we developed a web-based interactive system, VGI3D, with the ambition of becoming a VGI platform to collect 3D building models with semantic information by using the power of crowdsourcing. VGI3D is a platform-independent software program that is composed of a spatially relational database (PostgreSQL/PostGIS) for the storage and management of spatially geometrical data and other software modules, allowing users to import, analyse, reconstruct, visualise, modify and export 3D building models according to the OBJ/CityGML standard. In this paper, we present the VGI3D in detail, focusing on relevant technical implementations, and report the results of limited usability testing aimed at optimising the system and user experience. After limited expert and non-expert participants' testing, we proved the usefulness of VGI3D and its promising value for the 3D modelling community.

Keywords 3D building modelling · Spatial relational database · Python Flask · User interaction · CityGML · VGI images · Low cost

Introduction

As the technology centre in Norway and the first big city to implement and test 5G communication in 2020, the Trondheim municipality is currently gearing up the construction of a smart city. In the coming years, a number of smart city-related applications will be planned. Meanwhile, increasing applications in smart cities necessitate a large number of 3D building models (3DBMs)—just think of solar simulation (Li et al. 2019), virtual tourism (Templin et al. 2020), urban planning (Park et al. 2021), augmented reality (Blut and Blankenbach 2021), path navigation (Liu et al. 2020), disaster management (Haynes et al. 2018), architectural design (Li et al. 2017), etc. Therefore, it is urgent to generate 3D building models with semantic

information. For this purpose, a joint laboratory called Nordig Lab was established as a collaboration between the Norwegian University of Science and Technology (NTNU) and Trondheim municipality. The main focus of Nordig Lab is digitalisation, which includes generating large scale 3D building models with rich semantic information as well.

3D building models are generally defined by five different levels of details (LoDs) according to the CityGML2.0 standard (Gröger and Plümer 2012). LoD0 is a representation of the ground boundary (or footprint) of a building. LoD1 is a cuboid obtained by extruding the LoD0 model. LoD2 consists of a simplified roof shape and multiple semantic classes of a building (e.g. wall, roof). In comparison with LoD2, LoD3 is often considered a more architecturally detailed model in that it contains windows, doors and other rich semantic information. LoD4 is more complicated and completes an LoD3 model by including indoor components. In general, most of the 3D building model-related applications are not just satisfied with the simple LoD1 models and instead want models to have at least a roof shape (i.e., LoD2 or better), particularly in smart cities (Monteiro et al. 2018). They prefer photorealistic 3D building models with windows, doors or balconies in LoD3 or higher, such as the

✉ Chaoquan Zhang
chaoquan.zhang@ntnu.no

¹ Department of Civil and Environmental Engineering,
Norwegian University of Science and Technology,
Trondheim, Norway

² School of Remote Sensing and Information Engineering,
Wuhan University, Wuhan, China

estimation of solar irradiation of buildings (Machete et al. 2018) and urban CO₂ emission simulation and measurement (Eicker et al. 2018). Even some attempts (Tang et al. 2020; Biljecki et al. 2016) have been tried to propose the less generic and more application-driven specification for 3D building models, since standard LoDs specification lacks flexibility and degree of freedom to some extent.

To meet the requirements of the abovementioned applications that require 3D building models with different LoDs, a few cities like New York and Berlin have created and freely released 3D city models based on the CityGML standard over the last decade. These 3D city models not only contain buildings but also include streets, trees, bridges and even terrain. They are defined and presented as the respective real-world objects with respect to their geometrical, topological, sematic and appearance properties (Stadler et al. 2009). Nevertheless, most of these 3D city building models are constructed in LoD1 or LoD2, and large-scale LoD3 models with semantic information are hardly available. Hence, that is the main motivation of this paper to generate 3D building models in LoD3 with semantic information.

3D building models are mainly generated from 3D point clouds and images. 3D point cloud data can be classified into three categories: airborne, terrestrial and mobile laser scanning data. While 3D modelling methods based on point clouds have been widely studied (Sun and Salvaggio 2013; Xiong et al. 2014; Wang et al. 2018), the quality of reconstructed 3D building models varies, owing to differences in point densities, scanning patterns and geometric characteristics (Yang and Dong 2013). In other words, the kind of data used depends on the specific needs of the application. For instance, we would like to create 3D building models in LoD3, and, in this case, airborne point cloud data are obviously inappropriate and cannot provide rich façade information due to the sparse point density of facades (Wu et al. 2017). However, terrestrial point clouds can. In the methods based on terrestrial point clouds, classic approaches usually learn attribute topology and grammar rules from precise descriptions and noisy observations to create high-resolution 3D building models (Becker 2009; Dehbi and Plümer 2011). Even though their reconstructed models are stable and complete, their algorithms are limited by the computational performance and complexity of the scene as well as types of architectural styles.

With the deepening of research and the updating and iteration of algorithms, great success has been achieved in recent years. Dehbi et al. (Dehbi et al. 2017) further optimised their previous work and proposed a statistical method, which is capable of successfully coping with complexity (a varying number of objects), uncertainty and unobservability in real-world problems. This idea also inspired other researchers and was applied to the 3D modelling of indoor environments (Tran et al. 2019). In addition, the great

method presented by Yu et al. (Yu et al. 2017) not only ensured a very high modelling precision but also stamped out the weaknesses of restricted architectural styles. However, to obtain impressive results, the above approaches have to rely on quite expensive point cloud acquisition devices not suitable for mass markets, and modelling procedures are time-consuming and laborious as well.

Another research branch for 3D building modelling is to use images. The classic multiview stereo (Furukawa and Ponce 2009) algorithm often needs to take a set of calibrated images that are surrounding the object. While it emphasises that it is an automatic method and does not require any initialisation, it often suffers from painful secondary editing to fix the dense 3D models they produce, contrary to the original intention of automation. Moreover, images cannot provide rich geometric and topological information about buildings. Hence, completely automatic modelling is known to omit user interaction, and it is generally accepted that such does not produce satisfying results in case of erroneous or partially missing data (Musialski et al. 2013). To address these issues, Wolberg and Zokai (Wolberg and Zokai 2018) designed a photo-centric 3D modelling system—Photo-Sketch—that not only benefited automatic camera pose recovery and point cloud generation by using the structure from motion algorithm (SfM) (Ullman 1979) but also introduced user interaction to overcome geometry incompleteness. However, SfM requires many images with overlap and must know the internal parameters of a camera for camera calibration. In their experiment, it took them around 20 min to reconstruct only one building model, which reflected the inefficient computation and great reliance upon the number of images. Getting benefit from advances in deep learning, Liu et al. (Liu et al. 2021) proposed a translational symmetry-aware façade parsing approach for 3D building modelling and achieved extremely photorealistic 3D models. They fused semantic segmentation and instance detection to parse the façade elements into semantic grammars from just one image, and then reconstructed final 3D models by using procedural modelling. Despite gaining highly detailed 3D building models, the system asked users to repeatedly interact and gradually add floors, ledges, roof, windows, doors and balconies, respectively. However, they did not report the whole modelling time and hence we cannot know the efficiency of this approach and cannot compare it to other methods. In addition, to be honest, it is not friendly or easy for non-expert users to use because they may find the interaction process cumbersome and may lose patience.

Furthermore, most of the existing 3D building modelling systems appear predominantly desktop or lab-based, such as Kim and Han's work (Kim and Han 2018), and might need users to do some extra environmental configuration. That will undoubtedly further increase the difficulty of promotion to ordinary users and the mass market

appeal, even though these 3D building modelling systems are outstanding products. On the other hand, many smart city applications now urgently demand 3D building models, but the existing 3D modelling costs are not low in terms of modelling time, labour or data collection. No matter whether from the economic or efficiency point of view, using the existing solutions to simultaneously meet the above points is difficult. Therefore, we are motivated to design a web-based interactive 3D building modelling system with lower costs, faster speed and more intelligent processes.

The idea of volunteer geographic information (VGI) has received widespread attention since its inception. VGI employs tools to create, assemble and disseminate geographic data provided voluntarily by individuals (Sangiambut and Sieber 2016). Nowadays, smart devices with cameras are becoming increasingly powerful and cheaper, particularly for mobile phones. People have been more willing to share their daily life through images than ever. Thus, everyone can become a sensor to contribute image data by using mobile phones, digital cameras and even a GoPro. Although it is hard to evaluate and ensure the quality of VGI images, the convenience of VGI image data acquisition and richness of image data quantity can make up for the shortage of image quality to some extent, which resolves the problem of low-cost data acquisition.

In terms of modelling speed and intelligence, a deep learning technique would be an excellent assistant. In our work, users only need to simply and directly outline the façade of a building upon the image; then, our system will automatically detect façade elements (e.g. windows, doors, balconies) and adjust their bounding boxes. Finally, 2D locations of building elements are going to be transformed into a 3D coordinate system and, meanwhile, show a 3D model to users in real time. Due to the simplicity of interaction, it is easier to promote the system to ordinary users and it will certainly be beneficial to the spread and development of urban 3D modelling.

The novelty of our work is in combining interactive and low-cost 3D building modelling, interactive model updating, real-time 3D model visualisation and integration with a spatially relational database (PostgreSQL/PostGIS) and Python Flask (Flask 2021) web framework. In summary, we intend to verify among both expert and non-expert users that our system is useful and has potential value for the 3D building modelling community.

The rest of this paper is organised as follows: the ‘Methodology’ section presents our methodology, which contains the system architecture and design, algorithm, testing and evaluation and usability testing. Then, we show the implementation with the results of testing in the ‘Software Implementation and Testing’ section. A discussion is presented in the ‘Discussion’ section, and the ‘Conclusion and Future

Work’ section provides the conclusions as well as future work.

Methodology

In this paper, our system has the following functionality: (1) simple user interaction, including roof type and façade orientation selection and façade drawing; (2) geographically matching façade with an edge of the footprint on a 2D map as a reference to get the real ratio/scale; (3) façade elements detection, correction and inference of locations of façade elements; (4) 3D building modelling from 2D locations; (5) interactively editing/updating incorrect parts of reconstructed models; (6) cloud server (AWS EC2) capability for system deployment and PostgreSQL for data storage/retrieval; (7) integration with a Python Flask (Flask 2021) web framework, Three.js (Dirksen 2015) as 3D real-time visualisation and WebGL.

The presented work is based on our two earlier works (Kong and Fan 2020; Fan et al. 2021). The first one (Kong and Fan 2020) proposed a deep learning method to detect façade elements from low-quality street-level images. The second one (Fan et al. 2021) introduced the first version of VGI3D from the perspective of geographic information system (GIS). Now, this paper will concentrate on changes and new features and, meanwhile, further explain our system from the perspective of software engineering.

Designed User Experience

A summary of the designed user experience is presented here to help better understand our system, VGI3D. A typical use case would be as follows: (1) users upload no more than two images that belong to the same building but different façade directions to the building; (2) they select the façade direction for each image and its corresponding footprint edge from the map; (3) they pick a roof type, draw the façade boundary for each image and then click the ‘Save’ button; (4) they repeat this process until no images need to be handled, and then the reconstructed 3D building model with different elements colour would be shown in real time; and (5) additionally, if certain façade element is wrongly reconstructed, they click the ‘Update’ button and start to update the model in an interactive fashion as well by deleting the incorrect element from the 3D model, drawing the element boundary on the façade image, selecting a right element type for it and then click the ‘Save’ button to finish the modification; (6) lastly, the reconstructed 3D building

model can be exported in CityGML/OBJ format by clicking the ‘Download’ button.

System Architecture, Design and Data Flow

An abstract overview of system architecture is shown in Fig. 1, which adopts a classic software design pattern, MVC (model-view-controllers). ‘M’ means model and corresponds to the data layer. All interactions with the database are done here. ‘V’ refers to the display or interaction layer in which we can operate the system through the graphic user interface. ‘C’, the controller, refers to the business layer. Most key modules are placed there and handle the user’s actions. Generally, the controller interprets the mouse and keyboard inputs from the user, then launches a request to the database to get the data, process the data and then update the view.

The VGI3D is currently deployed on a cloud server (AWS EC2) with an Ubuntu version 16.04 operating system. In the data layer, we set up a relational database, PostgreSQL, with PostGIS plugin, for the purpose of spatial geometry search. The data layer mainly stores all the building footprints with geographic locations in Norway (would be used in edge selection in interaction layer), all the images that users uploaded and reconstructed 3D building models. The business layer is the core part of our

system. The concise pipeline is, first, detect the façade elements by using a deep learning model, YOLOv3 (Redmon and Farhadi 2018), such as windows, doors and balconies. Second, we correct the locations of façade, façade elements and roof to unify them under the same coordinate system. Third, we convert 2D elements into 3D and then obtain the final 3D model. A detailed sequence diagram of each step and data flow between steps can be found in our previous work (Fan et al. 2021). User interaction takes place in the interaction layer, which involves roof type and façade orientation selection, façade drawing and geographically matching the façade with an edge of a footprint on the 2D map as a reference to get the real ratio/scale. The last important item would make sure that the reconstructed 3D building model has a real scale, real orientation and real geographic coordinates.

Display layers include HTML5, cascading style sheets (CSS) and Bootstrap for rendering the user graphics, and Three.js is used to provide 3D-rendering support. jQuery is also employed to handle user interaction, mainly listening to response events. An HTTP connection, Ajax and Jinja2 template engine are intermittently needed to communicate with the frontend and backend. Data flow between the frontend and backend is represented in the format of JSON.

Fig. 1 Abstract overview of system architecture

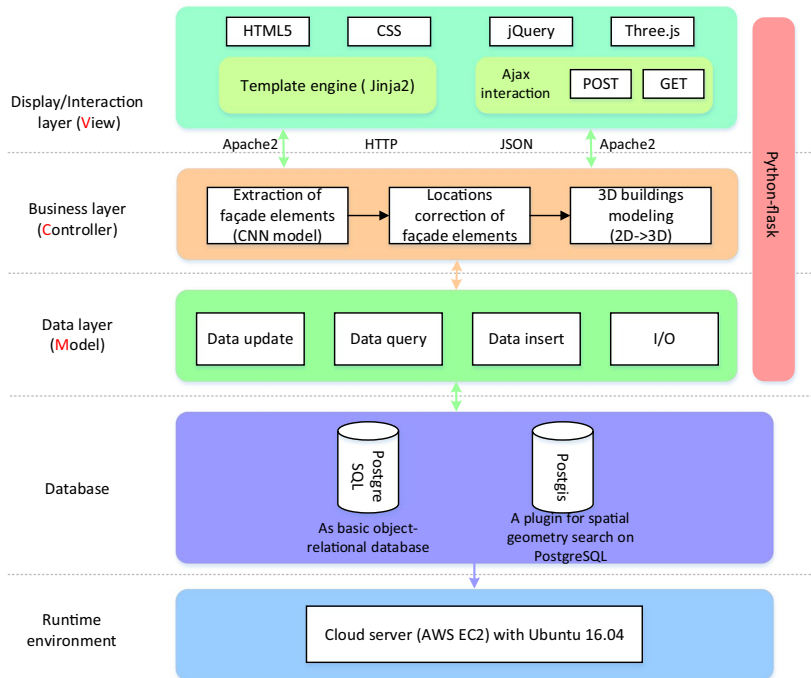


Fig. 2 System activities flow and communication between the database and Flask web service

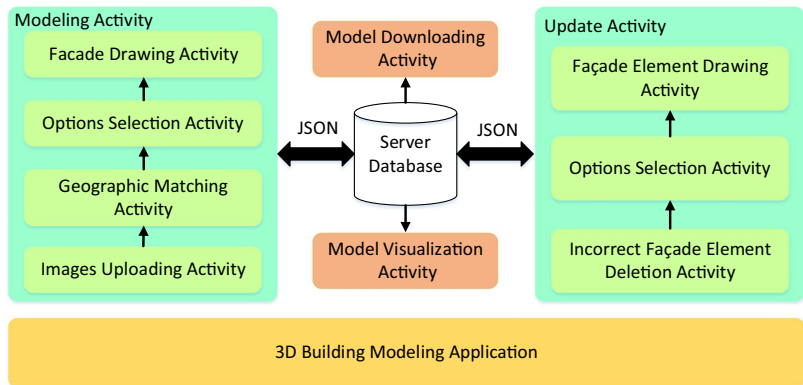


Figure 2 shows our core system design with the 3D building modelling application at the bottom. Activity flow follows the direction indicated, but the order within ‘modelling activity’ can be interchanged (i.e. no fixed order). The JSON data flow helps with communication between the database and web service and is capable of accessing via ‘modelling activity’ and ‘update activity’ as well.

Algorithms

Since the core concepts of our 3D modelling method are low cost and speed, one of the key algorithms is to automatically and accurately detect façade elements in a short time. As we know, building facades in street-level images generally have complex scenes, such as distortion, blur or bad illumination. The existing approaches demand users to manually draw façade elements, which is labour-intensive and time-consuming. Hence, we adopt a convolutional neural network (CNN) model, YOLOv3, to help us automatically extract façade elements. We follow the steps suggested by Kong and Fan (Kong and Fan 2020) and train YOLOv3 on the FaçadeWHU dataset (Kong and Fan 2020), which includes 900 street-level images (50 images in Trondheim, Norway, and 850 images in Paris, France). Then, we can obtain the location of every façade element on the image coordinate system. Since street-level images usually have more or less perspective distortion, we need to correct the extracted façade elements and then project them to a 3D space. Besides, for visualisation and modelling completeness purposes, we also need to construct the building’s roof according to roof type that was selected by the user before. The top edge of the façade is regarded as the bottom of the roof. The height of the roof can be estimated according to the ratio of façade length to width. Algorithm details can be found in the ‘Methodology’ section of our earlier work (Fan et al. 2021).

In some cases, however, this CNN model may sacrifice accuracy or completeness while ensuring speed. Some façade elements might be incorrectly detected because of their complicated scenes or low-quality images. Therefore, interactively updating 3D models is needed and necessary. For this purpose, VGI3D allows users to firstly select and delete the incorrectly reconstructed façade element based on the original model. Next, they can simply outline the wrongly reconstructed façade element upon an image and select a corresponding element type for it. Then, an updating algorithm will automatically merge the new façade element into the original 3D model and adjust the locations of elements to make the overall model look coordinated and harmonious. After that, the new modified model will be shown to users in real time. This algorithm worked well and met the requirements of fast speed, ease of use and low cost.

Last but not least, to apply 3D building models into various map-based applications, it is essential to give 3D building models real geographical coordinates instead of relative coordinates. We asked users to select a footprint of the building that they want to construct on the map (here, it was OpenStreetMap) before outlining the façade, then further select the edge from the footprint, which corresponds with the façade direction being drawn. In other words, users should pick up an edge for each façade within the image. Our modelling algorithm will be able to calculate the ratio (façade height/façade width) of real coordinates and assign real geographical coordinates for each vertex of the geometric 3D model.

Testing and Evaluation

In addition to usual unit tests during development, functional testing was also carried out to make sure the system worked as planned—for example, by raising technical issues. The

functional testing is mainly conducted around the following aspects:

- (1) Image-uploading activity, including uploading more than two images and continuously uploading images.
- (2) Geographic matching activity, including map layout/location, highlighting selected footprint and edge with different colours.
- (3) Options activity, including operating under different conditions, e.g., different combinations of façade direction and roof type.
- (4) Façade/façade element drawing activity, including drawing operation and deleting wrong drawing operations.
- (5) Incorrect façade element deletion activity, including selecting and highlighting incorrect elements and deleting incorrectly reconstructed elements.
- (6) Visualising and exporting the 3D building model.

The system was evaluated by conducting a user study with 30 non-expert/expert participants. The Participants were (a) shown a tutorial video and a slide presentation about how to operate the system, (b) were given some time to try the system by themselves, and finally (c) were asked to fill in a user feedback form. A copy of the user feedback form is available in Appendix A. Most questions of this form are for participants to rank a specific aspect of the system from one (least) to five (most) points. The raw data of their answers can be found in Appendix B. Please note that Nos. 1–15 are expert participants and Nos. 16–30 are non-expert participants. The grouping criteria is roughly based on: the research fields are strongly or relatively related to the 3D model as the expert group; the research fields are weakly or slightly related to the 3D model as the non-expert group.

Software Implementation and Testing

The VGI3D has been implemented and can be accessed at: <https://18.210.26.42:5002/facade/>. We also invited expert and non-expert users to test the software. In this section, software implementation will be demonstrated.

Implementation

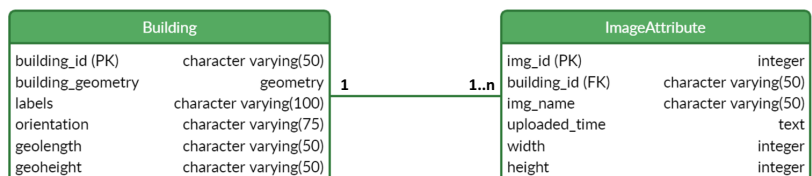
A driving principle behind our design is to enable the system's interaction as simple as possible, which would be friendly to non-expert users and would be more beneficially promoted to the mass market. For this purpose, we borrowed the drawing idea from the Mapbox (Mapbox 2021) SDK (software development kit) to achieve the façade polygons on images. A building footprint database with PostGIS was created in advance to supply geographic information of footprints and buildings' height. Then, Leaflet (Leaflet 2021) was utilized to select edges of the footprints from an embedded map and wrapped and transmitted them to the backend for calculation via GeoJSON. Relying on these geographical information, we could calculate the real ratio of building facades, which would further ensure reconstructed building models have real geographic coordinates.

Another driving principle behind our design is modelling buildings anywhere. This tries to ensure the modelling system is viable in a general environment and, hence, users will not be constrained to a limited number of facade styles. To realise this goal, we adopted a CNN model, YOLOv3, for helping to automatically detect façade elements (i.e. window, door, balcony) within one second. The utilization of CNN not only satisfied the modelling anywhere but also saved considerable time for the entire workflow because the step of façade elements extraction is the most time-consuming according to previous work (Nishida et al. 2016).

After the interaction with all images, the created data are transferred to the backend in JSON format to reconstruct 3D building models. The content of data includes edges' geographic coordinates, locations of façade boundaries, locations of all the façade elements, building's height, uploaded images, roof type and façade orientations. In addition, all these data will be stored in two separate tables of database (as shown in Fig. 3). One is used for storing building's geometry and the other is for image attributes. They can communicate with each other via a foreign key, *building_id*. The reconstruction part of models has been explained in detail in our previous work (Fan et al. 2021), so we are not planning to describe the reconstruction part here once again.

Although our objective is capable of automatically and accurately reconstructing building models with arbitrary

Fig. 3 Table structure of the database



façade styles, the actual situation always deviates from the ideal situation. For example, there are tons of styles of doors and windows in the world and our CNN model cannot correctly detect all of them due to an inadequate training dataset. Sometimes, incorrect or missing detection can also be caused by low-quality images, such as distortion, blur, illumination or reflection. Therefore, update activity was implemented to interactively correct the wrong façade elements. In our approach, users can (a) select and highlight the element that they want to modify; (b) remove it by clicking the ‘Delete’ button, which is located at the upper right corner of the 3D viewer; (c) draw the corresponding element’s boundary that we deleted before; (d) select the

element type and façade orientation as extra but essential attributes; (e) click the ‘Update’ button to wrap the data into a JSON file and transmit them to the backend; (f) the system would simultaneously execute the update function and update the database; (g) lastly, the modified 3D model would be returned to the frontend. Figures 4 and 5 illustrate the whole modelling process. The benefit of this approach was to make the modelling process more flexible and convenient and make the results more accurate to some degree.

Once modelling has been completed, exporting 3D building models is possible and available by clicking the ‘Download’ button. We currently support two types of 3D formats, OBJ and CityGML.

Fig. 4 (a) Main modelling activity; (b) geographic matching activity; and (c) reconstructed 3D building model visualisation

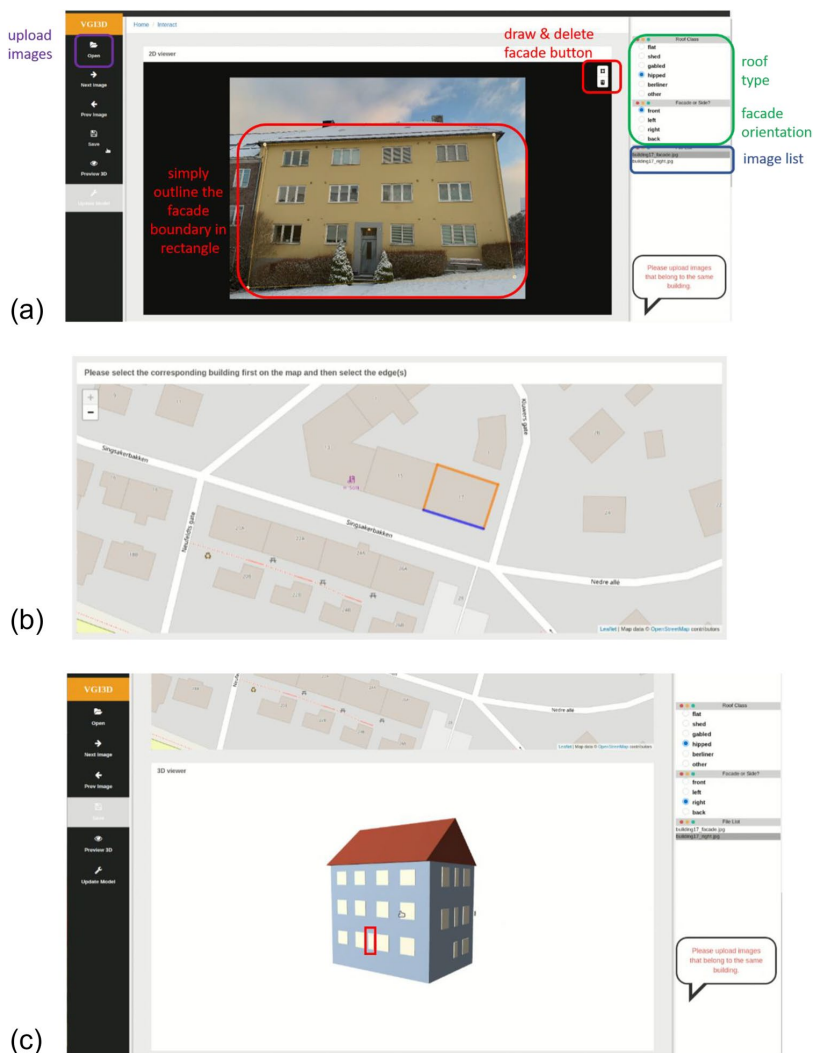
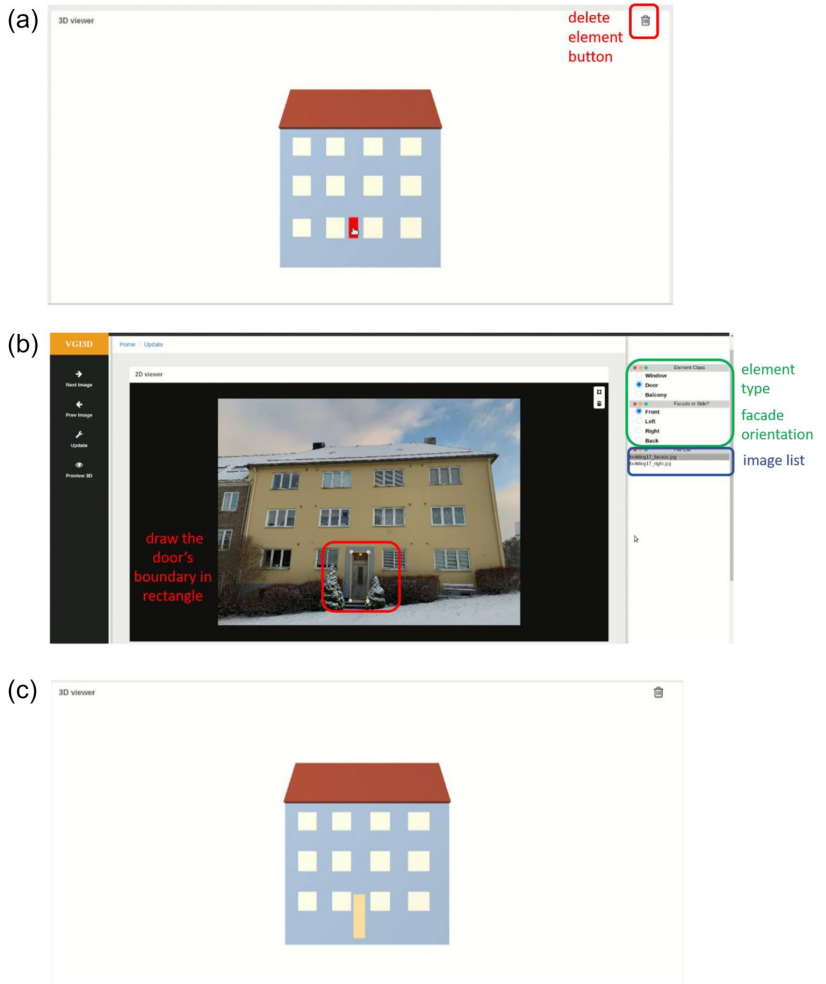


Fig. 5 (a) Incorrect façade element deletion activity; (b) model updating activity; and (c) new 3D building model after updating



Software Functional Testing

The system was opened on a laptop in browser mode and the main modelling page appeared. First, image uploading activity was activated, in which preprepared images that belong to the same building were uploaded by clicking the fold icon of the left sidebar, as shown in a purple rectangle of Fig. 4a. Then, one image in the image sequence was displayed in the 2D viewer of the workspace. Second, we searched and chose the footprint of this building (i.e. highlighted in orange) from the map, then further selected the corresponding edge (i.e. highlighted in blue) of the façade that the participant was handling, from the map to complete the geographic matching activity (as shown in Fig. 4b). Third, we respectively selected the roof type and façade orientation for the current processing image from the right sidebar to finish the options

selection activity (see green rectangle in Fig. 4a). Fourth, we began to draw the façade boundary on the image, which generally is a rectangle, to complete the façade drawing activity (see red rectangle in Fig. 4a), then clicked the ‘Save’ button to transfer the data to the backend and save them into the database. Fifth, we switched to the next image by clicking the ‘Next Image’ button, then repeated the second to fourth steps until all images were done. Once the functional testing of modelling activity had been finished, the reconstructed 3D building model was visualised in a 3D viewer workspace as expected (see Fig. 4c). Additionally, if participants did not select any item of a certain kind, such as roof type, edges or façade orientation, a dialogue box would pop up to remind participants not to omit certain items. Or, another case was that, if participants would like to reupload images and discard all the operations that they had done previously, the

system would also pop up a dialogue box to ask whether they wanted to discard them or not. These eventualities were all considered during the development stage and were worked as planned.

Sometimes, the incorrect façade elements (e.g. the door outlined in red in Fig. 4c) are inevitable after reconstruction due to the reasons we mentioned above. Such will lead us to test update activity by clicking the ‘Update Model’ button in the left sidebar. Meanwhile, we will advance to a new page, ‘update page’. In the 3D viewer workspace, a deletion icon was available, which was used to delete the selected wrong façade element with highlighted colour (see Fig. 5a). That was the first step under update activity, i.e., incorrect façade element deletion activity in Fig. 2. Second, we found the right image that we would like to draw upon via clicking the ‘Next Image’ or ‘Prev Image’ button, then manually drew the boundary of the façade element on the image whose location corresponded to the element we deleted just now. In this way, we completed the testing of the façade element drawing activity (see red rectangle in Fig. 5b). Third, we selected the appropriate element class and façade orientation from the right sidebar (see green rectangle in Fig. 5b). Fourth, we clicked the ‘Update’ button to transmit the data to the backend and executed the model updating functionality. Meanwhile, we updated the database (i.e., column *building_geometry* and *labels* of table *Building*) accordingly and the updated result would be displayed in the 3D viewer in real time (see Fig. 5c). We repeated this process until no other incorrect façade elements needed to be updated. Please note that only one façade element can be updated at a time. Finally, exporting 3D models was supported according to the OBJ/CityGML standard.

Until now, we have managed to test all functionalities and everything worked well, with the only possible challenge being the quality of images. Images of a low quality, particularly for façade parts, can affect the completeness or stability of façade elements extraction, but it is typically

not severe enough to kill performance. The system most likely performed well thanks to the fact that modelling has a powerful CNN detection model as a reliable cornerstone, simple but considerate interaction logic and novel interactive updates. The system was also tested extensively in practice, which tremendously reduced the potential problems during actual use. Furthermore, since VGI3D was developed based on HTML5, it is possible to open it in the browser on any device. Hence, we tested the system on a mobile phone, but, unfortunately, the user experience was not good at all because the screen of a mobile phone was too small and the layout of widgets had changed a lot and made the system look very ugly.

Software Evaluation

To evaluate the usability of VGI3D, we conducted a small scale of usability testing at the Norwegian University of Science and Technology, Norway, and Wuhan University, China. 30 expert/non-expert participants (aged 25–43 years old, including 7 women and 23 men) were invited to experience and test the VGI3D. Six of them are expert participants and the others are non-expert participants. Their research fields include 3D city modelling, 3D visualization, photogrammetry, urban planning, BIM for asset assessment/management, spatial analysis, computer science, GPS, railway design and facility management. Before the formal testing, they would be shown a tutorial video and a slide presentation about how to operate the system and be given some time to get familiar with the system. After that, our testing kicked off. Each participant was randomly assigned a preprepared folder where storing building images, and the participant could use them for testing. Once the testing was complete, they were asked to fill in a user feedback form. This form included questions about, for instance, their demographic information, previous experience (‘How experienced are you with 3D modelling, e.g., Sketchup?’), satisfaction (‘The

Table 1 Summarized suggestions/comments from participants during the usability testing

‘The user interface is clear and concise, and easy to understand overall.’
‘This button is slightly small; making it bigger would be better.’
‘Overall, the system is good and promising and is beneficial to my research.’
‘From the BIM point of view, the reconstructed 3D model is not as accurate and detailed as BIM model.’
‘The tips are not obvious. Had better move them to a more obvious place.’
‘Downloading model spent a little longer time in my case.’
‘The modelling time is beyond my expectation.’
‘Deleting incorrect façade elements sometimes would face bugs.’
‘The reconstructed model is cool and 3D viewing is smooth.’
‘Include building height to get proper element ratio.’
‘Some reconstructed windows are uneven in size.’
‘Hope to improve the system so that it can reconstruct buildings with arbitrary footprints.’
‘Cannot find a specific location name by searching on the map. Hope to fix this problem.’

interaction operation is easy to understand'), understanding of the system, any subjective comments/suggestions, etc. Their comments/suggestions have been summarized and listed in Table 1, as shown below.

After the usability testing, we fortunately received some positive feedbacks from participants: 93.3% (28/30) of the participants reported that the user interface was 'clear' and 'easy' to understand. In terms of 3D modelling, two thirds of participants (20/30; 66.7%) could manage to reconstruct 3D models. Most of the participants who could not successfully reconstruct the 3D models either were non-experts or were inexperienced in 3D modelling. Participants largely thought our system could reconstruct buildings at a rapid speed, and the plausibility of modelling results as well as 3D viewing capability exceeded their expectations. Most importantly, some participants said VGI3D was very beneficial to their own researches. Meanwhile, we also received negative feedbacks from participants. For instance, one of them reported

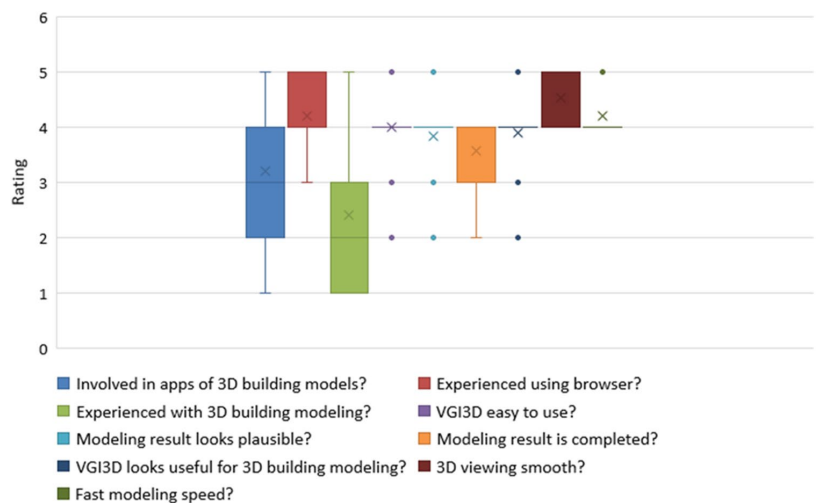
that the tips were not obvious and were easy to be ignored. Three of them found the same bug when trying to delete the wrong façade elements (i.e. failed to delete). Since BIM participants have been used to utilizing high-detailed 3D models for research, they thought our models were not very photorealistic.

Apart from the qualitative comments, we also performed the quantitative evaluation and analysis based on user-feedback-form data from the statistical point of view. The raw data in Appendix B has been statistically summarised in Table 2. Outliers are identified as single points in box plots as shown in Fig. 6. Besides observing the spread and centrality of data, we also proposed related questions by judging how some responses of a participant were associated with others. Since our sample size of participants was not too big, its use would lead to a sparse scatter plot. Sometimes it could be nonlinear in appearance but usually included bound data internally (see Fig. 7). Therefore, to identify associations between participants'

Table 2 Statistics about raw data collected from participants. Mean(μ), standard deviation(σ), lower quartile(Q1), middle quartile(Q2), upper quartile(Q3), interquartile range (IQR). 3DBMs: 3D Building Models; 3DBMing: 3D Building Modelling; 3DMing: 3D Modelling

	Involved in 3BMs apps?	Experienced using browser?	Experienced with 3DMing?	VGI3D easy to use?	Modeling result looks plausible?	Modeling result is completed?	VGI3D looks useful for 3BMinng?	3D viewing smooth?	Fast modeling speed?
μ	3.20	4.20	2.40	4.00	3.83	3.57	3.90	4.53	4.20
σ	1.30	0.60	1.33	0.52	0.58	0.56	0.60	0.50	0.40
Q1	2.00	4.00	1.00	4.00	4.00	3.00	4.00	4.00	4.00
Q2	3.00	4.00	2.00	4.00	4.00	4.00	4.00	5.00	4.00
Q3	4.00	5.00	3.00	4.00	4.00	4.00	4.00	5.00	4.00
IQR	2.0	1.0	2.0	0	0	1.0	0	1.0	0
Skew-ness	-0.19	-0.11	0.51	-1.45	-0.98	-0.84	-0.90	-0.13	1.50

Fig. 6 Box plot of data for illustrating skewness



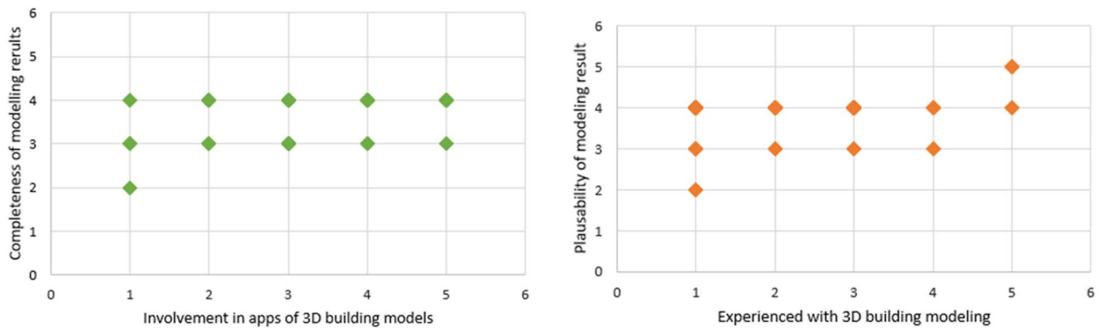


Fig. 7 Examples of sparse scatter plots with nonlinearity in appearance but including bound data points internally

responses data, we computed Spearman's rank correlation coefficient, which is capable of handling linear, nonlinear and skewed relationships. In our case, we had to calculate the Spearman's coefficient with full covariance rather than an approximate formula because of the presence of bound data and, thus, duplicate ranks. Table 4 compares the correlation coefficients between all possible question pairs.

Discussion

Data Analysis

The answers to questions in the user feedback form usually have skewness. We can find their skewed degrees (i.e. magnitude) and nature (i.e. positive or negative skewness) from Table 2. The first column of Fig. 6 shows a wide range of knowing about applications of 3D building models and with a negative skewness of the data. This is what we expected because our participants are from different research fields and the number of participants is limited. We cannot guarantee that every participant is quite involved in applications related to 3D building modelling.

Almost all participants were able to use the browser skillfully with a mean rating of 4.20, slightly negative skewness and no outliers. Most of the participants (24/30; 80%) were not very familiar with 3D building modelling, as seen by an apparent positive skewness and mean rating of 2.40, which is interesting when we compare it with the mean value of 3.20 recorded for the first column (involvement in apps of 3D building models), which has opposite skewness. This might indicate the participants currently do not often use 3D building modelling tools, not to mention the interactive 3D modelling system. To rule out the impact of non-expert participants (Nos. 16–30) on the results, we further calculated the mean value of expert group (Nos. 1–15) on question 'how experienced with 3D modelling?' and obtained the

mean rating of 3.47. This degree is between 'moderately' and 'very' and further verified our guess is correct. This could be interpreted as the novelty of our interactive VGI3D to the 3D building modelling community. almost all participants (28/30; 93.3%; rank over 4) thought our system was easy to use, with a mean rating of 4.00 and negative skewness. Furthermore, most participants (24/30; 80%) reported the modelling results looked plausible by a mean rating of 3.83 and a lower quartile (Q1) value of 4.0, reflecting the fact that most participants highly appraised to VGI3D. Both the modelling result completeness and system usefulness to the 3D building modelling field were considered positive, with mean ratings of 3.57 and 3.90, respectively.

In terms of system usefulness to 3D building modelling, we not only obtained the highest rating of 5 three times but also got the low rating of 2 once. As for the low rating, the participant had no 3D modelling experience and his research interest (railway design) was quite different from the 3D building modelling. As a result, he did not understand the system well; hence, he ended up failing to reconstruct the building model. The ability of understanding varies from person to person and we cannot expect everyone to be able to use our system smoothly; still, this special case encourages us to better optimise the system in the future.

In addition, to further analyse the assessment of different types of participants towards the system, we grouped them by their research fields and then compared the ratings between different groups on each question. Due to space limitations, only 'mean (μ)' is displayed here as shown in Table 3. The remaining statistical indicators (standard deviation and skewness) can be found in Appendix C. From Table 3, we can discover that all the groups gave the highly positive assessment (rank over 4) in terms of smooth 3D viewing capability and fast modelling speed. Together with the plausibility of modelling results and VGI3D usefulness, there is an interesting finding over them. The research fields (BIM and facility management) that need high-detailed 3D

Table 3 Mean value (μ) comparisons between different types of practitioners regarding each question in user feedback form

	Involved in 3BMs apps?	Experienced using browser?	Experienced with 3DMing?	VGI3D easy to use?	Modeling result looks plausible?	Modeling result is completed?	VGI3D looks useful for 3BMing?	3D viewing smooth?	Fast modeling speed?
3D city modelling	4.75	4.25	4.75	4.50	4.50	3.75	4.50	5.00	5.00
3D visualization	4.50	4.50	3.00	4.50	4.00	4.00	4.50	5.00	4.00
Photo-grammetry	4.30	3.67	3.33	4.00	4.00	4.00	4.00	4.67	4.33
Urban planning	4.00	3.50	3.00	4.00	4.00	3.50	4.00	4.50	4.00
BIM for asset assessment / management	3.75	4.50	2.75	4.00	3.50	3.25	3.75	4.50	4.25
Spatial analysis	2.67	4.33	1.00	4.00	4.00	3.67	4.00	4.33	4.00
Computer science	1.80	4.40	1.40	3.80	3.80	3.60	3.80	4.40	4.00
GPS	1.50	4.00	1.00	4.00	4.00	3.50	4.00	4.50	4.00
Railway design	1.50	4.50	1.00	3.00	2.50	2.50	2.50	4.00	4.00
Facility management	3.00	4.00	2.00	4.00	3.67	3.67	3.67	4.33	4.00

models tend to give the moderate evaluation, which might be because these two groups have got used to utilizing the realistic models and think our reconstructed models are not as realistic as the actual buildings, sometimes even worse. Therefore, they do not think our models can be applied to their researches. Additionally, strictly speaking, only two groups (3D city modelling and photogrammetry) have the 3D modelling experience with the mean rating of 4.75 and 3.33, respectively. However, other groups except railway design group still gave us the positive assessment on ease of use, plausibility, completeness, VGI3D usefulness, etc., which indeed reflect that overall our system is great.

Correlation Analysis

Table 4 presents the correlation between all questions based on Spearman's rank correlation coefficient. All correlations were positive except those that were associated with experience using browsers. In our testing, all participants have rich experience in using browsers. These negative correlations could be interpreted as whether or not having experience with using browsers has no direct relationship with other modelling-related responses. That makes sense if we connect with the practice. Nowadays, well-educated people generally have rich experience in using browsers, but they do not necessarily understand 3D building modelling. From Figs. 6 and

7, we can discern that Spearman's coefficient is appropriate in possible nonlinear data and skewed data.

First, we observed whether or not involved in 3D building model apps or experience with browsers or 3D building modelling associated with opinions regarding whether VGI3D was easy to use/understand, modelling results looked plausible and completed, and if VGI3D was indeed useful for the 3D building modelling community. Our findings in Table 4 demonstrate weak positive correlations between involvement in 3D building model apps and the completeness of modelling results, but with nearly 100% confidence and moderate positive correlations with ease of use, modelling results plausibility as well as VGI3D usefulness. However, this weak correlation does not mean that it lacks support from participants as illustrated in Fig. 7(left). Figure 7 (left) seems to show an upward trend; however, we still require more statistical data to verify our guess because the current sample size is not big enough. Another interesting finding was that experience with 3D building modelling showed weak positive correlations with the plausibility and completeness of modelling results as well as smooth 3D viewing capability. This finding was contrary to our intuition. After carefully analysing the raw data, the reason for this may be that non-expert participants have little experience in 3D modelling, which reduced the overall correlations among responses. However, the relationship between experience with 3D building modelling and the plausibility

Table 4 Correlation table showing Spearman's rho and significance values for parameters

	Involved in 3BMs apps?	Experienced using browser?	Experienced with 3DMing?	VGI3D easy to use?	Modeling result looks plausible?	Modeling result is completed?	VGI3D looks useful for 3BMing?	3D viewing smooth?	Fast modeling speed?
Involved in 3BMs apps?	1								
Experienced using browser?	-0.22 (p=0.25)	1							
Experienced with 3DMing?	0.76 (p=0.00)	-0.10 (p=0.60)	1						
VGI3D easy to use?	0.54 (p=0.00)	-0.17 (p=0.36)	0.50 (p=0.00)	1					
Modeling result looks plausible?	0.51 (p=0.00)	-0.28 (p=0.13)	0.33 (p=0.08)	0.68 (p=0.00)	1				
Modeling result is completed?	0.36 (p=0.05)	-0.01 (p=0.95)	0.14 (p=0.45)	0.45 (p=0.01)	0.64 (p=0.00)	1			
VGI3D looks useful for 3BMing?	0.47 (p=0.00)	-0.08 (p=0.66)	0.50 (p=0.00)	0.82 (p=0.00)	0.63 (p=0.00)	0.34 (p=0.06)	1		
3D viewing smooth?	0.50 (p=0.00)	0.05 (p=0.78)	0.31 (p=0.09)	0.40 (p=0.03)	0.56 (p=0.00)	0.60 (p=0.00)	0.27 (p=0.14)	1	
Fast modeling speed?	0.44 (p=0.01)	0.10 (p=0.60)	0.72 (p=0.00)	0.37 (p=0.04)	0.28 (p=0.13)	0.08 (p=0.68)	0.38 (p=0.04)	0.30 (p=0.11)	1

of modelling results in Fig. 7 (right) shows a completely different trend from the correlation. Higher ratings suggested that, regardless of whether participants have rich 3D modelling experience or not, they did feel that the modelling results looked plausible.

Next, we would like to know whether or not VGI3D usefulness for the 3D building modelling community was associated with any item of easy usage/understanding of VGI3D, the plausibility of modelling results or the completeness of modelling results. Table 4 clearly demonstrates strong correlations between usefulness for the 3D building modelling community and ease of use/understanding of VGI3D and the plausibility of modelling results with nearly 100% confidence. However, no strong correlation could be discovered between the completeness of modelling results and usefulness.

Our final observation concerned the ease of use/understanding of VGI3D and the plausibility of modelling results. We found a distinct positive correlation with value of 0.68 between them, which appeared to be consistent with our intuition. Until now, all observations and correlations have been based on limited sample data. We can only conjecture the meanings behind them, but they still give us much help and clues to identify which aspects of the system will affect users' perspectives and will be beneficial to us for optimising the system in the future.


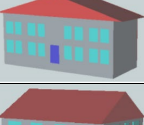
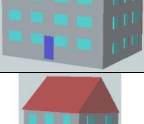
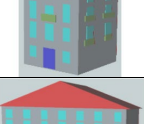

Sensitivity Analysis

To better display the low sensitivity of our system, we selected five different buildings from simple to complex in terms of façade complexity (as illustrated in Table 5) for sensitivity analysis. Building A had the simplest façade structure with 7 windows and it was reconstructed by 2 input images in 48 s. Due to its simple façades, our system correctly detected all windows and hence no need to manually and interactively update. Building A took another 19 s for export as CityGML model where mainly did the rotation, translation and scaling according to the real building size and its real orientation in reality, and did the coordinate transformation (i.e. convert from local coordinate system to world coordinate system).

For building B, its façade became a little more complex with 16 windows and 1 door, and was bigger in building size compared to building A. The door of building B was not correctly detected and was wrongly recognized as a window, probably because of the similar features between them in this case. Thus, we had to manually update the door and it took us another around 15 s. Since building B was reconstructed through one image, it would consume less time when exporting.

Building C had more windows than building B and was reconstructed through 2 images. The testing results showed that semi-auto time was similar to building A's and a similar

Table 5 Summary of buildings with various sizes and façade complexities for sensitive analysis. The fifth column shows the interactive modelling time (semi-auto), time of interactively updating incorrect façade elements (update), exporting 3D model time (export) and updated façade elements in (-). Blue is for door, cyan is for window and green is for balcony

No.	Num of imgs	Building size (L×W×H) m	Façade complexity	Modelling time (s) (semi-auto + update + export)	Output 3D model size (KB)	3D viewing
A	2	11×4.7×8.7	7 windows	48 + 0 + 19	87.9	
B	1	21.4×8.3×12.4	16 windows 1 door	24 + 15 + 11 (1 door)	112.7	
C	2	16×12.1×14.6	21 windows 1 door	49 + 32 + 21 (1 door + 1 window)	150.9	
D	2	11.7×10.4×20.7	17 windows 1 door 5 balconies	47 + 59 + 20 (1 door + 1 balcony + 2 windows)	162.5	
E	1	30.3×15.1×20.6	28 windows 4 doors 6 balconies	25 + 13 + 11 (1 balcony)	238.5	

situation appeared in the building D case. In fact, more than half the time was consumed by user's intersection; however, façade elements detection usually was done within a similar time regardless of the façade complexity. In other words, as long as the proficiency of user interaction can be improved, the overall time can be further reduced. Since two elements were wrongly detected, user spent another 32 s on updating. Meanwhile, exporting time (21 s) similar to building A's was recorded.

Balconies showed up in both building D and E and made the façades become more complicated. However, the semi-auto time and exporting time seemed not to change a lot and still keep similar/stable compared to previous cases. Additionally, updating time is affected by input images' quality and proficiency of user interaction. If doors or windows are obscured by cars, that undoubtedly cannot detect them correctly and thus would increase the time and workload of manual updates, just like two missed windows in building D. The proficiency of user interaction depends on users and hence it is hard to measure. In addition, the only thing that changed was the size of the output 3D model. Their sizes increased with the increment of façade complexity as we expected, but even the most complex building E, its output size was still less than 240 KB.

Finally, in terms of 3D viewing capability, our system gave users a good 3D visualization experience without any laggings. We can also verify this point according to

participants' feedback on Q11 (Is the 3D viewing capability of the model smooth?). All the participants select the 'Tend to agree' or 'Strongly agree'.

In summary, our system has low sensitivity regarding various sizes and complexities of buildings on the functionalities of the system including the modelling time, 3D viewing capability and the output size of 3D models.

Limitations

First of all, the number of participants was relatively small, and their comments and responses may not reflect the views of all those working in the domain of 3D building modelling.

Second, given the limited experience of the design team, we could not carry out sufficient user requirements analysis at the beginning of the system design. For this reason, we could only quickly develop the system and let users give us feedback through usability testing.

Third, the current VGI3D system cannot provide an overview of building models after one user managed to generate a few models. Additionally, current 3D models are visualised in '3D viewer' under the relative coordinate space. If they can be directly visualised on the map with real geographic coordinates, that would be greater.

Fourth, since the building heights are not available except in Trondheim, Norway, the current 3D models after exporting as CityGML are given a fixed height, 12 m. In the future, we plan

to add an option and allow users to enter the number of floors and then the system can roughly estimate the height of buildings. Furthermore, except for Norway, Sweden and Italy, footprints from other countries are not available for the time being.

Fifth, now VGI3D can only reconstruct buildings whose footprints are in rectangle-like shape. We will improve the modelling algorithm and extend it to arbitrary footprints. However, it might lead to a new challenge, mainly on roofs. Since our current roofs are automatically reconstructed according to user's roof type selection, if building's roof is fancy and consists of several different parts, the original method for automatic roof reconstruction will not be appropriate anymore. To solve this challenge, we are considering whether users could sketch a roof as the customized roof type in the future version.

Last but not least, the current performance in façade elements detection still has room for improvement. For example, one participant reported '...almost 25% of windows and one big door were not detected in my test case' and '...if [it] let me manually draw too many windows at one time, I would lose patience'. Although interactively updating models is one of our highlights, ideally, we want to make users reduce interactive manipulation as much as they can. One possible solution to this problem is to extend the training dataset with more different kinds of façade images. Another solution is to collect the labelled data when users manually draw façade elements, then add them into the training dataset and retrain the deep learning model again to attain a better detection accuracy.

Conclusion and Future Work

This paper presented the VGI3D system, which is a web-based and interactive solution for 3D building modelling, from the perspective of software engineering. VGI3D consists of a spatially relational database (PostgreSQL/PostGIS) for the storage and management of spatially geometrical data and other software modules, allowing us to upload images as input, analyse and reconstruct, visualise, modify and export virtual 3D building models according to the OBJ/CityGML standard. Functional and usability testing under limited participants were also conducted, which was intended to assess the potential usefulness of our work for the 3D building modelling community and to better optimise the system and user experience at the same time. We interpreted our findings as that, regardless of whether or not participants have rich 3D modelling experience, they did think VGI3D is useful and has promising value for the 3D building modelling community. However, our system still has much room for improvement and optimisation. On one hand, it is necessary to further strengthen our CNN model and enable it to detect façade elements as accurately as possible so as to reduce the user interaction costs for updating 3D models. On

the other hand, from suggestions/comments, it was apparent that reconstructing complex buildings would be helpful for applications that place high demands on models.

Overall, our VGI3D needs less input and user interaction and is easy to manipulate. It is suitable for quick modelling but still with relatively high details (i.e. LoD3). We hope our work could provide a new idea for the 3D building modelling domain and let more volunteers join this community to contribute to wider applications in smart cities.

Furthermore, VGI3D is currently running on an Amazon cloud server (EC2). At the end of this year or the beginning of 2022, it will be moved to the website of Trondheim municipality and released as a citizen participation project in Trondheim. The announcement regarding the host change will be made publicly available when the new website is ready to be accessed.

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1007/s41651-021-00086-7>.

Author Contribution Hongchao Fan formed the research idea and revised the manuscript. Chaoquan Zhang designed and conducted the experience, analysed the results and drafted the manuscript. Gefei Kong co-conducted the usability testing and analysed the results. All authors read and approved the final manuscript.

Funding This research was supported by research fund from the Norwegian Public Roads Administration under project name λ Road.

Data Availability The FacadeWHU dataset that support this study are available at: <https://drive.google.com/drive/folders/1BeKXMuNAARcwP6lSeiqFOSfscbP9ZKm7?usp=sharing>

Code Availability The code is currently stored on a local area network (LAN) of university and have not submitted to like Github. If this paper is accepted, we will release the code there immediately.

Declarations

Ethics Approval Not applicable.

Consent to Participate Not applicable.

Consent for Publication All authors approve that the Journal of Computational Urban Science can publish our article.

Conflict of Interest The authors declare no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

References

- Becker S (2009) Generation and application of rules for quality dependent façade reconstruction. *ISPRS J Photogramm Remote Sens* 64(6):640–653
- Biljecki F, Ledoux H, Stoter J (2016) An improved LOD specification for 3D building models. *Comput Environ Urban Syst* 59:25–37
- Blut C, Blankenbach J (2021) Three-dimensional CityGML building models in mobile augmented reality: a smartphone-based pose tracking system. *International Journal of Digital Earth* 14(1):32–51
- Dehbi Y, Plümer L (2011) Learning grammar rules of building parts from precise models and noisy observations. *ISPRS J Photogramm Remote Sens* 66(2):166–176
- Dehbi Y, Hadji F, Gröger G, Kersting K, Plümer L (2017) Statistical relational learning of grammar rules for 3D building reconstruction. *Trans GIS* 21(1):134–150
- Dirksen J (2015) *Learning Three.js—the JavaScript 3D Library for WebGL*. Packt Publishing Ltd
- Eicker U, Zirak M, Bartke N, Rodríguez LR, Coors V (2018) New 3D model based urban energy simulation for climate protection concepts. *Energy and Buildings* 163:79–91
- Fan H, Kong G, Zhang C (2021) An Interactive platform for low-cost 3D building modeling from VGI data using convolutional neural network. *Big Earth Data* 5(1):49–65
- Flask (2021). <https://flask.palletsprojects.com/en/1.1.x/>. (Accessed 29 March, 2021).
- Furukawa Y, Ponce J (2009) Accurate, dense, and robust multiview stereopsis. *IEEE Trans Pattern Anal Mach Intell* 32(8):1362–1376
- Gröger G, Plümer L (2012) CityGML—Interoperable semantic 3D city models. *ISPRS J Photogramm Remote Sens* 71:12–33
- Haynes P, Hehl-Lange S, Lange E (2018) Mobile augmented reality for flood visualisation. *Environ Model Softw* 109:380–389
- Kim H, Han S (2018) Interactive 3D building modeling method using panoramic image sequences and digital map. *Multimed Tools Appl* 77(20):27387–27404
- Kong G, Fan H (2020) Enhanced Facade Parsing for Street-Level Images Using Convolutional Neural Networks. *IEEE Trans Geosci Remote Sens*. <https://doi.org/10.1109/TGRS.2020.3035878>
- Leaflet (2021). <https://leafletjs.com/>. (Accessed 29 March, 2021)
- Li L, Tang L, Zhu H, Zhang H, Yang F, Qin W (2017) Semantic 3D modeling based on CityGML for ancient Chinese-style architectural roofs of digital heritage. *ISPRS Int J Geo Inf* 6(5):132
- Li L, Lei Y, Tang L, Yan F, Luo F, Zhu H (2019) A 3D spatial data model of the solar rights associated with individual residential properties. *Comput Environ Urban Syst* 74:88–99
- Liu H, Li W, Zhu J (2021) Translational Symmetry-Aware Facade Parsing for 3D Building Reconstruction. arXiv preprint arXiv:2106.00912
- Liu J, Luo J, Hou J, Wen D, Feng G, Zhang X (2020) A BIM Based Hybrid 3D Indoor Map Model for Indoor Positioning and Navigation. *ISPRS Int J Geo Inf* 9(12):747
- Machete R, Falcão AP, Gomes MG, Rodrigues AM (2018) The use of 3D GIS to analyse the influence of urban context on buildings' solar energy potential. *Energy and Buildings* 177:290–302
- Mapbox (2021). <https://www.mapbox.com/>. (Accessed 29 March, 2021)
- Monteiro CS, Costa C, Pina A, Santos MY, Ferrão P (2018) An urban building database (UBD) supporting a smart city information system. *Energy and Buildings* 158:244–260
- Musialski P, Wonka P, Aliaga DG, Wimmer M, Van Gool L, Purgathofer W (2013) A survey of urban reconstruction. In *Computer graphics forum* (Vol. 32, No. 6, pp. 146–177)
- Nishida G, Garcia-Dorado I, Aliaga DG, Benes B, Bousseau A (2016) Interactive sketching of urban procedural models. *ACM Transactions on Graphics (TOG)* 35(4):130
- Park Y, Guldmann JM, Liu D (2021) Impacts of tree and building shades on the urban heat island: Combining remote sensing, 3D digital city and spatial regression approaches. *Comput Environ Urban Syst* 88:101655
- Redmon J, Farhadi A (2018) YoloV3: An incremental improvement. arXiv preprint arXiv:1804.02767
- Sangiambut S, Sieber R (2016) The V in VGI: Citizens or civic data sources. *Urban Plan* 1(2):141–154
- Stadler A, Nagel C, König G, Kolbe TH (2009) Making interoperability persistent: A 3D geo database based on CityGML. In *3D Geoinformation sciences* (pp. 175–192). Springer, Berlin, Heidelberg
- Sun S, Salvaggio C (2013) Aerial 3D building detection and modeling from airborne LiDAR point clouds. *IEEE J Sel Top Appl Earth Obs Remote Sens* 6(3):1440–1449
- Tang L, Ying S, Li L, Biljecki F, Zhu H, Zhu Y, Yang F, Su F (2020) An application-driven LOD modelling paradigm for 3D building models. *ISPRS J Photogramm Remote Sens* 161:194–207
- Templin T, Popielarczyk D (2020) The use of low-cost unmanned aerial vehicles in the process of building models for cultural tourism, 3D web and augmented/mixed reality applications. *Sensors* 20(19):5457
- Tran H, Khoshelham K, Kealy A, Díaz-Vilarinho L (2019) Shape grammar approach to 3D modeling of indoor environments using point clouds. *J Comput Civ Eng* 33(1):04018055
- Ullman S (1979) The interpretation of structure from motion. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, 203(1153), 405–426
- Wang R, Peethambaran J, Chen D (2018) LiDAR point clouds to 3-D Urban Models: A Review. *IEEE J Sel Top Appl Earth Obs Remote Sens* 11(2):606–627
- Wolberg G, Zokai S (2018) PhotoSketch: a photocentric urban 3D modeling system. *Vis Comput* 34(5):605–616
- Wu B, Yu B, Wu Q, Yao S, Zhao F, Mao W, Wu J (2017) A graph-based approach for 3D building model reconstruction from airborne LiDAR point clouds. *Remote Sens* 9(1):92
- Xiong B, Elberink SO, Vosselman G (2014) A graph edit dictionary for correcting errors in roof topology graphs reconstructed from point clouds. *ISPRS J Photogramm Remote Sens* 93:227–242
- Yang B, Dong Z (2013) A shape-based segmentation method for mobile laser scanning point clouds. *ISPRS J Photogramm Remote Sens* 81:19–30
- Yu Q, Helmholtz P, Belton D (2017) Semantically enhanced 3D building model reconstruction from terrestrial laser-scanning data. *J Surv Eng* 143(4):04017015

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Appendix A: User feedback form

Part 1: Basic information about participant

Q1: How old are you?

Q2: What is your gender?

Q3: What is your research interest?

Q4: Have you ever got involved in applications of 3D building models?

Not at all Slightly Moderately Very Extremely

Q5: How experienced are you with using browser?

Not at all Slightly Moderately Very Extremely

Q6: How experienced are you with 3D building modeling?

Not at all Slightly Moderately Very Extremely

Part 2: Evaluation about the VGI3D

Q7: Is VGI3D easy to use/understand?

Strongly disagree Tend to disagree Neither nor Tend to agree Strongly agree

Q8: Does the modeling result look plausible?

Strongly disagree Tend to disagree Neither nor Tend to agree Strongly agree

Q9: Is the modeling result completed?

Strongly disagree Tend to disagree Neither nor Tend to agree Strongly agree

Q10: Is VGI3D useful for 3D building modeling?

Strongly disagree	Tend to disagree	Neither nor	Tend to agree	Strongly agree
-------------------	------------------	-------------	---------------	----------------

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

Q11: Is the 3D viewing capability of the model smooth?

Strongly disagree	Tend to disagree	Neither nor	Tend to agree	Strongly agree
-------------------	------------------	-------------	---------------	----------------

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

Q12: Is the modeling speed fast?

Strongly disagree	Tend to disagree	Neither nor	Tend to agree	Strongly agree
-------------------	------------------	-------------	---------------	----------------

<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>	<input type="checkbox"/>
--------------------------	--------------------------	--------------------------	--------------------------	--------------------------

Q13: Do you have any suggestions/comments for the VGI3D?

Thank you for your valuable responses and taking the time to complete this form.

Appendix B: Raw data collected from participants

No.	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12
1	28	M	A	5	4	5	5	5	4	5	5	5
2	43	M	A	4	4	4	4	4	3	4	5	5
3	30	M	A	5	4	5	4	4	4	4	5	5
4	26	F	A	5	5	5	5	5	4	5	5	5
5	38	M	B	5	5	3	4	4	4	4	5	4
6	35	M	B	4	4	3	5	4	4	5	5	4
7	27	F	C	5	3	3	4	4	4	4	4	4
8	26	M	C	4	4	4	4	4	4	4	5	5
9	30	M	C	4	4	3	4	4	4	4	5	4
10	31	M	D	4	4	3	4	4	4	4	5	4
11	28	M	D	4	3	3	4	4	3	4	4	4
12	32	F	E	3	5	4	4	3	3	4	4	5
13	27	F	E	5	4	2	4	4	3	4	5	4
14	28	M	E	4	4	2	4	4	4	3	5	4
15	29	M	E	3	5	3	4	3	3	4	4	4
16	29	M	F	3	5	1	4	4	4	4	4	4
17	33	M	F	3	3	1	4	4	3	4	4	4
18	28	M	F	2	5	1	4	4	4	4	5	4
19	27	M	G	2	4	1	4	4	4	4	5	4
20	25	M	G	3	5	1	4	4	4	4	5	4
21	28	M	G	1	5	1	3	3	3	3	4	4
22	27	M	G	1	4	1	4	4	4	4	4	4
23	29	F	G	2	4	3	4	4	3	4	4	4
24	33	M	H	2	4	1	4	4	4	4	5	4
25	37	M	H	1	4	1	4	4	3	4	4	4
26	28	F	I	2	4	1	4	3	3	3	4	4
27	27	M	I	1	5	1	2	2	2	2	4	4
28	26	F	J	3	4	2	4	4	4	4	4	4
29	28	M	J	3	4	2	4	4	4	3	5	4
30	34	M	J	3	4	2	4	3	3	4	4	4

For Q2, M = Male and F = Female.

For Q3, A = 3D city modelling; B = 3D building model visualization; C = photogrammetry; D = urban planning; E = BIM for asset assessment/management; F = spatial analysis; G = computer science; H = GPS; I = railway design; J = facility management

Appendix C

Table 1. Standard deviation (σ) comparisons between different types of practitioners regarding to each question in user feedback form.

	Involved in 3BMs apps?	Experienced using browser?	Experienced with 3DMing?	VGI3D easy to use?	Modeling result looks plausible?	Modeling result is completed?	VGI3D looks useful for 3BMing?	3D viewing smooth?	Fast modeling speed?
3D city modeling	0.43	0.43	0.43	0.50	0.5	0.43	0.50	0	0
3D visualization	0.50	0.50	0	0.50	0	0	0.50	0	0
Photogrammetry	0.47	0.47	0.47	0	0	0	0	0.47	0.47
Urban planning	0	0.50	0	0	0	0.50	0	0.50	0
BIM for asset assessment / management	0.83	0.50	0.83	0	0.50	0.43	0.43	0.50	0.43
Spatial analysis	0.47	0.94	0	0	0	0.47	0	0.47	0
Computer science	0.75	0.49	0.80	0.40	0.40	0.49	0.40	0.49	0
GPS	0.50	0	0	0	0	0.50	0	0.50	0
Railway design	0.50	0.50	0	1.00	0.50	0.50	0.50	0	0
Facility management	0	0	0	0	0.47	0.47	0.47	0.47	0

Table 2. Skewness comparisons between different types of practitioners regarding to each question in user feedback form.

	Involved in 3BMs apps?	Experienced using browser?	Experienced with 3DMing?	VGI3D easy to use?	Modeling result looks plausible?	Modeling result is completed?	VGI3D looks useful for 3BMing?	3D viewing smooth?	Fast modeling speed?
3D city modeling	-1.15	1.15	-1.15	0	0	-1.15	0	0	0
3D visualization	0	0	0	0	0	0	0	0	0
Photogrammetry	0.71	-0.71	0.71	0	0	0	0	-0.71	0.71
Urban planning	0	0	0	0	0	0	0	0	0
BIM for asset assessment / management	0.49	0	0.49	0	0	1.15	-1.15	0	1.15
Spatial analysis	-0.71	-0.71	0	0	0	-0.71	0	0.71	0
Computer science	0.34	0.41	1.50	-1.50	-1.50	-0.41	-1.50	0.41	0
GPS	0	0	0	0	0	0	0	0	0
Railway design	0	0	0	0	0	0	0	0	0
Facility management	0	0	0	0	-0.71	-0.71	-0.71	0.71	0

PAPER 2

**An interactive platform for low-cost 3D building modeling from VGI
data using convolutional neural network**

Hongchao Fan^{a*}, Gefei Kong^b, Chaoquan Zhang^a

^aDepartment of Civil and Environmental Engineering, Norwegian University of Science and
Technology, Trondheim, Norway

^bSchool of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China

This paper is published in *Big Earth Data*, April 2021; 5:1, 49-65.

Research publications



An Interactive platform for low-cost 3D building modeling from VGI data using convolutional neural network

Hongchao Fan^a, Gefei Kong^{b*} and Chaoquan Zhang^{a*}

^aDepartment of Civil and Environmental Engineering, Norwegian University of Science and Technology, Trondheim, Norway; ^bSchool of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China

ABSTRACT

The applications of 3D building models are limited as producing them requires massive labor and time costs as well as expensive devices. In this paper, we aim to propose a novel and web-based interactive platform, *VGI3D*, to overcome these challenges. The platform is designed to reconstruct 3D building models by using free images from internet users or volunteered geographic information (VGI) platform, even though not all these images are of high quality. Our interactive platform can effectively obtain each 3D building model from images in 30 seconds, with the help of user interaction module and convolutional neural network (CNN). The user interaction module provides the boundary of building facades for 3D building modeling. And this CNN can detect facade elements even though multiple architectural styles and complex scenes are within the images. Moreover, user interaction module is designed as simple as possible to make it easier to use for both of expert and non-expert users. Meanwhile, we conducted a usability testing and collected feedback from participants to better optimize platform and user experience. In general, the usage of VGI data reduces labor and device costs, and CNN simplifies the process of elements extraction in 3D building modeling. Hence, our proposed platform offers a promising solution to the 3D modeling community.

ARTICLE HISTORY

Received 29 October 2020
Accepted 28 January 2021

KEYWORDS

3D building modeling; VGI; convolutional neural network; user interaction; low cost

1. Introduction

3D building models play a significant role in designing urban 3D and virtual cities. They support a variety of applications in urban planning, environment analysis, virtual tourism, and augmented reality (Haala & Kada, 2010; Verma, Kumar, & Hsu, 2006). 3D building models are usually divided into five level of details (LoDs) (Biljecki et al., 2016) according to the CityGML 2.0 standard (Gröger, Kolbe, Nagel, & Häfele, 2012). LoD0 model means the footprint of a building; LoD1 model is represented as a cuboid by extruding the LoD0 model. On the basis of LoD1, LoD2 model contains the roof shape of a building, and the semantic information of building elements. Compared with LoD2, LoD3 model is more complex and can be called as “architecturally detailed model”, which contains facade elements of a building, such as windows and doors. LoD4 model is more complete and

CONTACT Hongchao Fan hongchao.fan@ntnu.no

*These authors contributed equally to this work.

has both internal and external building elements. 3D building modeling in LoD1 and LoD2 has had many mature approaches (Kim & Han, 2018; Li et al., 2019; Wu et al., 2017), and is a key focus for some platforms (Preka & Doulamis, 2016), but these fields actually prefer 3D building models in LoD3 or higher.

Classic methods that can create highly detailed 3D building models usually adopt grammar and topology rules (Becker, 2009; Dehbi & Plümer, 2011). While their productions are stable and complete, only limited architectural styles can be covered. In recent years, with the development of algorithm, software and hardware in computer science, more studies focused on automatic 3D building modeling and achieved a great success (Agarwal et al., 2011), which can reconstruct 3D models in most scenarios. For instance, automatic methods based on point cloud data from laser or LiDAR (Yu, Helmholz, & Belton, 2017) not only guarantee very high modeling precision, but also eliminate the defects of limited architectural styles. However, these automatic methods need the assistance of professional and expensive devices to ensure the size and quality of data to obtain impressive results. All of these requirements lead to higher costs on labor and time. Moreover, they are sensitive to noise and thus easily lead to unstable and incomplete results. In other words, that means automatic methods are difficult to reconstruct 3D models effectively when lacking data or facing data noise, without the help from users.

To address these issues, some researchers tried to reconstruct 3D building models in an interactive way. Wolberg and Zokai (2018) proposed a photocentric 3D modeling platform, and added user interaction capabilities to overcome the instability of automatic methods. Users firstly sketched the building's walls. Then structure from motion (SfM) algorithm (Ma, Soatto, Kosecka, & Sastry, 2012) was used to generate point clouds in order to more accurately reconstruct building models. However, SfM needs many images that belong to the same building (more images, the better the result) and need to know image's GPS location and camera internal parameters for camera calibration. In their experiment, it took them around 20 minutes to reconstruct two building models. That indicated its computational inefficiency and strong reliance on the number of images. Nishida, Garcia-Dorado, Aliaga, Benes, and Bousseau (2016) achieved highly detailed 3D building modeling with only single image by combining sketch-based and procedural methods. However, users were asked to draw windows and doors and select their corresponding styles. It is not friendly to non-expert users, as they are not familiar with it. In addition, they defined some patterns based on the buildings they previously used, and hence it might not be helpful to buildings with different styles. Although other interactive methods are able to reconstruct 3D building models from one image as well (Chen, Zhu, Shamir, Hu, & Cohen-Or, 2013; Zheng et al., 2012), they are dependent on accurate edge detection. They could be seriously affected by image's luminance.

In short, many existing methods rely on the size of datasets and are time consuming. Methods with higher efficiency and less image data often require user sketching, which is not convenient and friendly for non-expert users. Methods based on multiple images are often influenced by data quality, such as illumination change, camera position change, and perspective distortion. The collection and processing of datasets consume a significant amount of time and labor as well. Methods based on point clouds are sensitive to noise points. Thus additional data processing is typically necessary to solve this issue. Finally, existing automated methods for building modeling are time consuming

because they usually have to estimate camera's position and generate point clouds from a sequence of images.

Considering all the above shortcomings, a low-cost 3D building modeling method is necessary, and it can accept lower-quality images from various devices without additional information, such as GPS or camera parameters, which will reduce the cost of data collection. In addition, this method can reconstruct 3D building models with less image data. Hence, we designed an interactive platform (called *VG/3D*) based on volunteered geographic information (VGI) images for 3D building modeling. In our platform, we take VGI image data as input and all the facades prepared for modeling should be fully captured on an image. Users only need to simply outline the facade of a building, and then our platform will automatically detect and adjust the bounding boxes of facade elements. Finally, 2D locations of all the elements will be transformed into 3D coordinate system, and 3D model of this building will be shown to users in real time.

VGI employs tools to create, assemble, and disseminate geographic data provided voluntarily by individuals (Sangiambut & Sieber, 2016). With the development of WebGL and hardware, VGI data have played a more active role in geoinformation collection, especially in the collection of urban images. Although the quality of VGI images is difficult to guarantee and measure, the low-cost and high-richness of VGI image data can narrow existing research gaps to some extent. Furthermore, non-expert users modeling 3D buildings will be conducive to the development and spread of urban 3D modeling.

Compared to existing methods, the key contributions of our work are as follows:

- (1) Low-cost data requirements: the input to our platform is VGI image data. These images are captured by non-expert users or volunteers who upload images to the VGI image platform. Compared to traditional methods of data collection, VGI image data have lower costs, and can cover larger areas. The use of VGI image data will significantly reduce the time and labor costs during the data collection process. Moreover, considering that the geolocations of VGI images are usually scattered and that these images are at a street level, only one or two views of a building can be obtained in many cases. Hence, our platform is designed to model simple 3D buildings using less than two images. This implies that the time required and cost incurred for 3D modeling are lower.
- (2) Fast 3D building modeling: the algorithm for facade elements extraction embedded in our platform can automatically extract windows, doors and balconies within one second. Additionally, the entire workflow is completed in 30 seconds, which is faster than most existing automatic building modeling methods.
- (3) Relatively High-quality modeling results: in a low-cost environment, our platform can obtain modeling results that are of relatively high quality. The modeling results will be complete and stable even on low-quality images. Meanwhile, semantic information, such as facades, windows, balconies, and roofs, will be obtained and shown to users.

The remainder of this paper is organized as follows. [Section 2](#) describes the modeling workflow and detailed algorithms of our platform. Experimental results and a platform usability testing are going to present in [Section 3](#). [Section 4](#) provides the conclusions as well as future work.

2. Methodology

2.1. Workflow

Considering the convenience and availability of the proposed method, we developed a web-based platform embedded with all the algorithms (as shown in Figure 1). First, our platform accepts building images uploaded by users. Then, users provide three pieces of information to our web platform in the way of interaction, including facades boundaries, roof type, and facade views of the building in images. Next, images from users are taken into the module of facade elements extraction to automatically obtain the locations of the windows, doors and balconies. In this module, the object detection network in facade elements extraction module automatically detects facade elements and gives them semantic labels. Finally, these locations in 2D are transformed into the 3D coordinate system to construct the 3D building model and show it on the Web in real time. The generated 3D model can also be downloaded for further research.

2.2. User interaction

Since most input VGI images are at a street level and only one or two facades of a building are available, multi-view facades of a single building are difficult to obtain by non-expert volunteers. Actually, 3D building models for urban planning or virtual cities do not

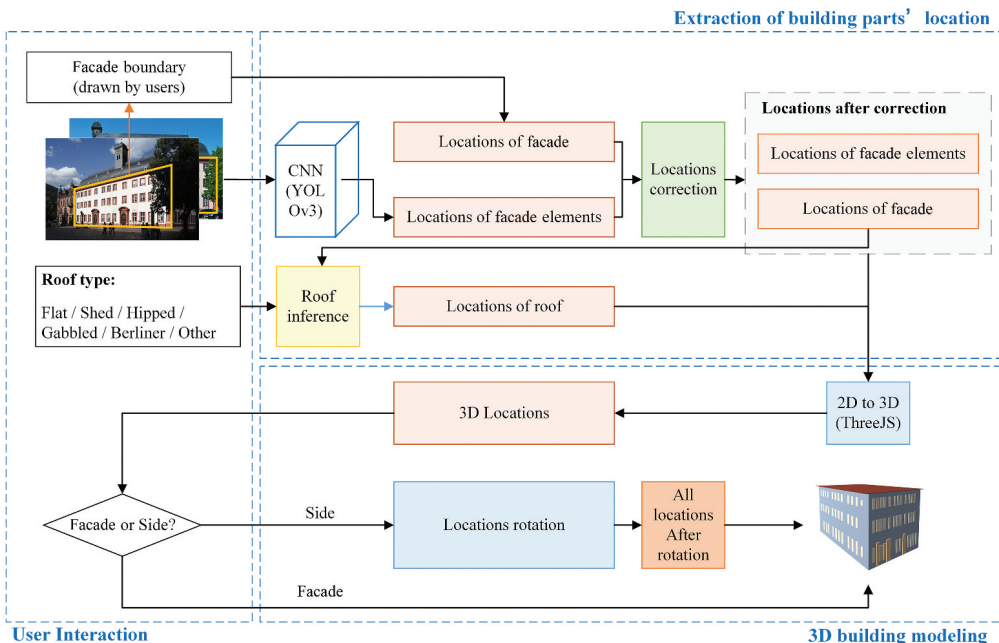


Figure 1. Framework of our platform. In the user interaction module, users provide the facades boundaries, roof type, and facade views of the building in images to our web platform. Then, images are fed into facade elements extraction module to automatically obtain the locations of the windows, doors and balconies. In the 3D building modeling step, the locations are transformed from 2D to 3D space. Then locations are rotated according to the information of facades views. Finally, the 3D building model is built and visualized to users.

require over-detailed models. Hence, in the user interaction module, users only need to upload no more than two images of a building. Facade and side images for the same building can be uploaded together to obtain a more comprehensive model. The number of images was recorded as N_i . In terms of the image requirements, there are no specific requirements regarding the cameras used or the overlap percentages. The only requirement is that users should ensure each image containing a complete facade of the building. Then, users draw the corners of the facade (X_F, Y_F) in an image to create a corresponding facade boundary, where $X_F = \{X_{F1}, X_{F2}, \dots, X_{Fn}\}$ and $Y_F = \{Y_{F1}, Y_{F2}, \dots, Y_{Fn}\}$, $(X_F, Y_F) \in \{(X_F, Y_F)\} = \{(X_F, Y_F)_{ni} \mid ni = 1, \dots, N_i\}$. The corners are used to build 3D facade and correct the locations of the facade elements. Meanwhile, the roof type of the building should be selected to help reconstruct the roof. These images and facade boundaries will be sent to our facade extraction method for the preparation of 3D building modeling.

2.3. Extracting locations of facade elements

2.3.1. Detecting facade elements

In user interaction module, we have obtained images and locations of the building facades. Images from our user interaction module are usually at a street level with various complex scenes, such as multi-view scenes, complex illumination and background. Traditional methods cannot handle these scenes well. Existing methods of interactive modeling require users to draw facade elements, such as windows and doors, which is time-consuming and labor-intensive. In addition, the accuracy and completeness of facade elements cannot be ensured in user drawings. Hence, we want to use a convolutional neural network (CNN) for object detection or semantic segmentation, to extract facade elements. However, many windows and balconies are too small in street-level images, and might be difficult to be completely segmented by semantic segmentation network. CNNs for semantic segmentation are not suitable for extracting them. Hence, we chose an object detection CNN, YOLO v3 (Redmon & Farhadi, 2018), for detecting facade elements (Kong & Fan, 2020). YOLO v3 is a one-stage, fast, and highly accurate object detection neural network. It can maintain a significant balance between detection accuracy and speed. As Kong and Fan (2020) have done, the Darknet53 was used as the backbone of YOLO v3. And the detection network was pretrained on our FacadeWHU dataset to detect windows, doors and balconies. Our dataset contains 900 street-level images (850 from Paris, France, and 50 from Trondheim, Norway) and corresponding annotations for semantic segmentation and object detection. These annotations contain six classes – window, door, balcony, roof, shop, and wall, which can meet the requirement of our facade elements detection. Uploaded images were detected directly by our trained model from Kong and Fan’s work (2020). Then, every location (which is also called “bounding box”) of windows, balconies, and doors was obtained and organized as $(C_{cnn1}, C_{cnn2}, classcode) = (x_{cnn1}, y_{cnn1}, x_{cnn2}, y_{cnn2}, classcode)$ for correction. $(C_{cnn1}, C_{cnn2}, classcode) \in \{(C_{cnn1}, C_{cnn2}, classcode)\} = \{(C_{cnn1}, C_{cnn2}, classcode)_{ni}^{ki} \mid k = 1, \dots, N_c; i = 1, \dots, n_k; ni = 1, \dots, N_i\}$, where N_c is the number of classes, n_k is the number of facade elements in every class of a facade, and N_i is the number of input images.

2.3.2. Correction and inference of locations of facade elements

The locations of the building facade and its elements are based on images. However, original VGI images commonly have perspective distortion, which distorts the shape and

layout of facade elements. Therefore, the locations of the facade and corresponding elements (windows, doors and balconies) cannot be directly applied to 3D building modeling. To solve this problem, we added a submodule for location correction. We considered the following measures to correct perspective distortion for the facade in every image:

(1) First, we calculated the aspect ratio $r_F = h_f/w_f$ of the bounding box of every corresponding facade. h_f and w_f are the height and width of the facade bounding box from the user interaction module, respectively. They were calculated using Equation (1). At the same time, the facade height after perspective correction is defined as the image height h_p , and the facade width after perspective correction is defined as $w_p = h_p/r_F$. Then, we obtained the bounding box of the facade after perspective correction, as $((0, 0), (w_p, h_p))$, and the location of facade is denoted as $(X_{pF}, Y_{pF}) = ((0, 0, w_p, w_p), (0, h_p, h_p, 0))$.

$$h_f = \max(Y_f) - \min(Y_f), w_f = \max(X_f) - \min(X_f) \quad (1)$$

(2) Second, we calculated a homography matrix, M , by the bounding box of facade before and after perspective correction as Equations (2)–(4). The homography matrix is regarded as a perspective transformation matrix and was used for perspective correction.

$$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = M \begin{bmatrix} x_f \\ y_f \\ 1 \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} x_f \\ y_f \\ 1 \end{bmatrix} \quad (2)$$

$$x_p = \frac{x'}{w'} = \frac{m_{11}x_f + m_{12}y_f + m_{13}}{m_{31}x_f + m_{32}y_f + m_{33}} \quad (3)$$

$$y_p = \frac{y'}{w'} = \frac{m_{21}x_f + m_{22}y_f + m_{23}}{m_{31}x_f + m_{32}y_f + m_{33}} \quad (4)$$

where (x_f, y_f) is a facade location from user interaction module input (X_f, Y_f) . (x_p, y_p) is the facade location after perspective correction.

(3) Then, the location of the facade and its elements were corrected using the perspective transformation matrix M . After this step, locations from user interaction and CNN, including (X_f, Y_f) for the facade and $\{(C_{cnn1}, C_{cnn2}, classcode)_{ni}\}$ for facade elements, were corrected. And locations after perspective correction were obtained and denoted as (X_{pF}, Y_{pF}) for the facade and $\{(C_{p1}, C_{p2}, classcode)_{ni}\}$ for facade elements, where every facade element's location of this facade is organized as $(C_{p1}, C_{p2}, classcode)_{ni} = ((x_{p1}, y_{p1}), (x_{p2}, y_{p2}), classcode)_{ni}$; $(C_{p1}, C_{p2}, classcode)_{ni} \in \{(C_{p1}, C_{p2}, classcode)_{ni}\}$.

The perspective distortion of the locations has been corrected. However, the facade-element layout of the perspective-corrected locations were misaligned, which will still affect the 3D building modeling process. Hence, in this step, we proposed a layout correction method for facade elements in every image:

(a) For every facade, first, we reorganized locations of facade elements after perspective correction from xy - xy to xy - wh as $\{(x_{pc}, y_{pc}), (w_p, h_p), classcode)_{ni}\}$. (x_{pc}, y_{pc}) is the center of every facade element, and (w_p, h_p) is the width and height of every facade element, respectively. Then, the locations were separated by $classcode$ and output as $\{(x_{pc}^k, y_{pc}^k), (w_p^k, h_p^k), classcode^k)_{ni}\}$, where $k = 1, 2, \dots, N_c$, and N_c is the number of classes. In

the following steps, we corrected the layout of the facade elements of a facade in every class.

(b) Second, in every class, the new $xy-wh$ locations were sorted by y_{pc} . We calculated the height differences between two neighboring locations and obtained the result of every class as $(h_{diff_p}^k, classcode^k)$.

$$\begin{aligned} h_{diff_p}^{k,i} &= y_{pc}^{k,i+1} - y_{pc}^{k,i} \\ k &= 1, 2, \dots, N_c; i = 1, 2, \dots, (n_k - 1) \end{aligned} \quad (5)$$

where N_c is the number of classes, and n_k is the number of facade elements in every class of a facade.

(c) Third, a cluster algorithm, *k-means++* (Arthur & Vassilvitskii, 2007), was used to divide the height differences into two groups. One group G_s had small height differences, which refers to the height difference of the facade elements on a single floor. The other group G_l had large height differences, which refers to the height difference of the facade elements between neighboring floors. The number of floors, N_f , was obtained using G_l as $N_f = N_{G_l} + 1$. N_{G_l} is the number in group G_l .

(d) Fourth, the facade elements in every class were divided into N_f groups based on the value of G_l and y_{pc}^k . Each group referred to one floor of the building. In every floor, we calculated the average y -center coordinate y_{ac}^{kj} and average height he_a^{kj} of the facade elements, $j = 1, 2, \dots, N_f$. The y -center coordinates of every facade element in every floor y_{pc}^{kj} is corrected to y_{ac}^{kj} , and the height of every facade element in every floor he_p^{kj} is corrected to he_a^{kj} .

(e) Fifth, the x -center coordinates of the facade elements also needed correction. Hence, we repeated steps (b) to (d). In these steps, the height differences, y -center coordinates and average height were replaced with width differences, x -center coordinates and average width. Finally, the layout of the facade elements was corrected.

The locations of the facade elements after layout correction are denoted as $\{(x_{ac}, y_{ac}), (w_e, he_a), classcode\}_{ni}$. This $xy-wh$ format cannot be used for 3D building modeling. Hence, we changed the $xy-wh$ locations to $xy-xy$ locations and obtained the location of every facade element as $(C_{1l}, C_{12}, classcode)_{ni} = ((x_{1l}, y_{1l}), (x_{12}, y_{12}), classcode)_{ni}$.

Steps (1) to (3) and (a) to (e) in section 2.3.2 were repeated to obtain the locations of all facades $\{(X_{pF}, Y_{pF})\}$ and locations of their facade elements $\{(C_{1l}, C_{12}, classcode)\}$.

Then the location of roof was calculated based on the location of facade from the image in the front direction. In general, basic roof shapes are divided into five classes: flat, shed, gabled, hipped, and berliner (Kada & McKinley, 2009). Hence, we set the basic roof types as these five types. After the users choose the corresponding roof type of the building from the five basic roof types in the user interaction module, we can reconstruct the roof based on the corrected facade corners. The group with corners on the top edge of the facade is selected and is regarded as the bottom edge of the roof. Then, the top edge of the roof is automatically calculated according to the roof type. The top and bottom edges of the roof were concatenated as the final roof location (X_R, Y_R) .

We grouped the locations of facades, a roof, and facade elements into N_l groups according to the number of images, and then sent them to the 3D building modeling module for 3D modeling.

2.4. 3D building modeling

After the location detection module, we obtained the locations of facades as $\{(X_{pF}, Y_{pF})\}$, the locations of windows, balconies, doors as $\{(C_{1i}, C_{2i}, \text{classcode})\}$, and the location of roof as (X_R, Y_R) . The range of these locations except roof is from 0 to h_p . h_p is based on every image height, which varies depending on the image. Thus, the size of the building models cannot remain stable. Hence, it is necessary to normalize these locations to the same range based on the height of every facade.

For locations from every image ni , we normalized these locations from range $(0, h_p)$ to range $(0, h_{th})$ in height, and from range $(0, w_p)$ to $(0, h_{th}/r_F)$ in width as shown in Equation (6). h_{th} is a constant and we defined it as 10 to ensure the same height of all images. Moreover, to obtain a better visualization result, the center of all locations will be moved to $(0, 0)$ by location translation.

$$\begin{aligned} x_{nr}^{k,i} &= x_l^{k,i}/w_p \times h_{th}/r_F - \frac{1}{2}(h_{th}/r_F), y_{nr}^{k,i} = y_l^{k,i}/h_p \times h_{th} - \frac{1}{2}h_{th} \\ k &= 1, 2, \dots, N_c, \quad i = 1, 2, \dots, n_k \end{aligned} \quad (6)$$

The location of the facade is recorded as (X_{nrF}, Y_{nrF}) , and the location of every facade element after normalization is recorded as $(C_{nr1}, C_{nr2}, \text{classcode})_{ni} = ((x_{nr1}, y_{nr1}), (x_{nr2}, y_{nr2}), \text{classcode})$, $(C_{nr1}, C_{nr2}, \text{classcode})_{ni} \in \{(C_{nr1}, C_{nr2}, \text{classcode})_{ni}\}$.

The normalized locations of facade and other elements are still in the 2D coordinate system. For 3D building models, these locations need to be transformed into a 3D coordinate system. Hence, we utilized an easy and lightweight JavaScript 3D library, Three.js (Dirksen, 2015), to address this problem. Three.js provides a modeling function, *ExtrudeGeometry*, to build 3D shapes from 2D polygons. This function is applied to our platform to transform the coordinate system. The 3D locations after transformation are recorded as $((X_{3D}, Y_{3D}, Z_{3D}), \text{classcode})_{ni}$. $X_{3D} = \{X_{3D}^0, X_{3D}^1, \dots, X_{3D}^{N_{bp}}\}$, and Y_{3D}, Z_{3D} are the same as X_{3D} ; $X_{3D}^i = (x_{3D}^{i0}, x_{3D}^{i1}, \dots, x_{3D}^{i(co-1)})$. N_{bp} is the number of all the elements, which can be obtained by $N_{bp} = 1 + \sum_{k=1}^{N_c} n_k$. "1" in this equation means a facade number. co is the number of corners for the facade and every facade element. For the image in the front direction, the location of roof needs to be considered, so $N_{bp} = 2 + \sum_{k=1}^{N_c} n_k$, where "2" in this equation means the number of a roof and a facade, and co is the number of corners for facade, roof and every facade element when transforming locations from the image in the front direction.

We repeat the steps from section 2.4 for every input image and obtain the N_i 3D location groups of one building as $\{((X_{3D}, Y_{3D}, Z_{3D}), \text{classcode})\} = \{((X_{3D}, Y_{3D}, Z_{3D}), \text{classcode})_{ni} \mid ni=1, \dots, N_i\}$. If $N_i > 1$, these 3D location groups represent different facades of one building, which have different directions in reality. However, because the final 3D location groups are obtained from 2D images without z -axis information before 2D-to-3D transformation, they will have the same direction after transforming the coordinate system. Hence, these location groups need to be rotated in order to maintain their relative direction for 3D modeling. The relative directions, D , of these images have been provided by users in user interaction step. And the angle of rotation can be obtained using Equation (7).

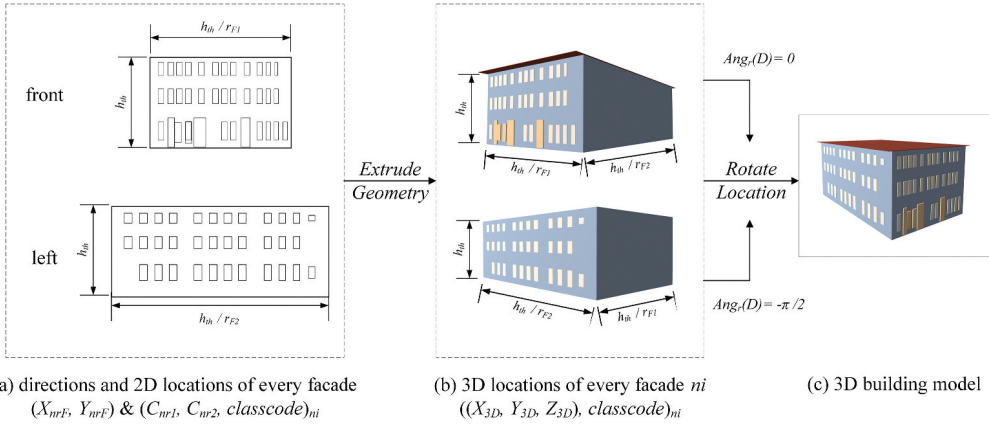


Figure 2. The process of 3D transformation. First, the 2D locations of every facade and its facade elements are extruded using *ExtrudeGeometry* method from Three.js, to obtain the corresponding 3D locations group of the facade in (b); Then, Every 3D locations group is rotated using Equations (7)~(8) based on the facade direction from user interaction step; Finally, the 3D building model is obtained.

$$Ang_r(D) = \begin{cases} 0 & D = front \\ -\pi/2 & D = left \\ +\pi/2 & D = right \\ \pi & D = back \end{cases} \quad (7)$$

In Three.js, it uses the right-handed coordinate system, and the y -axis is face-up. Hence, the rotation matrix M_r can be recorded as follows.

$$M_r = \begin{bmatrix} \cos Ang_r(D) & 0 & \sin Ang_r(D) \\ 0 & 1 & 0 \\ -\sin Ang_r(D) & 0 & \cos Ang_r(D) \end{bmatrix} \quad (8)$$

Then, the 3D location groups are rotated using Equation (9) and denoted as $\{((X_{3Dr}, Y_{3Dr}, Z_{3Dr}), classcode)\}$.

$$[X_{3Dr}^g \ Y_{3Dr}^g \ Z_{3Dr}^g]^T = M_r [X_{3D}^g \ Y_{3D}^g \ Z_{3D}^g]^T \quad (9)$$

$$g = 1, \dots, N_l$$

The process of 3D transformation is visualized in Figure 2.

The final 3D model of this building was built based on 3D locations after rotation. The semantic information of models was based on the class code of every element and was visualized using different colors.

3. Experiments

3.1. Experiment environment

We developed a web-based interactive platform. For the front-end, we used Hyper Text Markup Language (HTML), JavaScript (JS), Cascading Style Sheets (CSS), and bootstrap templates. The 3D models were rendered in the front-end using the 3D JavaScript library – Three.js. We used Python and a lightweight web application framework, Flask, to build our

algorithms and back-end. The convolutional neural network YOLOv3 was implemented with the open-source machine learning framework Pytorch (Paszke, Gross, Chintala, & Chanan, 2017). What's more, our platform was deployed on a server equipped with an Intel Core i7-8700 K CPU, 16 GB memory, and an NVIDIA GTX 1080Ti GPU.

3.2. Experiment results

In this subsection, we discuss the results of two key points in our interactive platform: the locations of all the elements and the 3D building modeling. The extraction results of facade elements and the results of 3D building modeling are shown in Section 3.2.1 and 3.2.2, respectively.

3.2.1. Extraction results of facade elements

In this paper, we used a pretrained model of YOLO v3 to extract windows, doors and balconies, which was trained on the Facade WHU dataset in Paris from Kong and Fan (2020). There are three subnetworks in Kong and Fan's work, and we only chose the window/door/balcony detection network. This model used Stochastic gradient descent (SGD), with momentum = 0.97 and weight decay = 0.00045 as the optimizer. The weighted multipart loss is also used as the loss function. This model was trained on a computer equipped with two NVIDIA 1080Ti GPUs. In the Facade WHU dataset, we only used the annotations of window, door and balcony to train the facade elements detection network, and it achieved state-of-the-art results: mAP = 71.6% and F1 score = 67.3%, where mAP is the mean average precision of objects from the three classes. We followed the standard definition of F1 score, where $F1\ score = 2 \times \frac{Precision \times Recall}{Precision + Recall}$. In the F1 score, precision = $\frac{TP}{TP+FP}$, and recall = $\frac{TP}{TP+FN}$. Here, TP, FP, TN, and FN mean true positive, false positive, true negative, and false negative, respectively. The visualization results are shown in Figure 3.

3.2.2. Results of 3D building modeling

The pretrained model for facade elements extraction used 850 images in Paris, France from the Facade WHU dataset as training and test data. Hence, to ensure that our workflow can work in different architectural styles, we chose five buildings from Heidelberg, Germany, and five buildings from Paris, France as test data. These images were from open-source VGI platforms Mapillary (Mapillary, 2020) and Flickr (Flickr, 2020). We used three metrics to evaluate the results of 3D building modeling. These three metrics are component precision (P_c), component recall (R_c), and component integrality (In_c), where P_c and R_c follow their standard definitions in computer vision, and In_c is defined by ourselves. TP_c , FP_c , TN_c , and FN_c are defined as true positive, false positive, true negative, and false negative of facade elements in every class.

$$P_c^k = TP_c^k / (TP_c^k + FP_c^k) \quad k = 1, 2, \dots, N_c \quad (10)$$

$$R_c^k = TP_c^k / (TP_c^k + FN_c^k) \quad k = 1, 2, \dots, N_c \quad (11)$$

where / is the division operator. The left side of the operator represents the number of corresponding objects, and the right side represents the total number of corresponding



Figure 3. Visualization results for extracting facade elements locations (Containing different views, lighting, and architectural styles).

objects. N_c is the number of classes. The number of facade elements in a building is usually less than 100, which means that a few false or missing detections will seriously affect the percentage of these three metrics. Hence, compared with the percent symbol, %, the division operator, /, can better represent the performance of 3D building modeling.

Unlike P_c and R_c for per-class evaluation, ln_c is designed for comprehensive evaluation. In 3D building modeling, we are more concerned with the total number of correct objects that were actually retrieved than the correct objects among the whole retrieved objects. Hence, we define ln_c as the ratio of the total number of the correct facade elements that were actually retrieved to the actual number of facade elements, which is similar to the definition of Recall. In the actual number of facade elements, the classes were not considered. ln_c is presented as follows.

$$ln_c = \sum_{k=1}^{N_c} TP_c^k / N_a \quad k = 1, 2, \dots, N_c \quad (12)$$

where N_c is the number of classes and N_a is the actual number of facade elements.

The facade elements that are obscured by trees, people, and other foreground were not calculated in P_c , R_c , and ln_c because the ground truth of these facade elements cannot be obtained. The statistical modeling results of the 10 buildings are shown in Table 1. The buildings from Heidelberg, Germany are numbered 1 to 5, and the buildings from Paris, France are numbered 6 to 10. From the table, we can see that the overall modeling integrality of our workflow ln_c is higher than 75% in various complex scenes. When facing two complex tasks of street-level 3D building modeling, (1) buildings with complex illumination and serious perspective distortion such as No. 2, 4, 7, 8, and 10, and (2) buildings with many facade

Table 1. Statistic modeling results of the 10 buildings.

Building No.	1		2		3		4		5	
	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c
window	19/21	19/21	18/19	18/19	18/19	18/18	10/12	10/10	66/66	66/68
balcony	3/3	3/4	0/0	0/1	-	-	-	-	-	-
door	-	-	1/1	1/1	1/1	1/1	1/1	1/2	2/5	2/2
In_c	22/25		19/21		19/19		11/12		68/70	
Building No.	6		7		8		9		10	
	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c	P_c	R_c
window	4/6	4/4	10/12	10/10	2/2	2/3	22/22	22/23	13/16	13/13
balcony	3/3	3/3	10/10	10/10	-	-	19/19	19/20	2/2	2/4
door	0/0	0/1	3/3	3/4	1/1	1/1	1/1	1/1	0/0	0/1
In_c	7/8		23/24		3/4		42/44		15/18	

elements such as No. 1, 2, 3, 5, and 9, our workflow can achieve stable 3D building modeling results with the In_c higher than 85%. At the same time, our workflow achieves impressive performance in the modeling of the key class. For the key class *window*, both of component precision P_c and recall R_c are higher than 67%. In the case of buildings with more than five windows, the P_c and R_c of window are even higher than 80%. Moreover, for the modeling results of buildings in Paris and Heidelberg, there are the similar In_c s of building models in

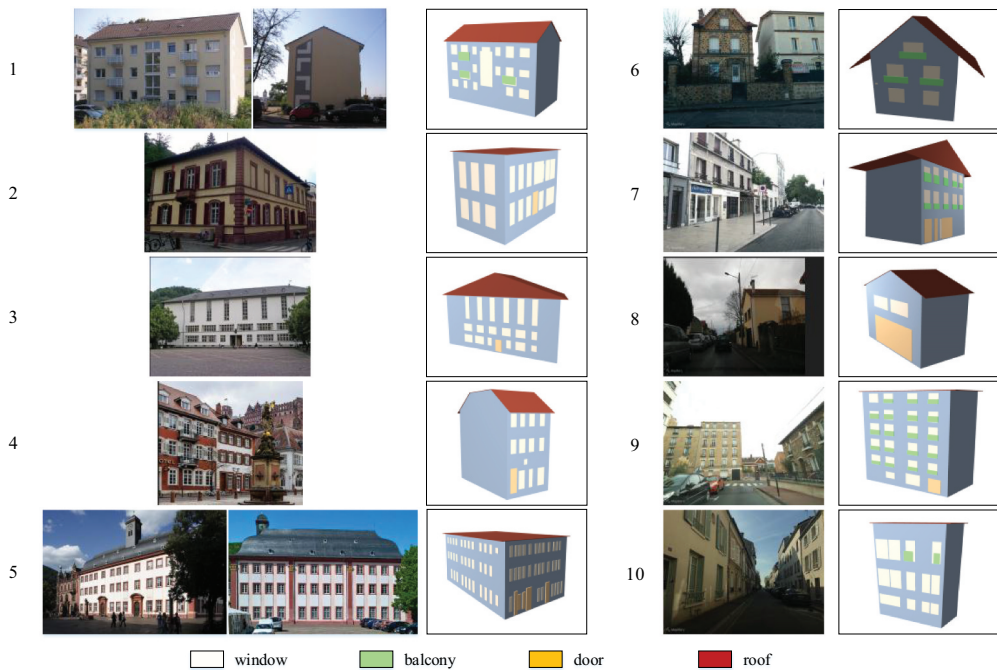


Figure 4. Qualitative result of 3D building modeling. The first and third columns are the original images of every building. The second and fourth columns are the 3D modeling results of every building through our workflow.

both cities. Hence, our workflow can be used for buildings with different architectural styles without extra work. The qualitative results of 3D building models are shown in Figure 4.

3.2.3. Performance of the platform

In addition to evaluating the performance of building extraction and modeling in our interactive platform, we tested the performance of our entire workflow. The modeling time of our workflow, T , was used for evaluation. Modeling time refers to the running time from when the user saves their drawing to the 3D building model shown on our website. Except for the time of facade elements extraction and 3D building modeling, the I/O time between the front-end and back-end was planned to be included to offer a more comprehensive evaluation. Given the different background of users, there usually is a big difference among them finishing user interaction, such as uploading images and drawing corresponding building facades. Hence, this time is not included in the modeling time T . Our platform was tested on a local area network (LAN). In our preliminary test, we found that the I/O time is less than 1 ms and can be ignored. The convolutional neural network takes most of the modeling time. Hence, in further tests, the I/O time was not taken into account, and the time taken by the convolutional neural network was tested in two different modes: GPU mode and CPU mode. In the GPU mode, we used an NVIDIA 1080Ti GPU to test the modeling time. The average modeling time in GPU mode $aT_G = 2.03$ s, and the maximum modeling time in GPU mode $mT_G = 6.62$ s. In the CPU mode, we used an Intel 8700 K CPU to test the modeling time. The average modeling time $aT_C = 15.35$ s, and the maximum modeling time $mT_C = 27.08$ s. The modeling time of our platform in the GPU mode is much faster than that in the CPU mode. Even in the CPU mode, our interactive platform can still achieve the 3D modeling of simple buildings in 30 s, which is faster than most 3D building modeling methods.

Figure 5 illustrates our *VG3D* platform. Users can upload images and draw facades of buildings using our website. The website for our platform is available at <https://18.210.26.42:5002/facade/>.

3.3. Usability testing

User interaction plays an important role in the interactive 3D modeling workflow. However, this is difficult to evaluate. Hence, we invited four students to experience and test our platform. Then user interaction module will be evaluated according to their feedback. Four students included two men and two women. One among them is an expert user of 3D city modeling, and the others are users focusing on other fields, such as spatial analysis. The usability testing didn't follow any additional guidance from our development team. They were only told the purpose of the platform and the functions of every button and every workspace. Fortunately, we received positive responses from all four participants. In terms of user interaction logic, all the participants, both men and women, considered the interface to be clear and concise. As for 3D modeling, our platform can model buildings at a rapid pace. The speed and completeness of the modeling were beyond their expectations. At the same time, the three participants without background in 3D city modeling were also able to construct 3D building models only with basic guidance as we mentioned before.



Figure 5. Results of our platform. (a) Interface of our platform. (b) Result of facade elements extraction, which is a middle part of our workflow. (c) Final result of 3D building modeling shown on our website.

4. Conclusions

In this study, we propose a new interactive platform, *VG3D*, for 3D building modeling. This platform is mainly composed of user interaction, automatic facade elements extraction, automatic roof inference, and real-time modeling of 3D buildings. Our platform only uses one or two images and simple user sketching as input. The extraction of facade elements and building modeling are automatically achieved by an object detection network and inference in 30 s, leading the platform to realize fast and complete urban 3D building modeling with lower labor and time costs. The 3D modeling performance of our platform is evaluated on images captured by various mobile phones and basic digital cameras, in which buildings have different architectural styles. Our experimental results demonstrate the capability of our platform for lightweight 3D building modeling. We also conducted a usability testing by inviting four students to try our platform and then give feedback. Their feedback indicates that our platform is easy to use and the interface is user friendly.

For further progress on 3D building modeling, we have released our platform on <https://18.210.26.42:5002/facade/>. In the future, we will further improve our platform to model complex buildings with more facades, such as shopping complexes and museums. Moreover, we will try to support the geographical matching of building models in the platform to make it easier to apply to urban 3D and other fields.

Disclosure statement

There are no conflicts of interest to declare.

ORCID

Hongchao Fan  <http://orcid.org/0000-0002-0051-7451>

Data Availability

The FacadeWHU dataset that supports this study is openly available at: <https://drive.google.com/drive/folders/1BeKXMuNAaRcwP6lSeiqFOSfscbP9ZKm7?usp=sharing>.

Funding

This work is supported by the National Natural Science Foundation of China (NSFC) under project [no. 41771484].

Notes on contributors



Hongchao Fan is professor for 3D Geoinformatics at the Department of Civil and Environmental Engineering at the Norwegian University of Science and Technology (NTNU). He received his master's degree in Geodesy and Geoinformatics at the University of Stuttgart and obtained his Ph.D. at the Technical University of Munich in Germany. After that he worked as Group Leader for 3D Data Infrastructure at the Heidelberg University for six years. In 2018, he started his work as professor at NTNU in Trondheim, Norway. His research interests include 3D city modeling, spatial data mining from VGI data and laser scanning.



Gefei Kong received the bachelor's degree in geographical condition monitoring from Wuhan University, Wuhan, China, in 2018, and is currently pursuing the master's degree with Wuhan University, Wuhan, China. Her research interests include geographical information engineering, computer vision, and 3D modelling, especially urban 3D modeling with their applications.



Chaoquan Zhang received the M.S. degree in software engineering from Nanjing University of Finance and Economics, China, in 2018. He is currently pursuing the Ph.D. degree with the Department of Civil and Environmental Engineering, Norwegian University of Science and Technology, Trondheim, Norway. His research interests include digital city 3D visualization, deep learning-based point cloud object detection and segmentation, and 3D digital modeling.

References

- Agarwal, S., Furukawa, Y., Snavely, N., Simon, I., Curless, B., Seitz, S. M., & Szeliski, R. (2011). Building rome in a day. *Communications of the ACM*, 54(10), 105–112.
- Arthur, D., & Vassilvitskii, S. (2007, January). k-means++: The advantages of careful seeding. In Proceedings of the eighteenth annual ACM-SIAM symposium on Discrete algorithms (pp. 1027–1035). New Orleans, LA: Society for Industrial and Applied Mathematics.
- Becker, S. (2009). Generation and application of rules for quality dependent facade reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(6), 640–653.
- Biljecki, F., Ledoux, H., & Stoter, J. (2016). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59, 25–37.
- Chen, T., Zhu, Z., Shamir, A., Hu, S.-M., & Cohen-Or, D. (2013). 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG)*, 32(6), 195.
- Dehbi, Y., & Plümer, L. (2011). Learning grammar rules of building parts from precise models and noisy observations. *ISPRS Journal of Photogrammetry and Remote Sensing*, 66(2), 166–176.
- Dirksen, J. (2015). *Learning Three. js—the JavaScript 3D Library for WebGL*. Birmingham, UK: Packt Publishing Ltd.
- Flickr. (2020). Retrieved from <https://www.flickr.com/>
- Gröger, G., Kolbe, T. H., Nagel, C., & Häfele, K. H. (2012). OGC city geography markup language (CityGML) encoding standard.
- Haala, N., & Kada, M. (2010). An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65(6), 570–580.
- Kada, M., & McKinley, L. (2009). 3D building reconstruction from LiDAR based on a cell decomposition approach. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 38(Part 3), W4.
- Kim, H., & Han, S. (2018). Interactive 3D building modeling method using panoramic image sequences and digital map. *Multimedia Tools and Applications*, 77(20), 27387–27404.
- Kong, G., & Fan, H. (2020). Enhanced facade parsing for street-level images using convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 1–13. doi:10.1109/TGRS.2020.3035878
- Li, Y., Ding, Z., Miao, W., Zhang, M., Li, W., & Ye, W. (2019, October). Low-cost 3D building modeling via image processing. In *2019 IEEE 10th International Conference on Software Engineering and Service Science (ICSESS)* (pp. 331–335). Beijing, China: IEEE.
- Ma, Y., Soatto, S., Kosecka, J., & Sastry, S. S. (2012). *An invitation to 3-d vision: From images to geometric models* (Vol. 26). New York, NY: Springer Science & Business Media.
- Mapillary. (2020). Retrieved from <https://www.mapillary.com/>
- Nishida, G., Garcia-Dorado, I., Aliaga, D. G., Benes, B., & Bousseau, A. (2016). Interactive sketching of urban procedural models. *ACM Transactions on Graphics (TOG)*, 35(4), 130.
- Paszke, A., Gross, S., Chintala, S., & Chanan, G., 2017. Pytorch: Tensors and dynamic neural networks in python with strong GPU acceleration, 6.
- Preka, D., & Doulamis, A. (2016). 3D building modeling in LoD2 using the CityGML standard. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42, 11–16.
- Redmon, J., & Farhadi, A. (2018). Yolov3: An incremental improvement. arXiv preprint arXiv:1804.02767.
- Sangiambut, S., & Sieber, R. (2016). The V in VGI: Citizens or civic data sources. *Urban Planning*, 1(2), 141–154.
- Verma, V., Kumar, R., & Hsu, S. (2006, June). 3d building detection and modeling from aerial lidar data. In *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)* (Vol. 2, pp. 2213–2220). New York, NY: IEEE.
- Wolberg, G., & Zokai, S. (2018). PhotoSketch: A photocentric urban 3D modeling system. *The Visual Computer*, 34(5), 605–616.
- Wu, B., Yu, B., Wu, Q., Yao, S., Zhao, F., Mao, W., & Wu, J. (2017). A graph-based approach for 3D building model reconstruction from airborne LiDAR point clouds. *Remote Sensing*, 9(1), 92.

- Yu, Q., Helmholz, P., & Belton, D. (2017). Semantically enhanced 3D building model reconstruction from terrestrial laser-scanning data. *Journal of Surveying Engineering*, 143(4), 04017015.
- Zheng, Y., Chen, X., Cheng, M. M., Zhou, K., Hu, S. M., & Mitra, N. J. (2012). Interactive images: Cuboid proxies for smart image manipulation. *ACM Transactions on Graphics*, 31(4), 99–100.

Research publications

PAPER 3

Automated detecting and placing road objects from street-level images

Chaoquan Zhang^a, Hongchao Fan^a, Wanzhi Li^b

^aDepartment of Civil and Environmental Engineering, Norwegian University of Science and Technology, Trondheim, Norway

^bSchool of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China


This paper is published in *Computational Urban Science*, August 2021; 1, 1-18.

Research publications



Automated detecting and placing road objects from street-level images



Chaoquan Zhang^{1*} , Hongchao Fan¹ and Wanzhi Li²

Abstract

Navigation services utilized by autonomous vehicles or ordinary users require the availability of detailed information about road-related objects and their geolocations, especially at road intersections. However, these road intersections are generally represented as point elements without detailed information, or are even not available in current versions of crowdsourced mapping databases including OpenStreetMap (OSM). This study proposes an approach to automatically detect road objects from street-level images and place them to correct locations according to urban rules. Our processing pipeline relies on two convolutional neural networks: the first one segments the images, while the second one detects and classifies the specific objects. Moreover, to locate the detected objects, we propose an attributed topological binary tree (ATBT) based on urban rules for each image in an image sequence to depict the coherent relations of topologies, attributes and semantics of the road objects. Then the ATBT is further matched with map features on OSM to determine the right placed location. The proposed method has been applied to a case study in Berlin, Germany. We validate the effectiveness of the proposed method on two object classes: traffic signs and traffic lights. Experimental results demonstrate that the proposed approach provides promising results in terms of completeness and positional accuracy.

Keywords: Object placing, Attributed topological binary tree, Street-level images, OpenStreetMap, Completeness, Traffic lights, Traffic signs

1 Introduction

The rapid development of advanced driver assistance systems and autonomous vehicles in recent years has attracted the ever-growing interest in smart traffic applications. Such intelligent applications can provide detailed road asset inventories of all stationary objects, such as street furniture (traffic lights and signs, various poles, bench, etc.), road information (lanes, edges, shoulders, etc.), small façade elements (antennas, cameras, etc.), and other minor landmarks. However, these detailed road map productions are mainly generated by mobile mapping systems (MMS), which require high costs both in the investment of equipment and in labor-intensive data post-processing. In addition, data updating is again a huge challenge. For instance, official road

maps suffer from a long update cycle that can last several months or even years (Kuntzsch et al., 2016).

The last decade has witnessed an explosion of geospatial data. An increasing number of crowdsourced geospatial data repositories/services allow volunteers to utilize information from various data sources when contributing data to a crowd-sourced platform. That is known as Volunteered Geographic Information (VGI) (Goodchild, 2007). Amongst them, OSM and Mapillary are the typical representatives of maps and street-level crowdsourcing platforms, respectively. The large amount of detailed map data provided by OSM not only enriches the data sources of map making, but also supports and promotes data-driven (Hachmann et al., 2018; Melnikov et al., 2016) and data-intensive (Chang et al., 2016; Gao et al., 2017) spatial analysis. Additionally, literature (Neis et al., 2012) has shown that OSM road data in Germany and Netherlands can be comparable to official data. With the introduction of Mapillary in 2014, it has

* Correspondence: chaoquan.zhang@ntnu.no

¹Department of Civil and Environmental Engineering, Norwegian University of Science and Technology, Trondheim, Norway

Full list of author information is available at the end of the article

become the biggest and the most active crowdsourced street-level imagery platform around the world. Tens of billions of street view images covering millions of kilometres of roads and depicting street scenes at regular intervals are available (Solem, 2017).

Even though OSM has made remarkable achievements, it still has some drawbacks. For example, road intersections in OSM are mainly represented as point elements without any semantic information (e.g. traffic signs/lights), or are even not available for most cities/countries (Fig. 1). According to Ibanez-Guzman et al. (2010), a high percentage of traffic accidents occur at road intersections, which reflects the importance of road intersections for traffic safety. If we can provide more information about intersections to the relevant authorities and let them use this information together with trajectory data to optimize the setting of traffic lights or vehicle speeds, that may help reduce the incidence of traffic accidents to some extent.

To the best of our knowledge, Mapillary submitted an additional layer to OSM where marked the traffic signs on the map. However, the locations of traffic signs in the layer differ greatly from the actual ones. Besides, multiple traffic signs with the same category would appear within a small area at the same time, which is obviously inconsistent with the actual situation. This may be related to the fact that Mapillary only adopted pure computer vision methods to detect the traffic signs without considering the correctness of their locations in the real world.

Considering all the above shortcomings, in this paper we aim to automatically detect and classify traffic lights/signs at road intersections by using deep learning method from street-level images, and localize their positions based on urban rules and proposed attributed topological binary trees (ATBT). In this way, we can further enrich the OSM data. Since these kinds of information are hard to be seen on satellite and aerial images

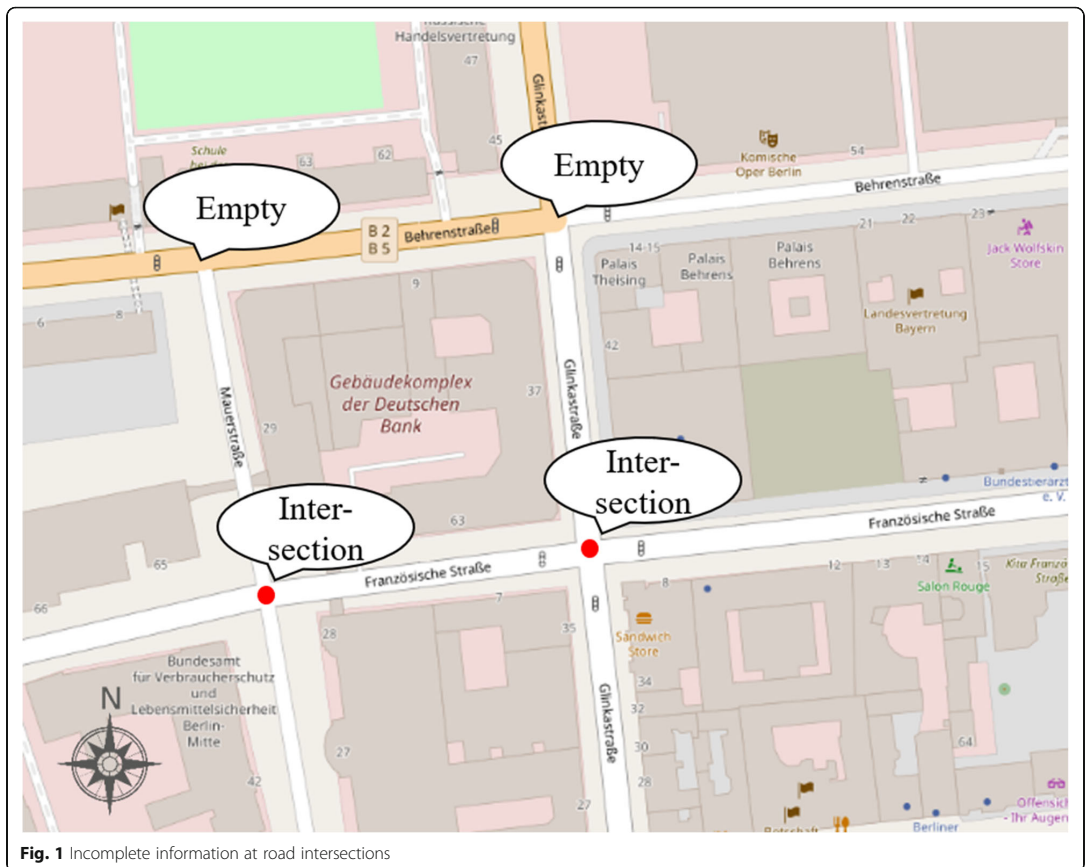


Fig. 1 Incomplete information at road intersections

and hence they cannot be mapped by volunteers on OSM, the proposed method provides a good solution for this issue. To summarize, the main contributions and innovations of our work are as follows:

- We propose a simple convolutional neural network, namely ShallowNet, for traffic sign classification, which is characterized by low model complexity, high detection accuracy and fast recognition speed.
- We propose 6 urban rules (or grammar) to assist the determination of the relative position and topological relationship of the road-related objects.
- Based on the urban rules, we propose an attributed topological binary tree (ATBT) for image sequences to effectively describe the coherent relations of topologies, attributes and semantics of the road objects.
- With the proposed ATBT, we can easily and accurately determine where the traffic lights and signs should be placed by matching with map features on OSM. Furthermore, our experiments show that the whole workflow performs well in terms of object completeness and positional accuracy.

The remainder of this paper is organized as follows. We first review some relevant state-of-the-art approaches in Section 2. Section 3 presents our complete detection and localization pipeline. A set of experimental analyses are presented in Section 4. Conclusions and future work are discussed at the end of this paper as Section 5.

2 Related work

Benefiting from the ubiquitous street view images accessible from Google Street View (GSV), Mapillary, etc., many efforts have been directed towards the intelligent use of them to assess urban greenery (Li et al., 2015; Li et al., 2018), to enhance existing maps with fine-grained segmentation categories (Mattyus et al., 2016), to explore urban morphologies by mapping the distribution of image locations (Crandall et al., 2009), to analyze the visual elements of urban space in terms of human perception (Zhang et al., 2018) and urban land use (Li et al., 2017). Furthermore, street view images have also been combined with aerial imagery to achieve tree detection/classification (Wegner et al., 2016), land use classification (Workman et al., 2017), and fine-grained road segmentation (Mattyus et al., 2016). Together with (Timofte & Van Gool, 2011), these methods rely on a simplified locally flat terrain model to evaluate object locations from street-level images.

The last ten years have witnessed the quick development of Convolutional Neural Network (CNN) and

CNN-based image content analysis. It has been proven efficient in learning feature representations from a large-scale dataset (LeCun et al., 2015). And as a consequence, urban studies involving street-level images have been largely enhanced since it was proposed. By leveraging street view images, many studies employ deep learning for object detection and classification, as well as image semantic segmentation to monitor neighbourhood change (Naik et al., 2017), to quantify the urban perception at a global scale (Dubey et al., 2016), to estimate demographic makeup (Gebu et al., 2017), to predict the perceived safety responses to images (Naik et al., 2014), to predict the socio-economic indicators (Arietta et al., 2014), and to navigate without maps in a city (Mirowski et al., 2018). In contrast, less attention has been paid to extracting traffic elements within road intersections from street view imagery. Furthermore, all of these methods use GSV as input data, but GSV charges a fee after downloading a certain amount for free, which is no doubt not a good choice for teams or individuals with insufficient research funds. Therefore, we introduce Mapillary, a fully free, crowdsourced, almost real-time updated and ubiquitous street-level imagery, into our work.

In terms of localization, so far, several approaches have been made available to map particular types of objects from street view imagery: traffic lights (Jensen et al., 2016; Trehard et al., 2014), road signs (Soheilian et al., 2013), and manholes (Timofte et al., 2011). These methods determine the positions of the road assets from individual camera views based on position triangulation. All of them depend heavily on various visual and geometrical features to match when multiple objects appear in the same scene. As a result, the performance of these methods is poor when multiple identical objects exist at the same time. Hence, an improved method is proposed. Hebbalaguppe et al. (2017) describe the problem as an object recognition task, and then adopt a stereo-vision (Seitz et al., 2006) approach to estimate the object coordinates from sensor plane coordinates using GSV. However, different from GSV, Mapillary street view images do not contain any camera intrinsics and projective transformation in their EXIF information, and thus we cannot perform the camera calibration. In other words, we cannot apply the same method for traffic lights/signs localization using Mapillary images. Recently, Krylov et al. (2018) combine the use of monocular depth estimation and triangulation to enable automatic mapping of complex scenes with the simultaneous presence of multiple, visually similar objects of interest, and achieve the position precision of approximately 2 m.

In this study, we focus on the research of road intersections to enrich the objects related to OSM intersections, such as traffic signs and lights, and to locate them

for the reference of autonomous driving or navigation. We propose a complete pipeline to extract scene elements such as buildings, sky, roads, sidewalks, traffic lights and signs based on image semantic segmentation from road intersections images. For localization purposes, the hierarchy of semantic objects needs to be applied, as there are the coherent relations of topologies, attributes and semantics of the road objects. In further, together with the segmentation results, an attributed topological binary tree (ATBT) based on urban rules can be established to depict the topologies among road objects. These are then matched with map features on OSM. In the end, road objects can be localized as promising results.

3 Methodology

In this section, we discuss a complete pipeline for the localization of traffic lights and signs from image sequences at road intersections. The pipeline has the following three modules: (1) data preprocessing and cleaning module; (2) object segmentation and recognition module; (3) localization module. Figure 2 depicts the whole framework. The first module is for preparing preprocessed and cleaned data for the next two modules (see Section 3.1). The second module mainly extracts road-related information by using image semantic segmentation as well as object detection and classification (see Section 3.2). In the last module, an attributed topological binary tree (ATBT) is constructed to represent the relative position relation between extracted objects at the intersections and to locate the objects with urban rules (see Section 3.3). Ultimately, the located objects can be integrated to enrich the OSM data.

3.1 Data preprocessing and cleaning

The main purpose of the data module is to prepare data for the next two modules. First of all, the intersections are identified by using the DBSCAN algorithm (Ester et al., 1996) based on incomplete traffic lights existing in

OSM data. The incompleteness is reflected in, for example, there should have had four traffic lights at the intersection but only one or two are marked in the OSM. Additionally, their positions are roughly estimated which means traffic lights are with lower positional accuracy. After identifying the intersections, selecting experimented intersections mainly obeyed two rules: (1) the intersections can be clearly seen in the Mapillary images; (2) the image sequences can be corrected well by employing the SfM algorithm. Second, all the available images can be downloaded by querying the relevant Mapillary application programming interface (API). Specifically, a bounding box in geographic coordinates that covered the entire Berlin has been calculated in advance. To form a request to the Mapillary API, we then construct an API URL in which includes a common server address, user’s unique client ID, bounding box and other search parameters such as start time, end time, searching radius, etc. Third, a buffer is then set up for each road intersection to extract image sequences within the buffer. An image sequence refers to a trajectory of a user traveling along the street. For an intersection with four road branches, we are able to theoretically build four image sequences by merging multiple image sequences according to their geolocations because of four kinds of rough driving directions, i.e. west-east, east-west, south-north and north-south. In addition, camera location including latitude and longitude, and camera angle are extracted.

Furthermore, we have found that the GPS positions of image sequences often drift, which may be associated with the geographical environment during the shooting (for example, tall buildings or heavy tree canopies block the GPS signal), or it may be because the GPS receiver built in the camera itself is inaccurate. Fortunately, one of the big advantages of Mapillary is that street view images of the same road segment may be uploaded repeatedly by different volunteers. And there is a certain degree of overlap between the two adjacent images,

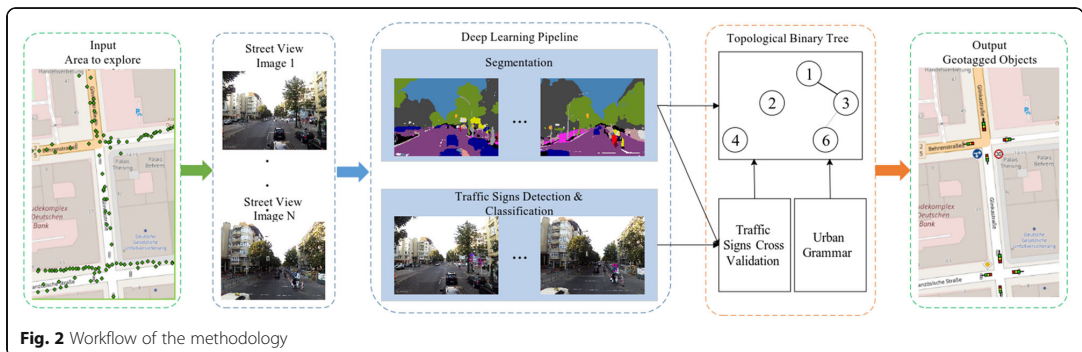


Fig. 2 Workflow of the methodology

which makes it possible for us to correct their shooting positions.

To reduce the error as much as possible and to improve the accuracy of localization, we employ a technique called Structure from Motion (SfM) (Snavely et al., 2008), as depicted in Fig. 3(a), to match features between images and reconstruct their surroundings in three-dimensional space to form point clouds. Each point has its position in three-dimensional space, so we can estimate the correct shooting positions of images along with the camera angles. As a result, these corrections can place misaligned images in their original positions as much as possible. In general, the more images feeding into the system from an area, the better the results could be. An SfM-corrected sequence is shown in Fig. 3(b). We can easily discover the original image shooting locations (green dots) swinging from one side of the road to the other in an “S” pose. Red dots symbolize the corrected locations, which now are fully aligned with roads. Additionally, if there are many overlaps between those images, these corrections can be very promising.

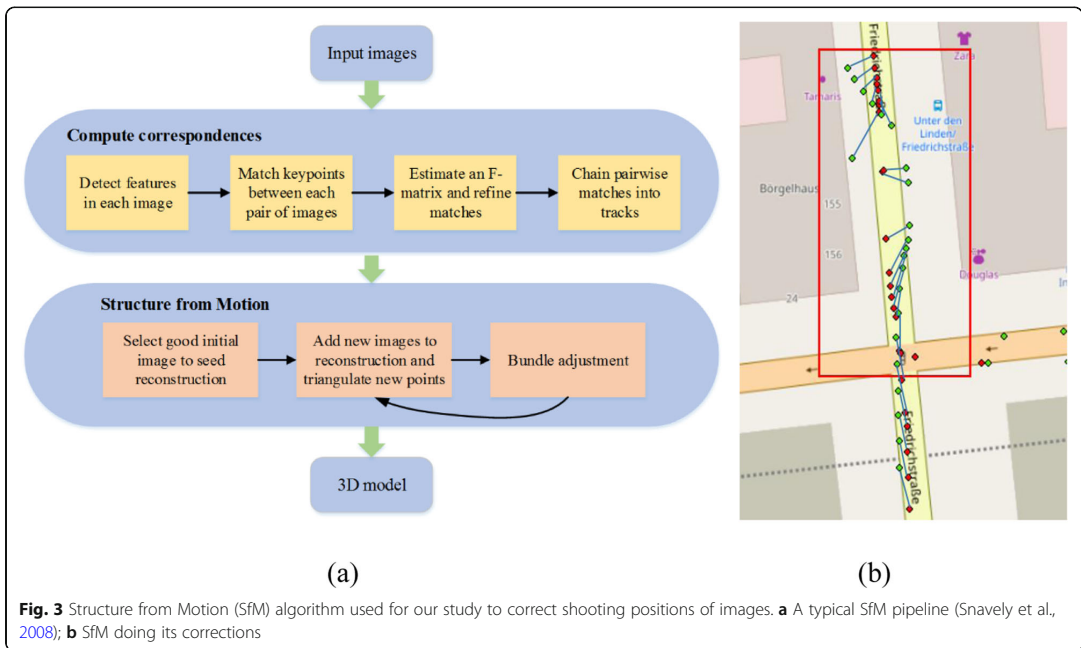
3.2 Object segmentation and recognition using deep learning

In theory, all road-related information can be extracted accurately from images only via semantic segmentation

(see Section 3.2.1). Nevertheless, the quality of crowd-sourced street-level images varies greatly, and hence it is difficult to ensure that all images can be segmented well, which would lead to inaccuracies or errors. Hence, in Section 3.2.2, we adopt an alternative strategy based on object detection to improve this problem.

3.2.1 Semantic segmentation using PSPNet

Image semantic segmentation is one of the key techniques used to understand a scene (Zhou et al., 2017), and is aimed at segmenting and recognizing object instances from images. Given an input image, the model can assign a class label for each pixel. One of the state-of-the-art semantic segmentation models with superior performance – PSPNet (Zhao et al., 2017) is applied in our study to perform object extraction. The PSPNet uses a new neural network sub-architecture, which retains global and local contextual information through a multi-scale representation of the previous convolutional layer’s output. Because of the validated performance of the PSPNet trained on the PASCAL VOC 2012 (Everingham et al., 2010) and Cityscapes (Cordts et al., 2016) datasets, we are confident to segment road-related objects well by using PSPNet, such as buildings, sky, roads, sidewalks, traffic lights/signs, etc. These extracted objects will later be used as nodes of the attributed topological binary tree (ATBT).



3.2.2 Object detection and classification using YOLOv3 and ShallowNet

After image semantic segmentation, we find that there are three limitations associated with semantic segmentation, especially for traffic signs. First, since we can only know this is a traffic sign through semantic segmentation, but we do not know which kind of traffic sign it belongs to. Second, if two signs are arranged together, semantic segmentation cannot identify them separately, which is not conducive to the supplement and enrichment of OSM semantic data. Third, our PSPNet model often misclassifies the isolation piles as traffic signs, or sometimes confuses two objects with similar features but they actually do not belong to the same category. The third limitation may be because the two objects have similar features (such as color, shape or texture), or the training dataset does not include such cases. As a result, the model did not learn the relevant features well.

Fortunately, object detection can address the above limitations. Taking into account the processing speed and detection accuracy, we choose YOLOv3 (Redmon & Farhadi, 2018) as our object detection model after some researches. Thus, we specially train a YOLOv3 model based on GTSDb (Stallkamp et al., 2012) dataset for detecting traffic signs, and then cross-validate the results of object detection and semantic segmentation to reduce errors and provide rich and effective attribute information for localization.

In our study, we not only need to know this is a traffic sign, but also need to know which kind of sign it belongs to. Consequently, in terms of traffic sign classification, we design a new shallow convolutional neural network called ShallowNet. As illustrated in Fig. 4, the network contains only five layers with weights; the first three are convolutional and the remaining two are fully-connected. The output of the last fully-connected layer is fed to a 45-way softmax which produces a distribution over the 45 class labels. We adopt batch normalization (Ioffe & Szegedy, 2015) right after each convolution and before ReLU non-linearity (Nair & Hinton, 2010) to

speed up the convergence of model training. Additionally, to reduce the size of feature maps as much as possible, the convolutional layers involved in the network are all performed filling operation, and each convolution is followed by downsampling.

The first convolutional layer filters the $48 \times 48 \times 3$ input image with 64 kernels of size $7 \times 7 \times 3$. The second convolutional layer filters the output of the previous layer with 128 kernels of size $4 \times 4 \times 64$. The third convolutional layer has 300 kernels of size $4 \times 4 \times 128$ connected to the outputs of the second convolutional layer. Then we expand the feature map and form 1500 feature vectors into the fully-connected layer. Moreover, to reduce the overfitting of the network, we introduce dropout (Hinton et al., 2012) at the first fully-connected layer.

In general, our proposed network model, ShallowNet, is characterized by:

- Simple network structure and low model complexity. With few parameters, it is easy to be deployed to mobile or embedded devices.
- High accuracy. It can correctly recognize the type of traffic signs
- Fast recognition speed. Real-time object recognition can be achieved.

3.3 Object localization

Since Mapillary street view images do not contain any camera intrinsics in their EXIF information, it is impossible to calculate the projective transformation matrix and then perform camera calibration. In other words, we cannot apply photogrammetry methods for traffic lights/signs localization using Mapillary images.

After observing a large number of images of road intersections, we note that many images show a structure where buildings are on both sides of the road and a portion of the sky appears between them, traffic lights and signs being often placed at street corners, as well as pedestrians and vehicles appearing on the road. We can

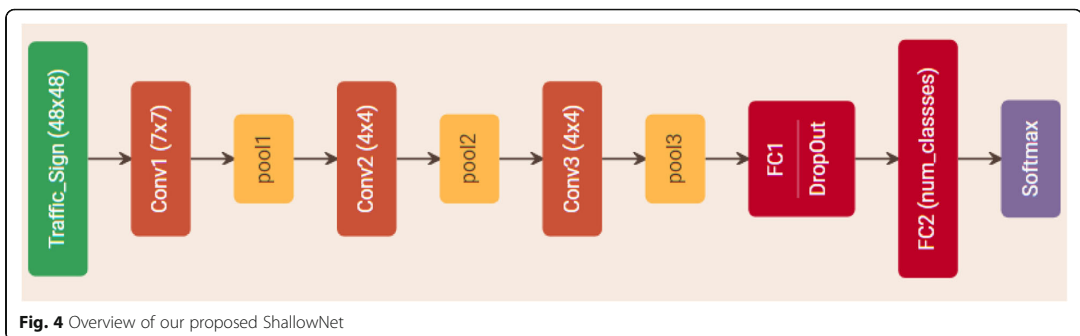


Fig. 4 Overview of our proposed ShallowNet

vaguely feel that there exist some certain arrangement rules between the objects in the images. Inspired by this, we propose a novel method to depict the coherent relations of topologies, attributes and semantics of the road objects at the intersections by establishing an attributed topological binary tree based on urban grammar (see Section 3.3.1). These objects (mainly traffic lights and signs) are then further matched with map features on OSM to determine the correctly placed location (see Section 3.3.2).

3.3.1 Attributed topological binary tree (ATBT) generation and updating

Taking the extracted objects through the image semantic segmentation as input, the ATBT can be created from top to bottom and from left to right. The left and right children of the binary tree can reflect the relative position relationship between the objects. We regard traffic lights, traffic signs and sidewalks as three types of nodes of the tree, and assign corresponding attributes to each type of node, such as centroid, area, height (optional), category, and role in the tree.

For traffic lights, there are two types of traffic lights: located on the sidewalk (low one) and located on the road (high one), which need to be recognized through following two urban rules (see Fig. 5):

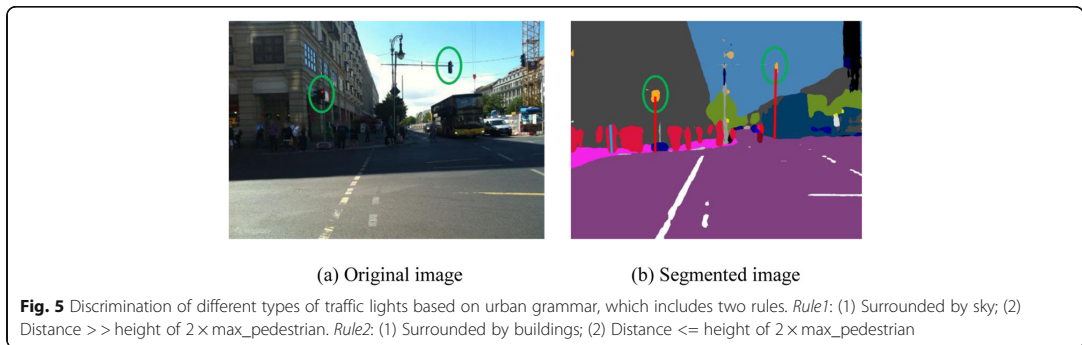
- (1). If the traffic light is surrounded by the sky, a ray (right red solid line in Fig. 5(b)) can be cast from the centroid of the segmented traffic light region downwards the road. If the distance between centroid and road surface is far more than twice the height of the tallest pedestrian (blue solid line in Fig. 5(b)), it can be inferred that this traffic light is at the road junction (i.e. high one), and its height is about 7 m (*another urban rule, searched from the Internet*).
- (2). If the traffic light is surrounded by the buildings, a similar ray (left red solid line in Fig. 5(b)) can be cast from the centroid of the segmented traffic light

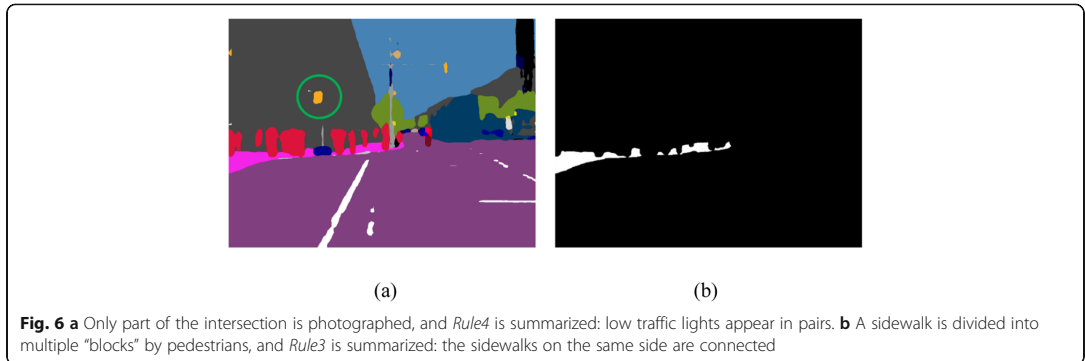
region downwards the sidewalk. If the distance between centroid and sidewalk surface is less than or equal to twice the height of the tallest pedestrian, it can be inferred that this traffic light is located on the sidewalk (i.e. low one), and its height is about 4 m.

In fact, *Rule1* implies an “up and down” relationship, that is, traffic lights are surrounded by the sky and the sky is above the high traffic lights. Similar to *Rule1*, *Rule2* also implies a “front and back” relationship, that is, the low traffic light is surrounded by buildings and buildings are behind the low traffic light.

As we can see from *Rule2*, sidewalks are very important for our judgment. But in many cases, sidewalks are divided into multiple independent “blocks” by the pedestrian (as shown in Fig. 6(b)) according to the results of image semantic segmentation. In this case, it is necessary to judge whether the adjacent independent “sidewalk blocks” meet a certain distance threshold based on another empirical knowledge (i.e. the sidewalks on the same side are connected, *Rule3*). If within this distance threshold, they are considered to be connected. Furthermore, many images do not capture the view of the whole intersections, but just a part of them as shown in Fig. 6(a). According to the urban rules, low traffic lights on the sidewalks tend to appear in pairs (*Rule4*). As long as there is a low traffic light on one side of the road, there definitely has another one on the other side of the road. This gives our topological binary tree the ability to reason.

Of course, there are also urban rules applicable to traffic signs. Since the study area of this paper is Berlin, Germany, we find that traffic signs at road intersections follow such patterns (*Rule5*, see Fig. 7): they either appear alone, or are usually close to the low traffic light above or both up and down, or arrange together. These are intrinsic combination patterns, and the distance between centroids of the internal objects of the combined pattern is within a small threshold.





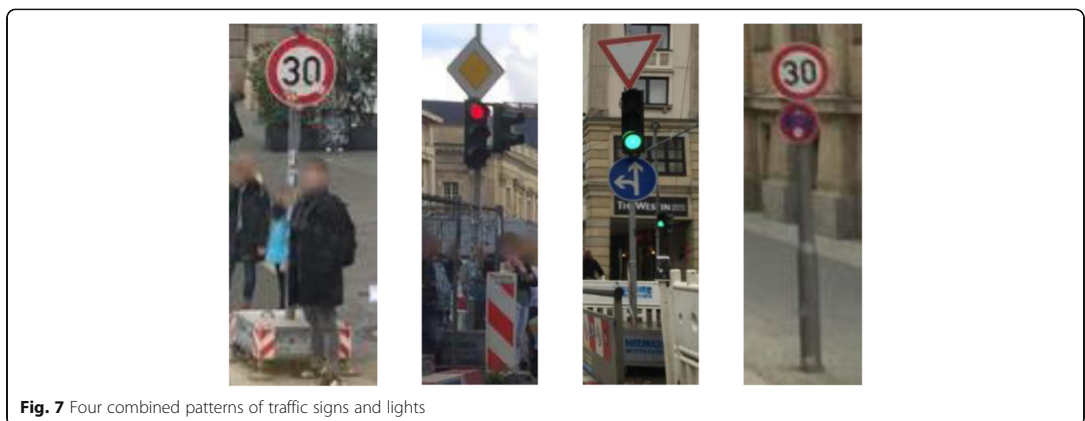
Finally, for each image in the image sequence, an ATBT can be established according to the results both from semantic segmentation and traffic sign detection/classification. The left subtree of the root node corresponds to the left side of the road, and the right subtree corresponds to the right side of the road. Additionally, for the convenience of computation, the node number of the tree is strictly in accordance with the node number of the complete binary tree. The left-right or top-bottom relationship between nodes is determined by the position of their centroids.

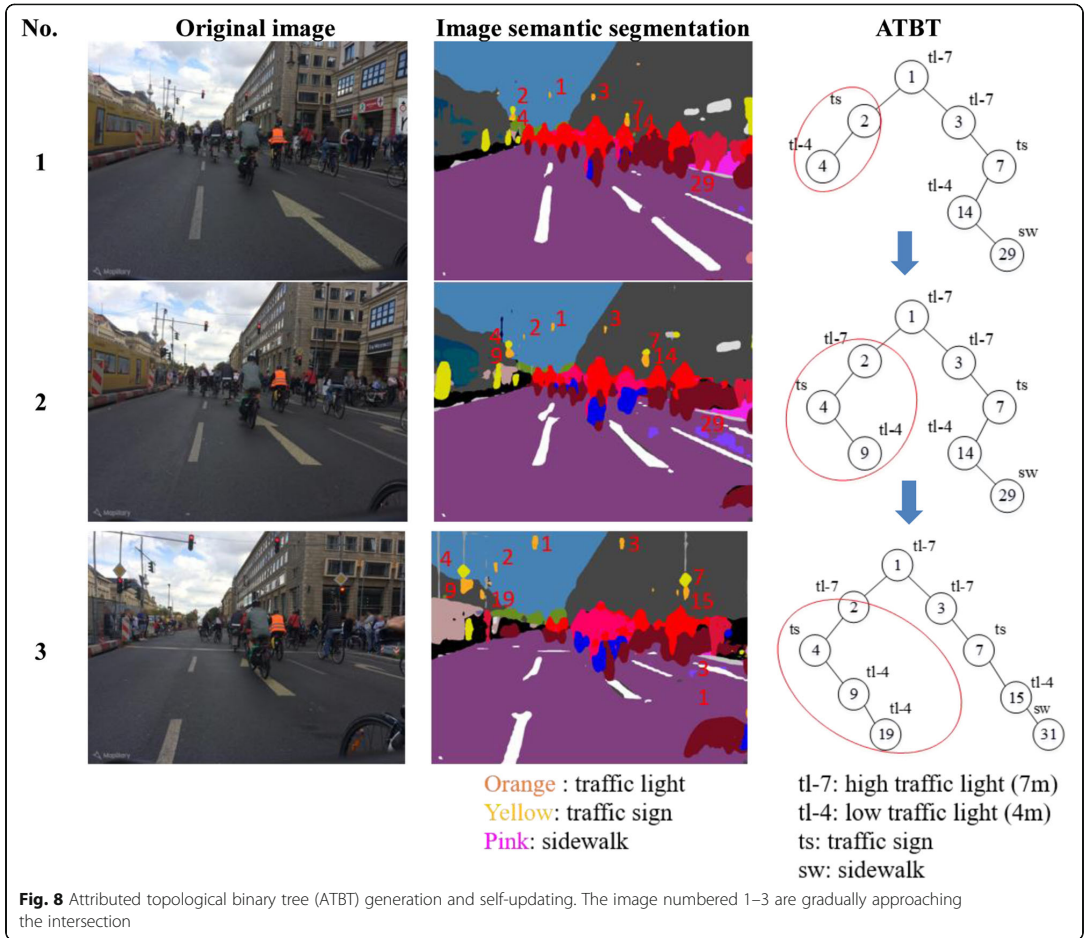
However, the first image of an image sequence was generally taken at the farthest location from the intersection, which may lead to some road objects not being segmented/detected and then affect the establishment of ATBT. Therefore, with the camera approaching the intersection, the image scene becomes clearer and can capture additional objects that are missed in previous images. Then, ATBT would dynamically update itself by comparing the difference

between the current tree and the previous tree as well as combining with the urban rules, as illustrated in Fig. 8. The image numbered 1–3 are gradually approaching the intersection.

Meanwhile, the scene depth information is also considered during the ATBT establishment and updating. For example, there are two low traffic lights (labelled by No. 9 and 19 in the third segmented image) as shown in Fig. 8. They belong to a pair as described in *Rule4* (i.e. *low traffic lights on the sidewalks tend to appear in pairs*). Their height should be the exactly same in reality, however, from the image, the No.19’s height is obviously lower than the No.9’s. This is because photo imaging follows the law that the object is big when near and small when far. Hence, in this case, we used the scene depth information and *Rule4* to infer that these two traffic lights should belong to a pair and should locate on the sidewalk.

In summary, as the image gets closer to the intersection, ATBT can dynamically update itself according to





the topological relationship of the objects, urban rules as well as scene depth information, and get the final ATBT.

3.3.2 Map matching

Based on the ATBT constructed earlier, we can use the shooting positions and camera angles provided by images as well as OSM footprints that are located around the intersections to match the left and right subtrees of the ATBT with the corresponding footprints. After that, the geographically placed locations of objects (e.g. traffic signs) in the real world can be determined.

Assuming there is an image sequence taken from west to east, the shooting positions of these images are represented by C1, C2 and C3 (as demonstrated in Fig. 9). Here, C1 is illustrated as an example. We take the red shooting point C1 as the centre of a circle, and draw the buffer with a radius of 26 m (determined by multiple

experiments) to get footprints intersecting with the buffer. After calculating the distance from footprints to C1, we get that the yellow highlighted footprint is closest to the C1 (i.e. it corresponds to the right subtree of the ATBT), and similarly, the green highlighted footprint is closest to the C1 (i.e. it corresponds to the left subtree of the ATBT). From Fig. 9, the yellow and green highlighted footprints are indeed at the intersection, which indicates that the results we got are correct. In this way, the placed positions of traffic signs and lights can be determined.

We have inquired about the “Code for Urban Road Design”, which clearly states that the minimum width of an ordinary sidewalk is 2~3 m (**Rule6**). Therefore, we place the low traffic lights and traffic signs about 2.5 m away from the corresponding footprint corner point (A1 or A2); the high traffic lights are placed at the midpoint

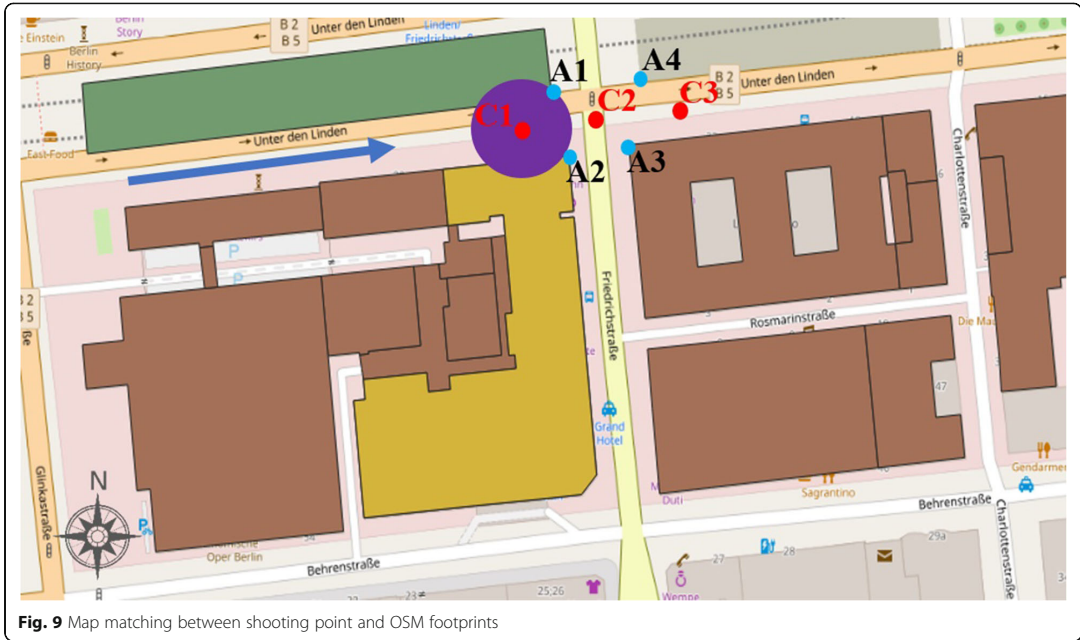


Fig. 9 Map matching between shooting point and OSM footprints

of connection between A1 and A2. In fact, this is not a precise localization, but it can indicate the approximate location of the traffic lights and signs.

The situation of C2 is a little bit complex. Since C2 is located in the middle of the intersection and none of the footprints is around it. If C2 is the centre of a circle and the corner points obtained by intersecting with footprints are A1 and A2, it indicates that C2 just passed one side of the intersection. Since the content of an image is always the scene in front of C2, but at this time A1 and A2 are behind the C2, so these two corners are not the corner points we want. Similarly, if the circle with C2 as centre intersects with footprints and yields A3 and A4 as their corner points, which are what we want because they are in front of C2.

Compared with the situation of C2, the situation of C3 is much simpler. Since C3 is about to leave the intersection area, the corner points that C3 intersecting with footprints are always behind it. This situation is not what we want as well.

For a more intuitive view of the urban rules used in this paper, we summarize and list them in Table 1 as shown below.

4 Experimental results

4.1 Study area and data

As the capital and largest city of Germany, Berlin was chosen as our study area. The study area has various intersection types, which range from the most common

Table 1 Summarized urban rules used in attributed topological binary trees (ATBT)

Rule No.	Description of the urban rules
Rule1	1) Traffic light is surrounded by sky; 2) distance between the traffic light and road surface is far more than twice the height of the tallest pedestrian. Conclusion: high traffic lights
Rule2	1) Traffic light is surrounded by buildings; 2) distance between the traffic light and road surface is less than or equal to twice the height of the tallest pedestrian. Conclusion: low traffic lights
Rule3	The sidewalks on the same side are connected.
Rule4	Low traffic lights on the sidewalks tend to appear in pairs.
Rule5	Traffic signs in Germany either appear alone, or are usually close to the low traffic light above or both up and down, or arrange together.
Rule6	The minimum width of an ordinary sidewalk is 2 ~ 3 m.

intersections with three/four road branches to the complicated intersections, like roundabouts.

The datasets used in this study include OSM building footprints data, Mapillary street view images, Mapillary Vistas, German Traffic Sign Detection Benchmark (GTSDB), and German Traffic Sign Recognition Benchmark (GTSRB) (Stallkamp et al., 2011). The OSM building footprints data was collected from Geofabrik. The Mapillary street view images were downloaded via querying Mapillary APIs including the metadata of each image, from 2014 to 2018. To facilitate further study, we only extracted images located in the intersection buffer. Mapillary Vistas was from Neuhold et al. (2017), which contains 25,000 high-resolution images annotated into 66 object categories. They are used as the training set for the semantic segmentation model—PSPNet. Last but not the least, GTSDB and GTSRB were from Stallkamp et al. (2011, 2012), and are applied for training object detection model—YOLOv3 and proposed object classification model—ShallowNet, respectively.

In summary, above all are reasons why we choose Berlin as our study area. Fig. 10 depicts the example area of Berlin as well as the distribution of Mapillary street view camera locations and OSM building footprints.

4.2 Extraction of road-related objects with PSPNet

4.2.1 Training

For the segmentation task, our implementation is based on the public framework TensorFlow. Like the Zhao et al. (2017), we also use the “poly” learning rate policy (the learning rate is multiplied by $(1 - \frac{iter}{max_iter})^{power}$). We set the base learning rate to 0.01 and power to 0.9. The training is performed on three NVIDIA GTX 1080Ti GPUs using stochastic gradient descent (SGD) with momentum $m = 0.99$ and weight decay = 0.0001. Due to limited physical memory on GPU cards, we set the “batchsize” to 4 for each GPU card during training. In addition, we crop the Mapillary training images to a size of 720×720 , and start with a pre-trained ResNet34 (He et al., 2016) model with the dilated network strategy (Yu & Koltun, 2015) to extract the feature map. For data augmentation, we adopt random mirror, rotations $[-5^\circ, 5^\circ]$, random resize between 0.5 and 2, and small enhancements in the image’s color, sharpness, and brightness for Mapillary Vistas. This comprehensive data augmentation scheme makes the network resist overfitting.

4.2.2 Evaluation and comparison

The performance on Mapillary street-level images was evaluated with PSPNet. Figure 11 shows several

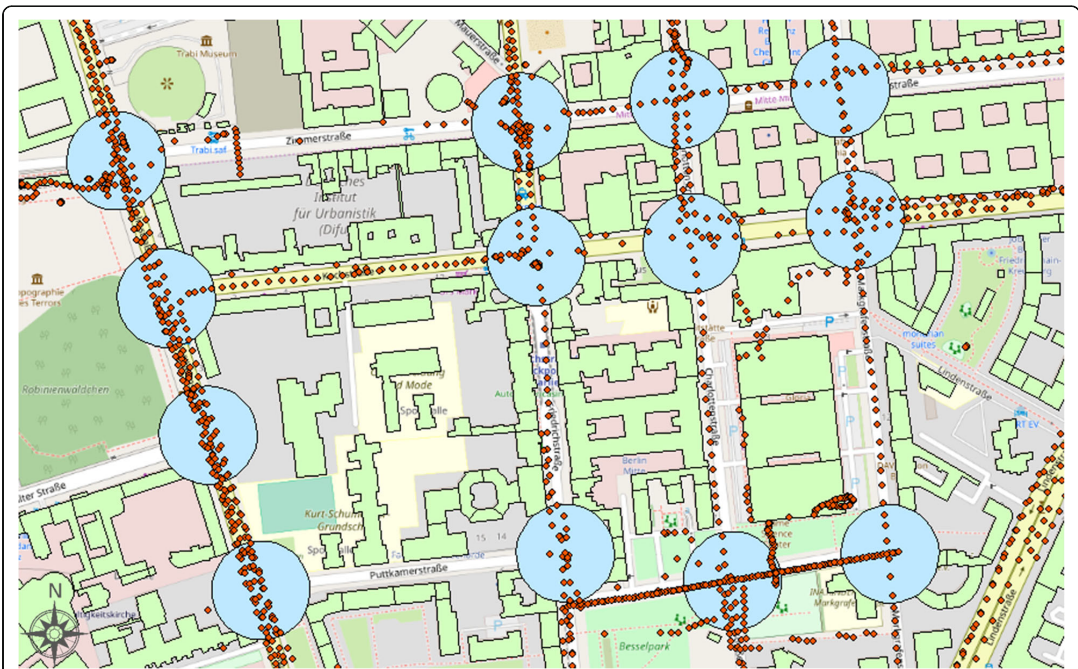


Fig. 10 Example area of Berlin (map:© OpenStreetMap contributors). Red dots, green polygons, and blue circles are the Mapillary street view camera locations, OSM building footprints, and intersection buffers, respectively

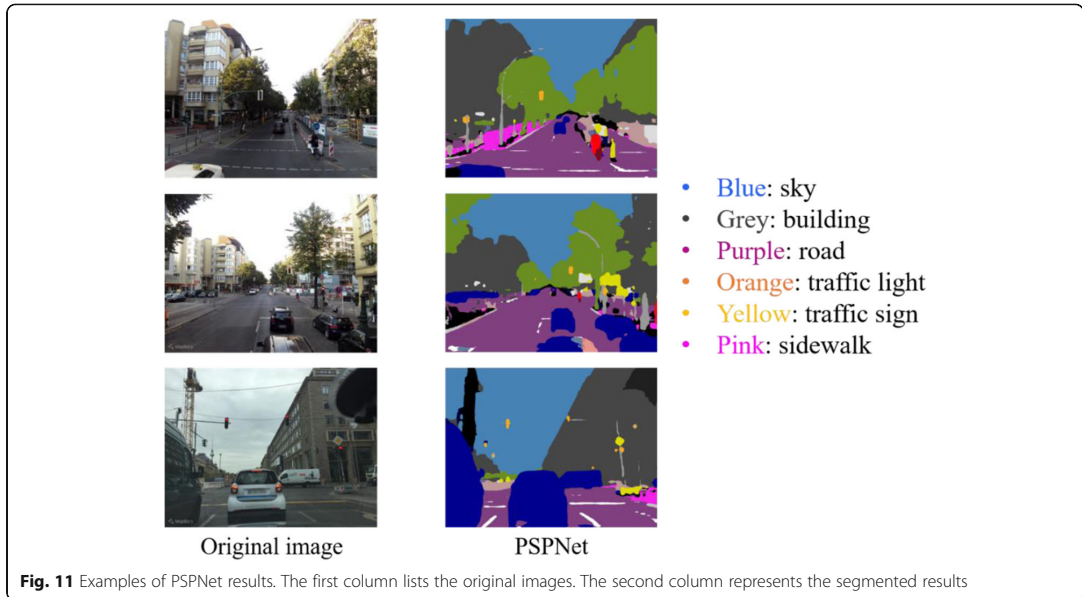


Fig. 11 Examples of PSPNet results. The first column lists the original images. The second column represents the segmented results

segmented examples, where the first column represents the sample images located at the intersections, the second column corresponds to the segmented results. From the segmentation results, the PSPNet model we trained can well segment the sky, buildings, roads, traffic signs and other objects that we want. Furthermore, to prove the superiority of our PSPNet model, a comparison test is conducted with the state-of-the-art model, DeepLabv3+ (Chen et al., 2018). In Table 2, our trained PSPNet model achieves Mean IoU 34.17% and Pixel Acc. 91.3%, and both of them outperform the DeepLabv3+.

4.3 Detection and classification of traffic signs

In this subsection, to prove the superiority of our used or proposed network, we will conduct a series of comparative experiments on detection network YOLOv3 and the classification network ShallowNet.

4.3.1 Traffic signs detection

4.3.1.1 Training Due to some differences between the street view at intersections and the ordinary street

view, we add 300 extra annotated Mapillary images at road intersections into the GTSDB dataset to form a hybrid dataset. The dataset is divided into 750/450 images for training and testing. We train the YOLOv3 with Darknet on an NVIDIA GTX 1080Ti GPU card, and set “batchsize” to 8. The warm-up strategy is adopted in the training phase, i.e. starting with a very small learning rate at the beginning of training. As the number of iterations increases, the initial learning rate gradually increases to 0.001. Starting from the second epoch, the normal gradient descent is made with 0.001 as the initial learning rate. Meanwhile, to augment the data, we use rotations $[-5^\circ, 5^\circ]$, random flipping, random scale $[20, 200]$, and color space conversion.

4.3.1.2 Evaluation and comparison To prove that our trained YOLOv3 model is excellent at both processing speed and detection accuracy, we compare YOLOv3 with the previous best-performing method (Faster R-CNN (Ren et al., 2015)) on the testing set. In Table 3 our trained YOLOv3 model yields mAP (mean Average Precision) 94.7% and sec/img (second per image) 0.025 s, and both of them outperform the Faster R-CNN. The detection speed of approximately 30FPS is much faster than two-stage detector like Faster R-CNN. In addition, the performance of traffic sign detection on Mapillary street-level images is evaluated with YOLOv3. Figure 12 shows several example results.

Table 2 Comparison of mIoU and pixel accuracy between our trained PSPNet and DeepLabv3+

Method	Mean IoU(%)	Pixel Acc.(%)
PSPNet	34.17	91.3
DeepLabv3+	33.97	90.2

Table 3 Comparison of mAP and detection time between our trained YOLOv3 and Faster R-CNN on the GTSDB + Mapillary images hybrid testing set

Method	Input size	mAP(%)	Model size(M)	Sec/img(s)
YOLOv3(Darknet-53)	608 × 608	94.7	246.4	0.025
Faster R-CNN (ResNet)	1280 × 720	90.5	267	0.230

4.3.2 Traffic signs classification

4.3.2.1 Training The public GTSRB dataset contains only 43 types of traffic signs, but it does not cover signs that often appear at intersections. Hence, we add two more categories to reach 45 categories in total. The dataset is divided into 75 K/12 K images for training and testing. Due to the uneven number of different categories of traffic signs, we also use a data augmentation technique during training, which includes histogram equalization of color images, affine transformation, contrast enhancement, Gaussian blur, Gaussian random noise, color space conversion, and random inactivation of pixel values. The training is performed on an NVIDIA GTX 1080Ti GPU using Adam Optimizer with Cross Entropy Loss Function.

4.3.2.2 Ablation study for ShallowNet To evaluate ShallowNet, we conduct experiments with several settings, including batch normalization (BN), dropout, and data augmentation. As listed in Table 4, the accuracy of manual recognition is 98.84%. Although the accuracy of manual recognition is very high, the automation degree is low, which is not conducive to information extraction. For the simplest ShallowNet (only convolution, pooling and full connection operation), the test accuracy on GTSRB is 95.89%. While it does not work better than manual recognition, it has higher automation degree and faster forward propagation speed (it only takes 3.6 ms on average to detect an image in CPU mode).

Even though the ShallowNet structure is very simple, the number of neurons in the fully-connected layer is large, which may lead to overfitting to some extent. Hence, we introduce dropout at the first fully-connected layer and successfully increase accuracy by nearly 1.6%. Besides, batch normalization is adopted in ShallowNet_Drop to reduce the difference in the distribution of original data, and to help speed up the convergence of training. ShallowNet_BN_Drop has a similar performance to manual recognition. Finally, we explore whether data augmentation improves the accuracy of the model or not, and augment the data on ShallowNet_BN_Drop. It achieves the accuracy of 99.52% on the testing set, which surpasses the accuracy of manual recognition, and increases by over 1% compared to ShallowNet_BN_Drop. Through this experiment, it can be proved that data augmentation is very critical to improve the accuracy of the model. Figure 13 shows several examples.

4.4 Localization of traffic lights and signs

In this subsection, we apply the method introduced in Section 3.3 for locating the traffic signs and lights based on ATBT and urban rules. The experimented image sequences are merged from multiple image sequences according to their geolocations and meanwhile, misaligned images are corrected using Structure from Motion (SfM). Each image sequence refers to a trajectory of a volunteer user traveling along the road; and, over time, the same road segment may be covered by multiple sequences that are uploaded by different volunteers. One

**Fig. 12** Examples of traffic sign detection results based on YOLOv3

Table 4 Investigation of ShallowNet with different settings. ‘Drop’, ‘BN’ and ‘Aug’ represent dropout, batch normalization and data augmentation, respectively

Method	Accuracy(%)	Sec/img (ms)
Human performance	98.84	/
ShallowNet	95.89	3.6
ShallowNet_Drop	97.47	/
ShallowNet_BN_Drop	98.49	/
ShallowNet_BN_Drop_Aug	99.52	3.6

hundred intersections with four or three branches are tested in the experiment, with over 350 image sequences and more than 3400 images.

Using hybrid results both from semantic segmentation and traffic signs detection & classification, a parsed scene with detailed semantic and attributed information can be established. For this purpose, the hierarchy of semantic objects needs to be applied, as there are coherent relations of topologies, attributes and semantics of the road objects. Therefore, an ATBT can be created based on urban rules for each image in the image track to depict the topologies among road objects. Then, we integrate the final updated ATBTs, rather than only using the result of one ATBT. Because some important items (such as traffic signs) in a certain image may be occluded by cars but the next image does not, which can play a role of verification and supplement. Ultimately, it can produce the final localization results along the driving direction (or camera shooting direction). Please note that this is not a precise localization, but in fact, it can indicate the estimated location of the traffic lights and signs. In Fig. 14, the qualitative localization results of one crossroad and one T-junction examples are displayed.

In general, for the localization task, two spatial data quality elements should be assessed: completeness and positional accuracy. While positional accuracy is the best-established indicator of accuracy in mapping science (Mobasheri et al., 2018), official position data (ground truth) of traffic signs and lights are not available. We cannot compare our generated positions with ground truth data on positional accuracy. However, we still manually collect the locations of traffic signs/lights from Google satellite map through visual observation and make these locations as “reference data” to access the completeness and positional accuracy of our localization results. In terms of completeness level, we get over 97% in all 100 testing intersections. Please note that only when all traffic lights and signs are detected and their predicted locations are not far away from the real locations at an intersection, then we would consider it as a complete and correct case. Figure 15 shows two examples corresponding to Fig. 14a-b respectively, where red dot 1 in the right figure of (a) contains three signs, and each of the red dots 1,2,3 in the right figure of (b) contains two signs because they are overlapped. As can be seen in the figures, both examples have obtained approximate positional accuracy compared to the annotated “reference data”.

5 Conclusions and future work

In this paper, we propose an automatic approach to detect and place traffic lights/signs at road intersections in relatively high completeness and positional accuracy. The proposed method relies on two deep learning pipelines (one for image semantic segmentation and the other for traffic sign detection & classification), as well as novel ATBTs based on six urban rules for traffic lights/signs localization. The method has been tested at multiple intersections using Mapillary street view images

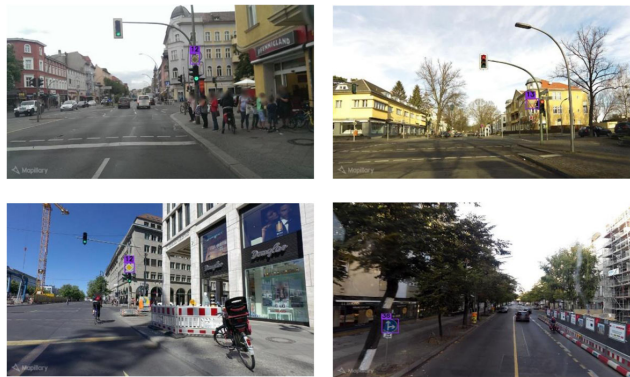


Fig. 13 Examples of traffic sign classification results based on ShallowNet

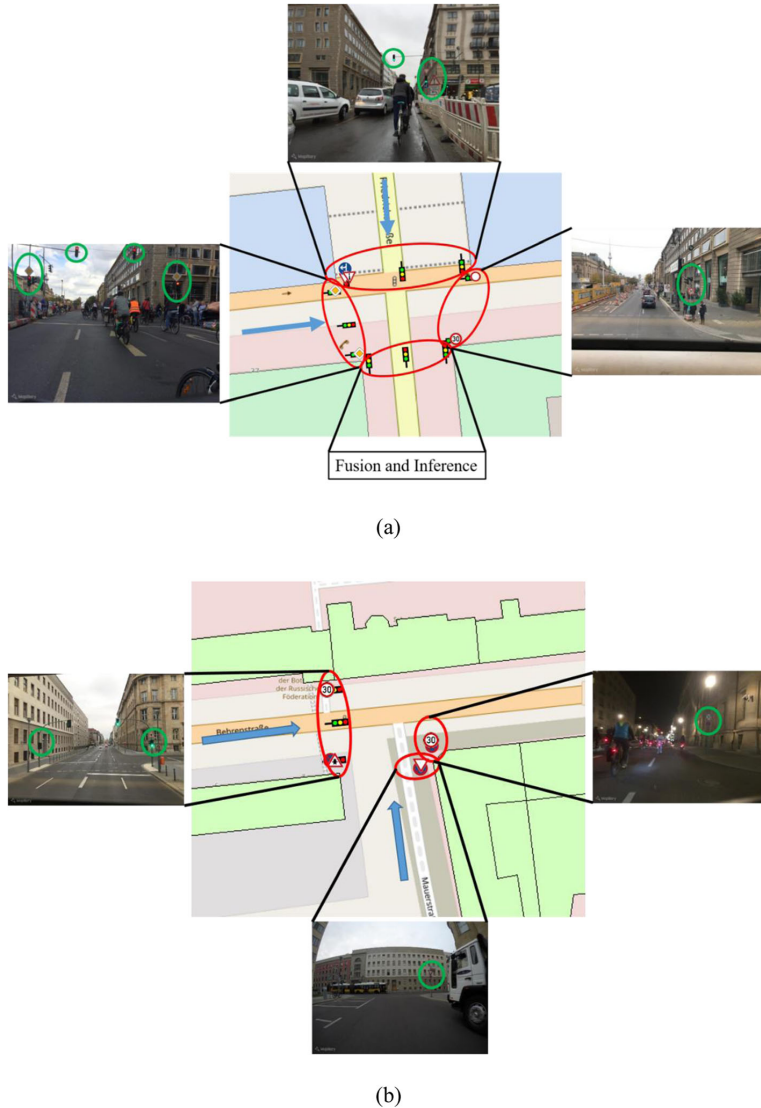
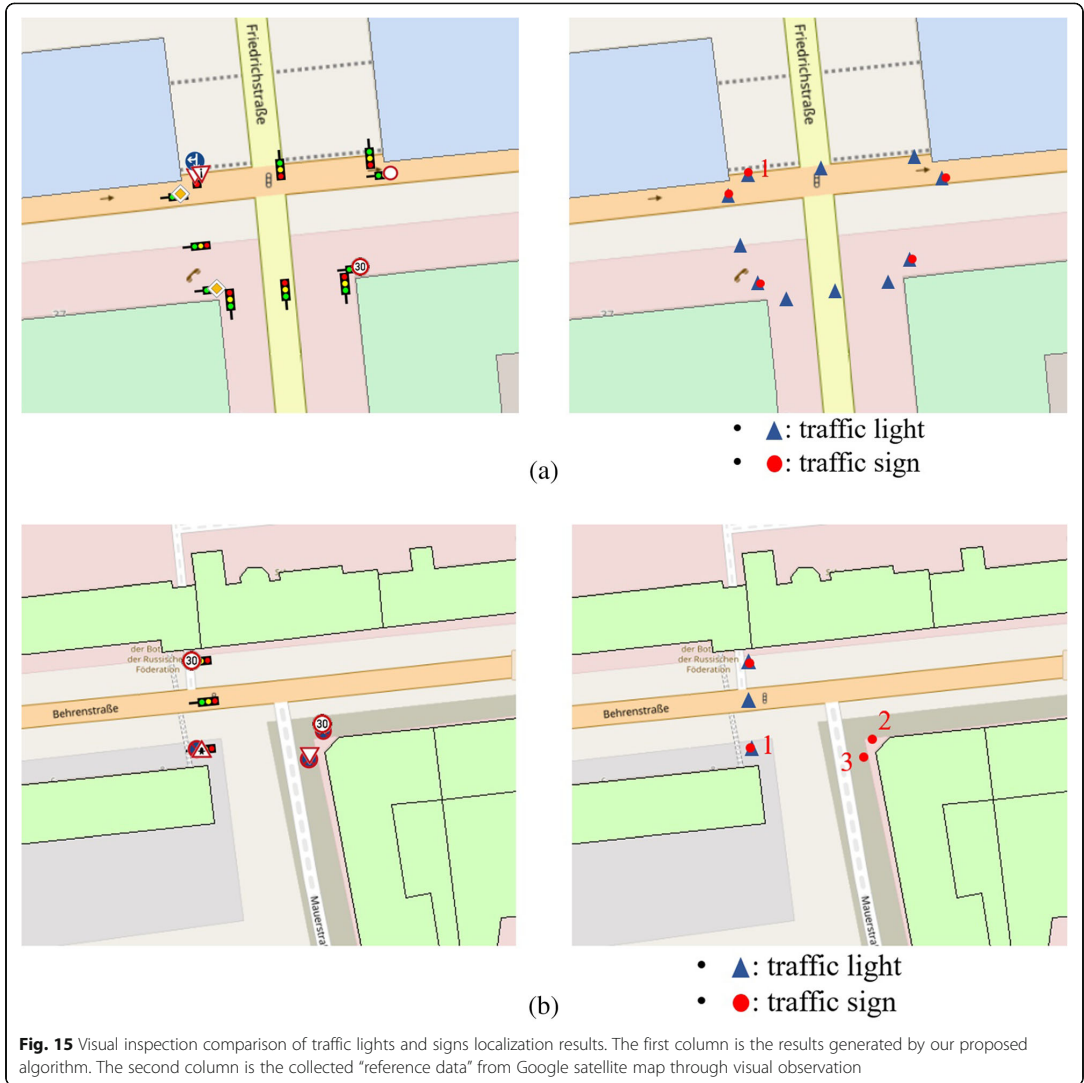


Fig. 14 Examples of qualitative localization results of traffic lights and signs at two types of intersections. **a** localization results at the crossroad; **b** localization results at the T-junction

in Berlin, Germany. We validate the effectiveness of the proposed approach on two object classes: traffic signs and traffic lights, and introduce two spatial data quality elements: completeness and positional accuracy. Experimental results demonstrate that our approach obtains great objects completeness level (over 97% among 100 testing intersections) and relatively high positional

accuracy compared to the manually collected “reference data”. Therefore, the proposed method provides a promising solution for enriching and updating OSM intersection data.

So far, to the best knowledge of the authors, there have not been digital maps with so detailed information. However, this kind of information is of vital importance



for many applications. For instance, together with trajectory data, information of traffic signs at road intersections may help offer more reasonable explanations for many spatial analyses related to urban structure and urban transportation. In this context, it is very useful for urban planning recommendations.

At present, the proposed method can only be applied to intersections with four or three branches and it is difficult to handle with complex intersections, such as roundabouts or five-branch intersections. In addition, the premise of employing this method is that there

needs to have at least one traffic light marked in OSM data. Otherwise, we cannot identify and select the intersections by using DBSCAN. That is the second limitation of the proposed approach. However, Europe’s OSM data is the richest compared to other continents, so the proposed method can be applied at least in Europe. In the future, we will further optimize the proposed approach and aim to resolve and overcome the above limitations. On the other hand, the GTSRB dataset used in this work only includes 45 categories, which does not cover all types of traffic signs in Germany or other

countries. Hence, another area for future research will be the extension of GTSRB dataset to increase the generalization of ShallowNet. Ultimately, we want to create and contribute a separate intersection layer to OSM, where contains the number of lanes, the width of roads and other road-related objects, and to provide some help for autonomous driving or navigation.

5.0.0.1 Code availability The code is currently stored on a local area network (LAN) of university and have not submitted to like Github. If this paper was accepted, we will release the code there immediately.

Authors' contributions

Hongchao Fan formed the research idea and revised the manuscript. Chaoquan Zhang designed and conducted the experiment, and drafted the manuscript. Wanzhi Li co-conducted the experiment and analyzed the results. All authors read and approved the final manuscript.

Funding

This research was supported by research fund from the Norwegian Public Roads Administration under project name Road.

Availability of data and materials

- <https://www.mapillary.com/data>
- <http://download.geofabrik.de/>

Declarations

Ethics approval and consent to participate

Not applicable.

Consent for publication

All authors approve that the Journal of Computational Urban Science can publish our article.

Competing interests

The authors declare that they have no competing interests.

Author details

¹Department of Civil and Environmental Engineering, Norwegian University of Science and Technology, Trondheim, Norway. ²School of Remote Sensing and Information Engineering, Wuhan University, Wuhan, China.

Received: 19 April 2021 Accepted: 18 July 2021

Published online: 04 August 2021

References

- Arietta, S. M., Efron, A. A., Ramamoorthi, R., & Agrawal, M. (2014). City forensics: Using visual elements to predict non-visual city attributes. *IEEE Transactions on Visualization and Computer Graphics*, *20*(12), 2624–2633. <https://doi.org/10.1109/TVCG.2014.2346446>.
- Chang, V., Walters, R. J., & Wills, G. B. (2016). Organisational sustainability modelling—An emerging service and analytics model for evaluating cloud computing adoption with two case studies. *International Journal of Information Management*, *36*(1), 167–179. <https://doi.org/10.1016/j.ijinfomgt.2015.09.001>.
- Chen, L. C., Zhu, Y., Papandreou, G., Schroff, F., & Adam, H. (2018). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 801–818).
- Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., et al. (2016). The cityscapes dataset for semantic urban scene understanding. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 3213–3223).
- Crandall, D. J., Backstrom, L., Huttenlocher, D., & Kleinberg, J. (2009). *Mapping the world's photos, Proceedings of the 18th international conference on world wide web* (pp. 761–770).
- Dubey, A., Naik, N., Parikh, D., Raskar, R., & Hidalgo, C. A. (2016). Deep learning the city: Quantifying urban perception at a global scale. In *Computer Vision – ECCV 2016*, Bastian Leibe, Jiri Matas, Nicu Sebe, and Max Welling (Eds.). Springer International Publishing, Cham, 196–212.
- Ester, M., Kriegel, H. P., Sander, J., & Xu, X., 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd* (Vol. 96, no. 34, pp. 226–231).
- Everingham, M., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, *88*(2), 303–338. <https://doi.org/10.1007/s11263-009-0275-4>.
- Gao, S., Li, L., Li, W., Janowicz, K., & Zhang, Y. (2017). Constructing gazetteers from volunteered big geo-data based on Hadoop. *Computers, Environment and Urban Systems*, *61*, 172–186. <https://doi.org/10.1016/j.compenvurbsys.2014.02.004>.
- Gebru, T., Krause, J., Wang, Y., Chen, D., Deng, J., Aiden, E. L., & Fei-Fei, L. (2017). *Using deep learning and google street view to estimate the demographic makeup of neighborhoods across the United States, Proceedings of the National Academy of Sciences* (pp. 13108–13113).
- Goodchild, M. F. (2007). Citizens as voluntary sensors: Spatial data infrastructure in the world of web 2.0 (editorial). *International Journal of Spatial Data Infrastructures Research (IJSIDR)*, *2*, 24–32.
- Hachmann, S., Arsanjani, J. J., & Vaz, E. (2018). Spatial data for slum upgrading: Volunteered geographic information and the role of citizen science. *Habitat International*, *72*, 18–26. <https://doi.org/10.1016/j.habitatint.2017.04.011>.
- He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778).
- Hebbalaguppe, R.; Garg, G.; Hassan, E.; Ghosh, H.; Verma, A., 2017. Telecom Inventory management via object recognition and localisation on Google Street View Images. In *Proceedings of the 2017 IEEE winter conference on applications of computer vision (WACV)*, Santa Rosa; pp. 725–733. IEEE.
- Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. R., 2012. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.
- Ibanez-Guzman, J., et al. (2010). Vehicle to vehicle communications applied to road intersection safety, field results. In *International IEEE conference on intelligent transportation systems* (pp. 192–197). Funchal Portugal: IEEE.
- Ioffe, S., & Szegedy, C., 2015. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference of machine learning* (pp. 448–456). PMLR.
- Jensen, M. B., Philipsen, M. P., Mogellose, A., Moeslund, T. B., & Trivedi, M. M. (2016). Vision for looking at traffic lights: Issues, survey, and perspectives. *IEEE Transactions on Intelligent Transportation Systems*, *17*(7), 1800–1815. <https://doi.org/10.1109/TITS.2015.2509509>.
- Krylov, V., Kenny, E., & Dahyot, R. (2018). Automatic discovery and geotagging of objects from street view imagery. *Remote Sensing*, *10*(5), 661. <https://doi.org/10.3390/rs10050661>.
- Kuntzsch, C., Sester, M., & Brenner, C. (2016). Generative models for road network reconstruction. *International Journal of Geographical Information Science*, *30*(5), 1012–1039. <https://doi.org/10.1080/13658816.2015.1092151>.
- LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *Nature*, *521*(7553), 436–444. <https://doi.org/10.1038/nature14539>.
- Li, X., Ratti, C., & Seiferling, I. (2018). Quantifying the shade provision of street trees in urban landscape: A case study in Boston, USA, using google street view. *Landscape and Urban Planning*, *169*, 81–91. <https://doi.org/10.1016/j.landurbplan.2017.08.011>.
- Li, X., Zhang, C., & Li, W., 2017. Building block level urban land-use information retrieval based on Google street view images. *GIScience & Remote Sensing*, *54*(6), 819–835.
- Li, X., Zhang, C., Li, W., Ricard, R., Meng, Q., & Zhang, W. (2015). Assessing street-level urban greenery using Google street view and a modified green view index. *Urban Forestry & Urban Greening*, *14*(3), 675–685. <https://doi.org/10.1016/j.ufug.2015.06.006>.
- Mattys, G.; Wang, S.; Fidler, S.; Urtasun, R., 2016. HD maps: Fine-grained road segmentation by parsing ground and aerial images. In *Proceedings of the 2016 IEEE conference on computer vision and pattern recognition (CVPR)*, Las Vegas; pp. 3611–3619. IEEE.
- Melnikov, V. R., Krzhizhanovskaya, V. V., Lees, M. H., & Boukhanovsky, A. V. (2016). Data-driven travel demand modelling and agent-based traffic simulation in Amsterdam urban area. *Procedia Computer Science*, *80*, 2030–2041. <https://doi.org/10.1016/j.procs.2016.05.523>.

- Mirowski, P., Grimes, M. K., Malinowski, M., Hermann, K. M., Anderson, K., Tepyashin, D., ... Hadsell, R., 2018. Learning to navigate in cities without a map. *arXiv preprint arXiv:1804.00168*.
- Mobasheri, A., Huang, H., Degrossi, L., & Zipf, A. (2018). Enrichment of openstreetmap data completeness with sidewalk geometries using data mining techniques. *Sensors*, 18(2), 509.
- Naik, N., Kominers, S. D., Raskar, R., Glaeser, E. L., & Hidalgo, C. A. (2017). Computer vision uncovers predictors of physical urban change. *Proceedings of the National Academy of Sciences*, 114(29), 7571–7576. <https://doi.org/10.1073/pnas.1619003114>.
- Naik, N., Philipoom, J., Raskar, R., & Hidalgo, C. (2014). Streetscore-predicting the perceived safety of one million streetscapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops* (pp. 779–785).
- Nair, V., & Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning (ICML-10)* (pp. 807–814).
- Neis, P., Zielstra, D., & Zipf, A. (2012). The street network evolution of crowdsourced maps: OpenStreetMap in Germany 2007–2011. *Future Internet*, 4(1), 1–21.
- Neuhold, G., Ollmann, T., Rota Bulo, S., & Kotschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *Proceedings of the IEEE international conference on computer vision* (pp. 4990–4999).
- Redmon, J., & Farhadi, A., 2018. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.
- Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91–99).
- Seitz, S.M.; Curless, B.; Diebel, J.; Scharstein, D.; Szeliski, R., 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Las Vegas; Volume 1, pp. 519–528. IEEE.
- Snavely, N., Seitz, S. M., & Szeliski, R. (2008). Modeling the world from internet photo collections. *International Journal of Computer Vision*, 80(2), 189–210. <https://doi.org/10.1007/s11263-007-0107-3>.
- Sohellian, B., Paparoditis, N., & Vallet, B. (2013). Detection and 3D reconstruction of traffic signs from multiple view color images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 77, 1–20. <https://doi.org/10.1016/j.isprsjprs.2012.11.009>.
- Jan Erik Solem., 2017. Mapillary: Celebrating 200 million images. Available online: <https://blog.mapillary.com/update/2017/10/05/200-million-images.html>. Accessed May 2020
- Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C., 2011. The German traffic sign recognition benchmark: A multi-class classification competition. In *The 2011 international joint conference on neural networks* (pp. 1453–1460). IEEE.
- Stallkamp, J., Schlipsing, M., Salmen, J., & Igel, C. (2012). Man vs. computer: Benchmarking machine learning algorithms for traffic sign recognition. *Neural Networks*, 32, 323–332. <https://doi.org/10.1016/j.neunet.2012.02.016>.
- Timofte, R.; Van Gool, L., 2011. Multi-view manhole detection, recognition, and 3D localisation. In *Proceedings of the 2011 IEEE international conference on computer vision workshops (ICCV workshops)*, Barcelona; pp. 188–195. IEEE.
- Trehard, G.; Pollard, E.; Bradai, B.; Nashashibi, F., 2014. Tracking both pose and status of a traffic light via an interacting multiple model filter. In *proceedings of the international conference on information FUSION (FUSION)*, Salamanca, pp. 1–7. IEEE.
- Wegner, J.D.; Branson, S.; Hall, D.; Schindler, K.; Perona, P., 2016. Cataloging public objects using aerial and street-level images-urban trees. In *Proceedings of the 2016 IEEE conference on computer vision and pattern recognition*, Las Vegas; pp. 6014–6023. IEEE.
- Workman, S.; Zhai, M.; Crandall, D.J.; Jacobs, N., 2017. A unified model for near and remote sensing. In *Proceedings of the 2017 IEEE conference on computer vision and pattern recognition (CVPR)*, Honolulu; Volume 7. IEEE.
- Yu, F., & Koltun, V., 2015. Multi-scale context aggregation by dilated convolutions. *arXiv preprint arXiv:1511.07122*.
- Zhang, F., Hu, M., Che, W., Lin, H., & Fang, C. (2018). Framework for virtual cognitive experiment in virtual geographic environments. *ISPRS International Journal of Geoinformation*, 7(1), 36. <https://doi.org/10.3390/ijgi7010036>.
- Zhao, H., Shi, J., Qi, X., Wang, X., & Jia, J., 2017. Pyramid scene parsing network. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*. (pp. 2881–2890). IEEE.
- Zhou, B., Zhao, H., Puig, X., Fidler, S., Barriuso, A., & Torralba, A., 2017. Scene parsing through ADE20K dataset. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition* (pp. 633–641). IEEE.

Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

PAPER 4

**An improved multi-task pointwise network for segmentation of building roofs
in airborne laser scanning point clouds**


Chaoquan Zhang^a, Hongchao Fan^a

^aDepartment of Civil and Environmental Engineering, Norwegian University of Science and
Technology, Trondheim, Norway

This paper is published in *The Photogrammetric Record*, July 2022; 37(179), 260-284.

Research publications

AN IMPROVED MULTI-TASK POINTWISE NETWORK FOR SEGMENTATION OF BUILDING ROOFS IN AIRBORNE LASER SCANNING POINT CLOUDS

CHAOQUAN ZHANG  (chaoquan.zhang@ntnu.no)

HONGCHAO FAN* (hongchao.fan@ntnu.no)

*Department of Civil and Environmental Engineering, Norwegian University of Science and
Technology, Trondheim, Norway*

*Corresponding author

Abstract

Roof plane segmentation is an essential step in the process of 3D building reconstruction from airborne laser scanning (ALS) point clouds. The existing approaches either rely on human intervention to select the appropriate input parameters for different data-sets or they are not automatic and efficient. To tackle these issues, an improved multi-task pointwise network is proposed to simultaneously segment instances (that is, individual roof planes) and semantics (that is, groups of roof planes with similar geometric shapes) in point clouds. PointNet++ is used as a backbone network to extract robust features in the first step. The features from semantics branch are then added to the instance branch to facilitate the learning of instance embeddings. After that, a feature fusion module is added to the semantics branch to acquire more discriminative features from the backbone network. To increase the accuracy of semantic predictions, fused semantic features of the points belonging to the same instance are aggregated together. Finally, a mean-shift clustering algorithm is employed on instance embeddings to produce the instance predictions. Furthermore, a new roof data-set (called RoofNTNU) is established by taking ALS point clouds as training data for automatic and more general segmentation. Experiments on the new roof data-set show that the method achieves promising segmentation results: the mean precision (mPrec) of 96.2% for the instance segmentation task and mean accuracy (mAcc) of 94.4% for the semantic segmentation task.

KEYWORDS: airborne laser scanning (ALS) point clouds, data-set, instance segmentation, roof plane segmentation

INTRODUCTION

THANKS TO THE ADVANCEMENT of laser scanning technology, airborne laser scanning (ALS) point clouds have become a significant data source for various mapping applications due to high accuracy. In addition to traditional applications in the remote sensing and photogrammetry

communities (Zhang and Lin, 2017), three-dimensional (3D) ALS point clouds have drawn significant attention from many other domains as well. For instance, high-definition map creation for autonomous driving (Liu et al., 2020), estimation of tree-level forest biomass (Wang et al., 2019c), power line diagnostics (Siranec et al., 2021), digital protection of cultural heritage (Ulvi, 2021), and 3D urban building reconstruction (Zhang et al., 2018). 3D building models can further help estimate the solar energy potential in urban areas (Nelson and Grubestic, 2020). As a result, that could contribute to achieving the carbon emission targets set by the Paris Agreement. Additionally, by calculating and analysing each roof plane's geometric structure and attributes (including slope, area, orientation, etc.), it would be beneficial for urban planners or solar power distributors to find out the most suitable places for solar panel installations in urban cities. For this purpose, automatic segmentation of roof planes is a crucial task in the process of the generation of 3D buildings from the ALS point cloud, and hence it is also the focus of this paper.

Roof plane segmentation is a complex task, since point clouds carry no connection information and do not provide any semantic features of the underlying scanned surfaces (Gilani et al., 2018). The existing approaches for this task are mostly based on clustering, model-fitting and region-growing. Clustering relies on the human experience to select appropriate input parameters and is sensitive to noise and outliers (Yan et al., 2014). Model fitting methods, such as RANSAC (RANDOM SAMple Consensus; Fischler and Bolles, 1981), are likely to produce spurious planes on complex roof structures. The region-growing method is limited by the choice of non-robust seed points/regions, the definition of the general criterion as well as high computational cost. Therefore, the use of these traditional approaches is restricted.

Over the past few years, 3D point cloud together with deep learning-based networks have made remarkable progress in the tasks of classification (Wu et al., 2019), semantic segmentation (Zhao et al., 2021) and instance segmentation (Zhao and Tao, 2020). Particularly, previous studies (Mao et al., 2019; Zhao et al., 2021) used convolutional neural networks (CNNs) for part segmentation on the ShapeNetPart data-set (Yi et al., 2016) and obtained impressive results, which has the potential to apply CNNs to the roof plane segmentation task. Part segmentation aims to segment meaningful parts from an object, for example, a chair mainly consists of three major meaningful parts (legs, seat and back). The roof plane segmentation task is similar to part segmentation, but will further segment a coarse part (for example, legs) into several separated instances (that is, independent legs). Consequently, to implement the task, it is necessary to train a CNN on the general and large scale ALS point cloud data-set with roof plane annotations.

In the practical work, the results of deep learning-based networks rely on the training data-sets. The currently open-source ALS point cloud data-sets, such as The Vaihingen 3D Benchmark (V3D) (Rottensteiner et al., 2012), DublinCity (Zolanvari et al., 2019), The Hessigheim 3D Benchmark (H3D) (Kölle et al., 2021) and RoofN3D (Wichmann et al., 2018), are region dependent, and thus roof structures in different regions vary greatly. Furthermore, although all contain roof categories, they lack the fine subdivision of roof planes, except RoofN3D. Nevertheless, the average point density of RoofN3D is only about 4.72 points/m^2 , which is far less than the standard point density ($10\text{--}12 \text{ points/m}^2$) captured by today's mainstream ALS equipment. Another drawback of RoofN3D is that it only covers three simple roof types (that is, gabled, hipped and pyramid), not to mention covering typical roof structures in West Europe (for example, corner element, cross element and T-element; Kada, 2007). As a result, the network trained on RoofN3D would be not general enough to segment the typical roof structures of West Europe from the most standard ALS point cloud data.

To close the above research gap, a new roof data-set (called RoofNTNU) is created from ALS point clouds for the purpose of roof plane segmentation. The original ALS point cloud was collected by Trondheim Municipality in Norway in 2018, with the standard point density. The roof data-set contains 930 roofs and covers seven types of typical roof structures in West Europe. Every roof is manually segmented and carefully annotated with both distinguishable plane labels (that is, instance labels), and semantic labels according to similar geometric shapes.

To the best of the authors' knowledge, there is no research applying the deep learning technique to roof planes segmentation. However, it was found that many previous studies conducted the instance segmentation on indoor/outdoor scenarios by applying deep learning. Unfortunately, these methods cannot be directly employed to roof plane segmentation due to the huge differences in object features. Therefore, the possibility of segmenting roof planes using a deep learning technique is explored and an improved multi-task pointwise network (named RoofNet) based on the ASIS network is proposed (Wang et al., 2019b). The proposed network is then trained on the RoofNTNU data-set. The proposed RoofNet is capable of simultaneously segmenting instances and semantics in point clouds. PointNet++ (Qi et al., 2017b) is first employed as the backbone network to extract robust features, where the encoder is shared but has two parallel branch decoders corresponding to two segmentation tasks. Second, the features outputting from the semantics decoder are divided into two sub-branches. One of them is adapted to the instance branch to facilitate the learning of instance embeddings. Third, a feature fusion (FF) module is added to the end of the second semantics sub-branch in order to obtain discriminative fused features from the backbone network. Meanwhile, to increase the accuracy of semantic predictions, semantic features belonging to the same instance group and original fused semantic features are aggregated together via a max aggregation operation. The main contributions of this study can be summarised as follows:

- A new roof point cloud data-set (called RoofNTNU) with seven different types of typical roof structures in West Europe with the standard point density of ALS point clouds is established. It can provide general usability for roof plane segmentation from most ALS point cloud data.
- An improved network, termed RoofNet, is proposed to mutually facilitate instance and semantic segmentation. An FF module is added to the semantics branch to increase the accuracy of semantic segmentation and, in turn, to promote instance segmentation with semantics awareness.

In addition, this study is a continuity of two earlier studies (Fan et al., 2021; Zhang et al., 2021), where a web-based interactive platform, VGI3D, was developed for 3D building reconstruction from street-level images. However, the roof structures are usually invisible in the street-level images. In VGI3D, roof models are automatically generated by selecting a specific roof type, but their geometries are usually inaccurate. Moreover, some buildings have complex roof structures, but they are not included in the predefined simple roof types (for example, gabled and hipped) of VGI3D. To solve these issues, segmenting roof planes from ALS point clouds could be a possible solution for the purpose of the reconstruction of roof models in future. As a result, building models can be made to have more accurate roof geometries.

The rest of this paper is organised as follows. It next reviews the relevant methods and point cloud data-sets. The methodology presents the proposed network for roof plane segmentation. The following section introduces the new roof point cloud data-set.

Experimental results and variants study are then presented. Conclusions and future work are finally discussed.

RELATED WORK

Building Roof Segmentation Methods

The popular traditional roof plane segmentation methods are almost data driven and generally can be classified into three major categories: clustering-based, model fitting-based and region growing-based methods.

Clustering-based methods are basically considered as unsupervised learning where points are classified into distinguishable primitives based on pre-computed local surface features or properties. Kong et al. (2014) proposed a novel combination of the K -plane and K -means algorithms aiming to produce high-precision segmentation of roof structures. Additionally, an improved initialisation method was used to acquire the better initial clustering centres for the K -means algorithm. In another way, Wang et al. (2019a) presented a new roof plane segmentation algorithm based on the DBSCAN density clustering technique. The optimal searching radius ϵ of DBSCAN can be automatically estimated through the intrinsic properties of the point cloud data. Despite the popularity of clustering-based approaches for segmentation, they are sensitive to noise and outliers, and have difficulty in neighbourhood selection as well as being computationally expensive for multidimensional features in large data-sets.

Model fitting-based methods consist of two widely used algorithms: the Hough Transform (HT) (Ballard, 1981) and the RANSAC. In the 3D HT for plane detection, all LiDAR points are first mapped to parameter space, in which each point corresponds to a roof plane in object space. Voting accumulators are then applied to count the points falling into the corresponding planes by searching for the local maximum values in the parameter space. These accumulators are the detected shapes in the object space. Nevertheless, the HT is sensitive to the selection of parameter values and inefficient for computational time (Nguyen and Le, 2013). The standard RANSAC method typically fits a best mathematical plane with the most inliers from the LiDAR points and then considers this mathematical plane as the detected planar shape. The investigation by Tarsh-Kurdi et al. (2007) compared the 3D HT and RANSAC and suggested that RANSAC was more efficient than HT in both segmented results and running time. While RANSAC performed well on simple roof structures, it would tend to generate spurious planes on complex roofs. Hence, Li et al. (2017) formulated an improved RANSAC approach with Normal distribution transformation cells to reduce the spurious planes as much as possible. Although RANSAC is robust on data-sets with a large amount of noise and outliers, the issue of the spurious planes cannot be completely solved.

Region growing-based methods are another major category for roof plane segmentation. They begin with a chosen seed region/point and iteratively expand to the seed's neighbouring points until the growing meets the criteria. For instance, Nurunnabi et al. (2012) selected the seed points with the least curvatures in local surfaces, and then employed angle differences between normal vectors, the distance of points to planes, and distance between two points as the criteria for the growing. To improve the computational efficiency and robustness of region growing, Xu et al. (2017) presented a voxel-based region growing method with robust principal component analysis for building roofs segmentation. Both qualitative and quantitative results surpassed the representative algorithms. However, many studies (Li et al., 2020; Shao et al., 2021) have revealed that

compared with clustering-based methods, while region growing-based methods are more robust to outliers and noise, it is challenging to accurately determine the boundaries between adjacent planes and may tend to over- or under-segment.

Deep Learning Methods of Segmentation on Point Clouds

With the development of deep learning technique on point clouds, the segmentation tasks can be divided into semantic segmentation and instance segmentation.

For semantic segmentation, PointNet (Qi et al., 2017a) and PointNet++ (Qi et al., 2017b) represent two great milestones for point-based 3D deep learning approaches. Many following studies (Zhang et al., 2019; Wang et al., 2020) were inspired by them and raised suggestions for improvements to generate pointwise segmentation through multilayer perceptron. Parallely, some researchers also explored and employed recurrent neural networks (RNNs) to do the semantic segmentation. 3P-RNN (Ye et al., 2018) captured local structures at various densities by a pointwise pyramid pooling module, and introduced two-direction hierarchical RNNs to learn the long-range spatial context. Moreover, a graph-based method is another research direction. Landrieu and Simonovsky (2018) proposed the superpoint graph (SPG) and concentrated on segmenting large scale indoor/outdoor point clouds. Recently, 3D-GCN (Lin et al., 2020), a 3D graph convolution network, was designed to learn local geometrical features from point clouds across scales, with a graph max-pooling scheme. However, these previous works did not consider aggregating instance embeddings into semantic segmentation.

For instance, segmentation, SGPN (Wang et al., 2018) is the first neural algorithm to generate instance proposals from indoor point clouds by learning the similarity matrix of the features. 3D-BoNet (Yang et al., 2019) directly regressed 3D bounding boxes for all instances and meanwhile predicted a pointwise mask for every instance as well. Voxelising point clouds before learning features is a classic strategy to reduce computational costs. Thus, PointGroup (Jiang et al., 2020) first voxelised the points and then constructed a 3D U-Net with submanifold sparse convolution and sparse convolution to generate instances. Similarly, OccuSeg (Han et al., 2020) also performed the voxelisation on point clouds and learned point-level features from voxels by applying a 3D U-Net. The learned features were then decoded to construct a graph and predicted the final instances. However, these studies did not adapt semantic features into instance feature space or take advantages of the semantic-aware instance features.

ALS Point Cloud Training Data-sets

Well-known ALS point cloud data-sets for roof segmentation tasks include the V3D, H3D, DublinCity and RoofN3D. The V3D is one of the most famous and high-quality data-sets specifically created for urban classification and 3D reconstruction. It contains nine annotated point categories including the roof. However, nowadays the V3D is outdated with an average point density of 5–7 points/m² and an insufficient number of points, making it inappropriate for deep learning applications (Varney et al., 2020). To overcome the point density and resolution issues, the highly dense H3D captured in Germany recently has a tendency to replace the V3D as a new benchmark for semantic segmentation of 3D point clouds and meshes. The point density of the H3D is up to 800 points/m², which is much greater than the data collected by current standard laser scanning equipment. Consequently, owing to the high cost of data acquisition and post-processing, high-dense data-sets may be valuable for small scale projects (for example, cultural heritage protection projects). In

addition, while the roof structures in H3D have something in common with the buildings found in Norway, the H3D does not annotate the roofs into separated roof planes. Hence, the H3D is unsuitable for the roof plane segmentation task. Another dense point cloud data-set is DublinCity with a point density of about 348 points/m², which has 13 categories including roofs. Similar to the H3D, however, the roofs are not further segmented into separated roof planes. Moreover, since the DublinCity data-set is captured in the city of Dublin, most of the roof structures are different from those in Norway and hence cannot be directly applied to the task. Lastly, only one data-set with separated roof plane labels is known, that is, RoofN3D. As far as is known, RoofN3D is the first data-set to provide both distinct classes for buildings and plane labels of each roof, for the purpose of 3D building reconstruction using deep learning technique (Wichmann et al., 2018). However, the data-set has a point density of only around 4.72 points/m², and the data-set only covers a few of simple roof types (for example, gabled, hipped). Besides, the data is obtained in New York City, where the roof structures differ a lot from those in West Europe. As a result, the network trained on RoofN3D would be not general enough to segment the typical roof structures of West Europe from the most standard ALS point cloud data.

In summary, the existing traditional roof plane segmentation approaches depend on human intervention and prior knowledge, and they are not automatic and efficient enough during the process of segmentation. Additionally, due to the huge differences in object features of indoor scenes, there are no deep learning-based methods that can be used directly to roof plane segmentation, not to mention the suitable and general training data-set for this task. In this paper, an automatic method together with a new roof data-set is proposed by using a deep learning technique.

METHODOLOGY

This study can be viewed as an improvement of the ASIS network (Wang et al., 2019b) with an added FF module. The ASIS network has achieved impressive instance and semantic segmentation results on an indoor point cloud data-set, S3DIS (Armeni et al., 2016), with its fast and efficient network structure, and novel mutual features learning strategy. However, this approach did not consider the fused features generated from the backbone network, which are beneficial to producing more discriminative features and increasing the accuracy of predictions. Consequently, an FF module is added to ASIS in order to address the defects in ASIS that were mentioned above. Besides, the network structure of ASIS is also partially modified to improve the overall segmentation results.

This section first elaborates the structure of the proposed RoofNet for roof plane instance and semantic segmentation. It then introduces two key modules of the RoofNet: the FF module and the joint features learning module.

Network Structure

As illustrated in Fig. 1a, the whole network is composed of three major components including a PointNet++ as the backbone network, an FF module and a joint-features learning module. The backbone network consists of a shared encoder and two parallel decoders. The purposes of using two parallel decoders are to extract point-level semantic features for semantic predictions (that is, groups of roof planes with similar geometric shapes) with one decoder branch, and to carry out the instance segmentation (that is, separated roof planes) with the other decoder branch. The FF module is added right after the semantic decoder to fuse the high- and low-level features from the semantic decoder.

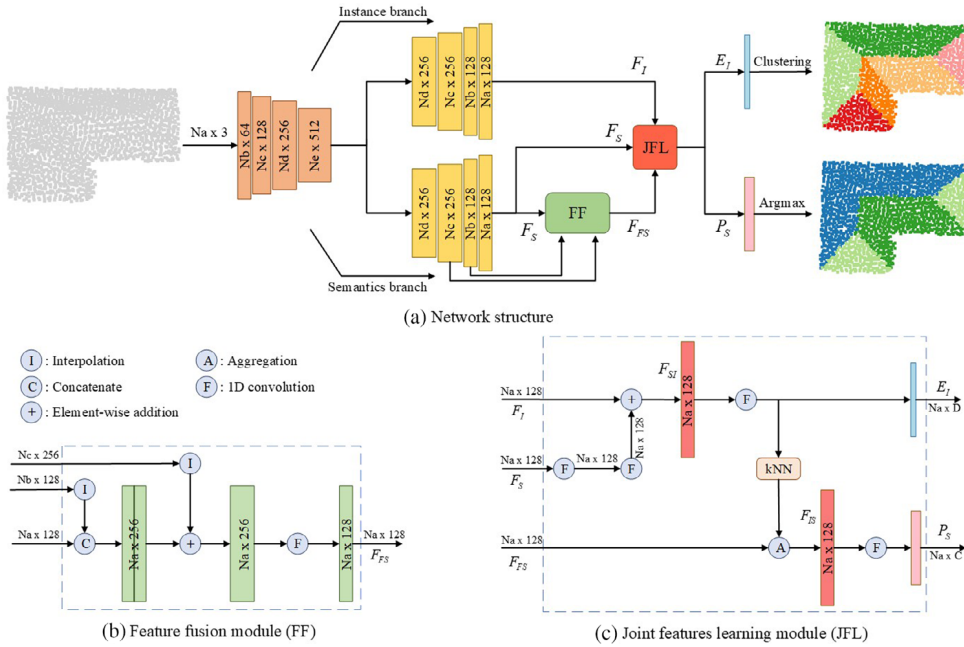


FIG. 1. (a) The architecture of RoofNet with two proposed modules: (b) the feature fusion (FF) module; and (c) the joint features learning (JFL) module.

The joint features learning module as the last part is to mutually promote the learning of semantic and instance features.

In the starting phase, a point cloud with size N_a is first input into the network and encoded into an $N_e \times 512$ feature matrix through the shared encoder (that is, four stacked set abstraction levels of PointNet++). Two parallel decoders then take the feature matrix as input and separately decode it into an $N_a \times 128$ -shaped feature matrix through four stacked feature propagation levels of PointNet++. F_S is for the output of the semantics decoder; and F_I is for the output of the instance decoder. Next, the semantics decoder further produces two parallel branches with the same semantic feature matrix F_S as input. One of them fuses the F_S with the features of different layers of the semantic decoder by an FF module, and it yields a new fused semantic feature matrix F_{FS} with shape of $N_a \times 128$. Lastly, three types of features, F_I , F_S and F_{FS} , will be inputted into the joint features learning module, which then produces two matrices (E_I and P_S) at the same time. The matrix E_I with the shape of $N_a \times D$ is the point-level instance embeddings, in which D is the dimension of the embeddings. E_I is used to predict the instance label for each point later by a clustering algorithm. Since the points belonging to the same instance are close to each other in embedding space, while points belonging to different instances are apart from each other (Wang et al., 2019b). Therefore, that is why embeddings from the network are only output rather than directly outputting the instance predictions. The other matrix P_S with shape $N_a \times C$ is the final semantic predictions and will output the most likely class from C types of candidate classes.

In the training phase, the RoofNet is supervised by a hybrid loss function ℓ_{all} , where consists of a standard softmax cross entropy loss ℓ_{sem} for learning per point semantics, and a discriminative loss function ℓ_{ins_emb} composed of three terms for learning instance

embedding, inspired by instance segmentation in 2D images (De Brabandere et al., 2017). The hybrid loss function ℓ_{all} is defined as follows:

$$\ell_{\text{all}} = \ell_{\text{sem}} + \ell_{\text{ins_emb}} = \ell_{\text{sem}} + (\alpha \cdot \ell_{\text{close}} + \beta \cdot \ell_{\text{apart}} + \gamma \cdot \ell_{\text{reg}}) \quad (1)$$

$$\ell_{\text{close}} = \frac{1}{K} \sum_{k=1}^K \frac{1}{N_k} \sum_t^{N_k} [\|\mu_k - e_t\|_1 - \delta_v]_+^2 \quad (2)$$

$$\ell_{\text{apart}} = \frac{1}{K(K-1)} \sum_{k_A=1}^K \sum_{k_B=1}^K [2\delta_d - \|\mu_{k_A} - \mu_{k_B}\|_1]_+^2, k_A \neq k_B \quad (3)$$

$$\ell_{\text{reg}} = \frac{1}{K} \sum_{k=1}^K \|\mu_k\|_1 \quad (4)$$

where ℓ_{close} exerts a pulling force that aims to draw embeddings towards the instance centre (that is, mean embedding of the instance); ℓ_{apart} exerts a pushing force that makes different instances separate from each other; and the regularisation term ℓ_{reg} is for pulling all instances towards the origin to form a boundary for embedding value. Furthermore, K is defined as the number of instances; N_k is for the number of points in the k -th instance; μ_k represents the centre of the k -th instance (the mean embedding); the embedding of a point is denoted as e_t ; $\|\cdot\|_1$ is for L_1 distance; δ_v and δ_d are the margins for the loss ℓ_{close} and ℓ_{apart} , respectively; and $[x]_+ = \max(0, x)$. The guidance of De Brabandere et al. (2017) is followed and $\alpha = \beta = 1$, $\gamma = 0.001$ are set in the experiments.

Finally, in the testing phase, a mean-shift clustering algorithm (Comaniciu and Meer, 2002) with bandwidth = 0.6 is employed to generate the instance labels on embeddings E_I yielded from the instance segmentation branch. As for semantic labels, an argmax operation is performed on semantic predictions P_S to acquire the semantic labels.

FF Module

As is known, the low-level layers of the network learn more detailed local features (for example, edges, curves, etc.), while the high-level layers would learn global semantic features (for example, the shape of objects). These insights are obtained by visualising the output of each network's layer (Qin et al., 2018). A number of researchers tried to fuse the features from different layers aiming to increase the accuracy of semantic segmentation. Their experimental results have verified that fused features benefitted the learning and resulted in better segmentation results, both for 2D images (Chen et al., 2018) and 3D point clouds (Hu et al., 2020).

Based on previous successful research cases, the FF strategy is also adopted and an FF module is introduced (Fig. 1b) into the network to promote the predictions. Given the computation efficiency and the consumption of graphics processing unit (GPU) memory, the last three layers of the decoder are only fused. The features output from the last three layers are denoted as F_c , F_b and F_a , respectively, corresponding to the shapes $N_c \times 256$, $N_b \times 128$ and $N_a \times 128$. In the FF module, an upsampling operation (that is, interpolation) is first performed on features F_b and F_c to make all features have the same points number, and thus obtain F'_b and F'_c . Second, F_a and F'_b are concatenated and $F_{ab'}$ is generated. $F_{ab'}$ is then added to F'_c through an element-wise addition, producing the feature $F_{ab'c'}$ shaped with $N_a \times 256$. Finally, a 1D convolution (Conv1D) with batch normalisation and ReLU non-linearity is applied on $F_{ab'c'}$ to yield the final fused feature F_{FS} with the shape

$N_a \times 128$. Moreover, the upsampling operation is conducted by using inverse distance-weighted average based on three nearest neighbours, following (Qi et al., 2017b).

Joint Features Learning Module

In general, instance segmentation is generated based on semantic segmentation results. As semantic segmentation has already grouped all points with the same semantic class together, it then needs an instance segmentation task to further separate each distinguishable instance from the groups. According to the above facts, the quality of the semantic features undoubtedly would influence a lot on instance segmentation results. Although the ASIS network has revealed that associative learning of two tasks can lead to a win-win situation (Wang et al., 2019b), it is not clear whether the fused features could better facilitate the semantic features learning and in turn benefit the learning of instance embeddings. Hence, in this study, the ASIS is modified by introducing an FF module, and then eventually form the joint features learning module, as illustrated in Fig. 1c.

Specifically, to integrate the semantic features into instance features, the original semantic feature matrix F_S is first transferred into an instance feature space as F'_S by two consecutive Conv1Ds with batch normalisation and ReLU non-linearity. Next, F'_S is element-wisely added to instance feature F_I and then it yields the instance features with semantic awareness as F_{SI} . Lastly, the instance embeddings E_I shaped with $N_a \times D$ are generated from semantic-aware instance features F_{SI} by a Conv1D together with dropout (Drop). The entire integration process can be described as follows:

$$F_{SI} = \text{Conv1D}(\text{Conv1D}(F_S)) + F_I \quad (5)$$

$$E_I = \text{Conv1D}(\text{Drop}(F_{SI})). \quad (6)$$

For the semantics branch, it is expected that the instance embeddings can feed back the learning of semantic features and make the boundary between two groups of semantic categories become more distinguishable and clearer. In the instance embedding space the points belonging to the same instance are closer, whereas different instances are repelled. Thus, ASIS is also followed and the M nearest neighbours (kNN) of each point are sought for in the instance embedding space. One can then obtain an index matrix which stores all indices of M nearest neighbouring points of each point, shaped by $N_a \times M$. Based on the index matrix, groups of fused semantic features denoted as G_{FS} with shape $N_a \times M \times 128$ can be generated from the fused semantic matrix F_{FS} (getGroupFeatures). Each group represents a local region of instance embedding space, which is close to its centroid point. Next, fused semantic features of each group G_{FS} and original semantic matrix F_{FS} are aggregated together via a max aggregation operation (Aggregation), producing the instance-aware semantic feature matrix F_{IS} . Finally, the final semantic predictions P_S with shape $N_a \times C$ can be generated from instance-aware semantic features F_{IS} by a Conv1D together with dropout. The whole procedure is shown as follows:

$$G_{FS} = \text{getGroupFeatures}(kNN(E_I), F_{FS}) \quad (7)$$

$$F_{IS} = \text{Aggregation}(G_{FS}, F_{FS}) \quad (8)$$

$$P_S = \text{Conv1D}(\text{Drop}(F_{IS})). \quad (9)$$

Roof Plane ALS Point Cloud Data-set

As mentioned above, the existing public ALS point cloud data-sets either lack roof plane annotations or their point density is too high/low compared with the standard point cloud, or their coverage of roof types is not sufficient, or the number of roof samples is not large. To close these research gaps, a new roof plane point cloud data-set suitable for training the deep learning models called RoofNTNU is manually established, which means a 3D roof-plane structure data-set built by the Norwegian University of Science and Technology (NTNU) in Trondheim, Norway. The raw ALS point cloud data with a point density of 12–20 points/m² is obtained from the mapping authority of Trondheim Municipality and was captured in April 2018, covering the whole of Trondheim. The typical roof structures are then selected from residential regions following the roof types (or roof primitives) proposed by Kada (2007). In addition, the roof types proposed by Kada are not completely followed, since some of them cannot be suitable in Norway. For example, the roof type “asymmetric hipped” normally does not exist in the raw point cloud data, so it is changed to “hipped” instead. Another finding is that many complex roof structures are a combination of two or three “sub-roof structures”, and hence these buildings are defined as an additional new roof type, combination. As a result, seven types of typical roof structures are presented in this paper, labelled from 1 to 7: flat, hipped, gabled, corner element, T-element, cross element and combination. Owing to the enormous variations in roof type combination, the other six roof types are only shown in Fig. 2.

For the instance labelling of the roof planes, it is observed that the planes comprising the roof structures have distinguishable geometric shapes. This observation is crucial because it will teach the neural network to learn the fundamental geometric shapes from point cloud data. Thus, these geometric shapes are summarised as five different categories: rectangle, isosceles trapezoid, triangle, parallelogram and ladder-shaped. Each category is given two or four labels based on the proposed roof types in Fig. 2. As a result, the label digits range from 0 to 11. For visualisation purposes, each label is assigned a unique but distinguishable colour, as shown in Fig. 3. Furthermore, in the idea of network design, it is expected that the features from the semantic segmentation branch can benefit the learning of instance embeddings. Therefore, the semantic labels for semantic features learning are also defined, being label digits from 0 to 4, which represent groups of roof planes with the same geometric shapes. For instance, the roof type hipped can be decomposed into two groups: a pair of triangles and a pair of isosceles trapezoids. Similar to the colours of instance labels,

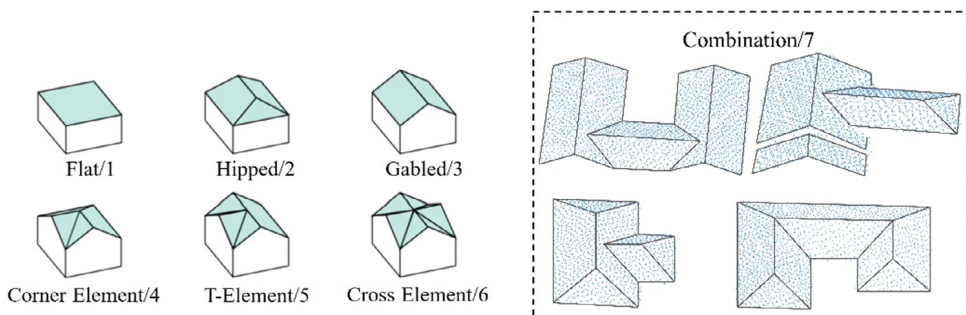


FIG. 2. Visualisation of the defined roof types (or primitives) in the data-set (reproduced from Kada, 2007): (left) abstract representation of roof types 1–6; and (right) some practical examples of roof type combination 7.

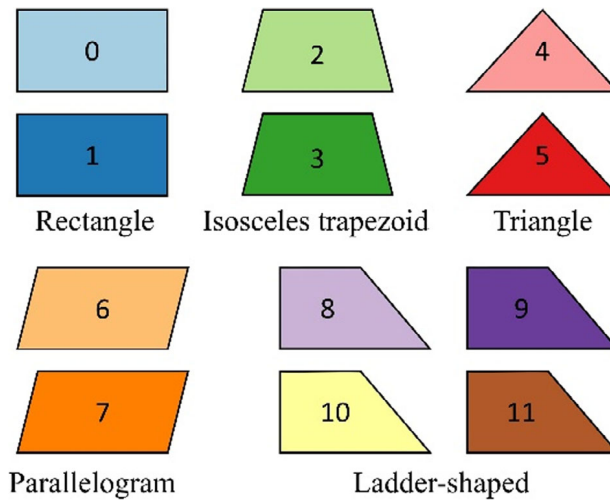


FIG. 3. Visualisation of the defined geometric plane shapes with their corresponding labels.

a specific colour is also assigned for each semantic label. Although some colours are the same between semantic and instance labels, they belong to two different tasks and hence the colours of semantic labels should not be confused with the colours of instance labels. The two sorts of labels and their correspondence with plane geometries and roof types can be found in Table 1. Thus, the structure of the ground truth is formulated as $[x, y, z, \text{roof-type}, \text{instance-label}, \text{semantic-label}]$, where the coordinate $[x, y, z]$ has been normalised.

Based on the principle and definitions formulated above, three research assistants and one of the authors used CloudCompare (2021) for the point cloud annotation. The detailed procedures of data annotation by using CloudCompare are as follows:

- The desired building roofs are extracted from the raw ALS point clouds and separately saved as .txt files with the file names of corresponding building IDs and roof types.
- For every single building roof (that is, stored as the .txt file), it is further segmented into separated roof planes according to the geometric shapes defined in Fig. 3. Each segmented roof plane would then be temporarily stored as a .txt file with the file name “buildingID_roofType_instanceLabel_semanticLabel”.
- An automated script is applied to merge all segmented roof planes that belong to the same roof into a complete building roof, writing the relevant labels/information for each point.
- A normalisation on the coordinates $[x, y, z]$ was performed, resulting in the final data-set.
- A cross-check strategy was then adopted to check the annotation quality and correct errors/noises if necessary.

A total of 930 different roofs with a total of over 2.2 million points were manually annotated and segmented into 3498 planes for the data-set RoofNTNU. The quantity distribution of the different roof types in RoofNTNU is presented in Fig. 4a, while the spatial geographical distribution of all roofs is illustrated in the form of highlighted building

TABLE I. Overview of the correspondence among semantic and instance labels, plane geometries and roof types in the ground truth.

<i>Semantic labels</i>	<i>Instance labels</i>	<i>Plane geometries</i>	<i>Roof types</i>
0	0	Rectangle	Flat, gabled, T-element, cross element, combination
1	1	Isosceles trapezoid	Hipped, corner element, combination
2	2		
3	3	Triangle	Hipped, corner element, combination
4	4		
5	5	Parallelogram	Corner element, combination
6	6		
7	7		
8	8	Ladder-shaped	T-element, cross element, combination
9	9		
10	10		
11	11		

Note: Do not confuse the semantic label colour with the instance label colour because they belong to two different tasks.

footprints in Fig. 4b. From Fig. 4a, an imbalance in the data-set with respect to the roof types is obvious. Gabled and T-element roof types dominate the data-set, while corner element and cross element roofs only occupy a small portion. However, this is actually consistent with the quantity distribution of these two roof types in the real world. Corner element and cross element roof structures are quite rare in residential areas of Norway, whereas gabled and T-element roofs are rather common. Besides, for the neural network training purpose, each roof type is randomly split into three sub-data-sets according to the ratio of 8:1:1, that is, training data-set (744), validation data-set (93) and testing data-set (93). Finally, the RoofNTNU data-set with these point clouds and their labels is established.

EXPERIMENTS

Experimental Settings

Data-set. The experiments were conducted on the RoofNTNU data-set. As mentioned above, the number of roofs in each category is unbalanced, especially for corner element and cross element roof structures. On the other hand, the roof structures in type combination are various and complex, but the geometric components that are similar to those in roof types 2–6 can be found. Therefore, based on this observation, the combination roofs were decomposed into separate geometric components by using CloudCompare (Fig. 5), and also randomly split into training, validation and testing data-sets according to the ratio of 8:1:1. The annotation principles for semantic and instance labels were followed as shown in Table I. Each point (x, y, z) was associated with a roof type label, an instance label and a semantic label. In this way, the training samples were also augmented for rare or specific roof geometries. Hence, the final amount for training, validation and testing data-sets were 834, 99 and 99, respectively.

Evaluation Metrics. For semantic segmentation evaluation, mean accuracy (mAcc), overall accuracy (oAcc) and mean intersection over union (mIoU) were computed. For instance for segmentation evaluation, the RoofNet network was evaluated by calculating the four metrics: mean coverage (mCov), mean weighted coverage (mWCov) (Ren and Zemel, 2017), mean recall (mRec) with IoU threshold of 0.5, and mean precision (mPrec).

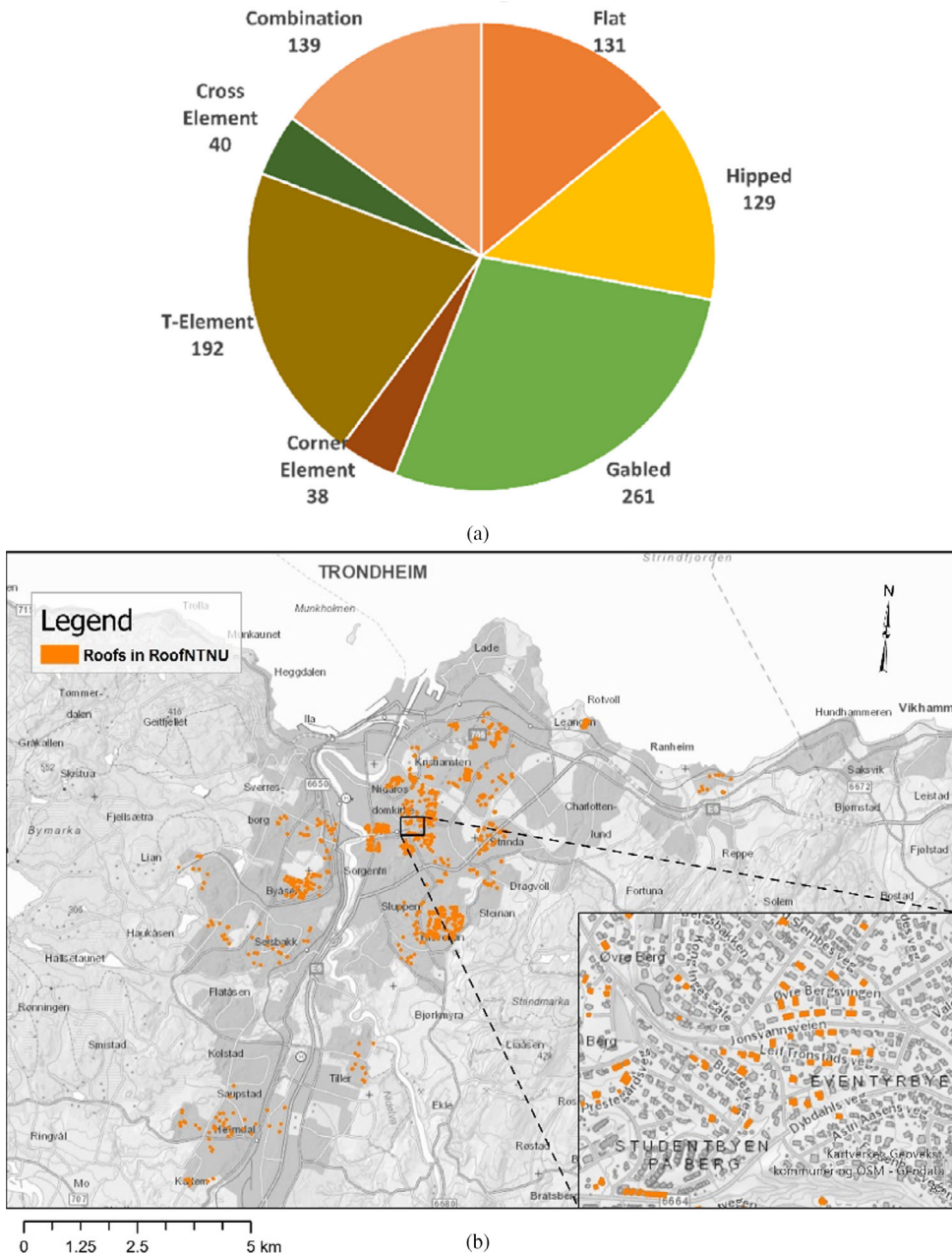


Fig. 4. Overview of the distribution of the RoofNTNU data-set: (a) the quantity distribution of roof types in the data-set; and (b) the spatial geographical distribution of all roofs, presented in form of highlighted building footprints.

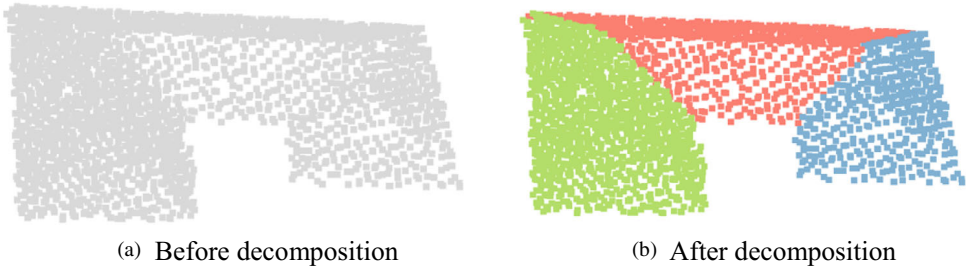


FIG. 5. Data augmentation by decomposing the combination roofs into several geometric components that have similarities to those in roof types 2–6: (a) before decomposition; and (b) after decomposition.

The Cov scores measure the instance-wise IoU of prediction matched with ground truth; the Cov score is then further weighted with the size of ground truth instances to acquire WCov (Wang et al., 2019b). Regarding ground truth regions GT and predicted regions P , Cov and WCov can be presented as follows:

$$\text{Cov}(GT, P) = \sum_{a=1}^{|GT|} \frac{1}{|GT|} \cdot \max_b \text{IoU}(r_a^{GT}, r_b^P) \quad (10)$$

$$\text{WCov}(GT, P) = \sum_{a=1}^{|GT|} w_a \cdot \max_b \text{IoU}(r_a^{GT}, r_b^P) \quad (11)$$

$$w_a = \frac{|r_a^{GT}|}{\sum_k |r_k^{GT}|} \quad (12)$$

where r_a^{GT} is the number of points in ground truth region a .

Training Details. The RoofNet network is implemented by using an open-source deep learning framework Pytorch (Paszke et al., 2019) and Python 3.5. At the training stage, the whole network is trained on a powerful NVIDIA Quadro GV100 GPU and an Intel Xeon(R) Gold 6146 central processing unit (CPU). The network is trained for 200 epochs with a batch size of 16. Each roof is randomly sampled into a sub-point cloud with 2048 points. The initial learning rate is set to 0.001 and divided by 2 every 20 epochs. Adam solver is used to optimise the network, with momentum = 0.9 and weight decay = 0.0001. The margin parameters $\delta_v = 0.5$ and $\delta_d = 1.5$ are set in the loss function and the embedding output dimensions D are set as 5 for the instance segmentation branch. For the kNN method in joint features learning module, 30 nearest neighbouring points were chosen. ReLU and batch normalisation are employed after each Conv1D except for the last layer for both instance and semantic branches. Additionally, for data augmentation strategies during the training, the sampling points were randomly scaled, shifted, jittered and perturbed. At the testing stage, a mean-shift clustering algorithm with bandwidth = 0.6 is adopted to generate the instances (that is, roof planes).

Segmentation Results

Instance Segmentation on RoofNTNU Data-set. The network is evaluated on the RoofNTNU data-set for the task of instance segmentation. Both quantitative and visual

TABLE II. Quantitative results of instance segmentation in the RoofNet (Ours).

Method	mCov (%)	mWCov (%)	mPrec (%)	mRec (%)
ASIS	85.0	85.0	94.8	91.7
JSNet	66.7	66.5	93.9	71.6
RoofNet (Ours)	85.3	85.2	96.2	91.7

Highest scores achieved by our proposed method are indicated in bold.

results achieved from the network are then compared with the other two multi-task networks: JSNet (Zhao and Tao, 2020) and ASIS (Wang et al., 2019b). As the backbone network, PointNet++, cannot perform the instance segmentation, it only provides the part segmentation, which is equivalent to the semantic segmentation task in this paper. Therefore, PointNet++ will not be compared here with other methods, but instead will be done so below.

Table II reports the quantitative results of different methods on instance segmentation task. The network slightly outperforms the ASIS in terms of metrics mCov and mWCov, but achieves a more significant improvement of 1.4% mPrec when evaluating on testing data-set. This reflects that the FF module and modified joint feature learning (JFL) module help to optimise the network structure, and the instance branch does better benefit from the semantics branch, which is consistent with the original assumption. Besides, as the other comparison method, JSNet is surprisingly and greatly inferior to the method on all four metrics except mPrec. Only its mPrec is close to the authors' method, but it is still worse than their method by 2.3%. The reason may be because JSNet is mainly suitable for indoor data-sets (for example, S3DIS; Armeni et al., 2016), where the indoor scenes are much more complex than roof structures. Thus, the architecture of JSNet is designed complicatedly with many convolution layers and other tensor operations. Based on this observation, it could be inferred that complex and deep network structures would not necessarily promote the features learning on data-sets with relatively simple scenes. Moreover, from the quantitative results, they also prove that the JFL module is reasonably designed and able to improve the performance of instance segmentation.

The visual comparison results produced from ground truth and different methods are presented in Fig. 6, where all roof types are listed, except roof type 7 combination because this category has been decomposed above as a special data augmentation strategy. Overall, the roof plane segmentation results generated from the authors' method are better than other two methods. The first three columns of Fig. 6 correspond to the most common and simple roof structures in Norway and other countries. Their amount is sufficient in the RoofNTNU data-set as well. Due to their simple geometric structures, all three methods perform well on them, yielding good roof plane segmentation results. The last three columns correspond to the roof types unique to Norway and Western European countries. For the fifth column, the amount of this roof type (T-element) is a lot in the data-set and its geometric structure is relatively simple compared with corner element and cross element. Therefore, in general, both ASIS and JSNet segment T-element roofs well, but the segmentation at some boundaries between adjacent roof planes is not good enough (see the red circles in Fig. 6). Some incorrectly classified points appear at the boundaries, or the planes are under/over-segmented. Similarly, the same errors can be observed from segmentation results on roof types corner element and cross element. Owing to their geometric complexity, more errors showing up can be observed. Even one roof plane is completely segmented wrongly, produced from JSNet on roof type cross element. In contrast, the segmentation results look great and distinguishable on all roof types because the semantic-aware instance features help

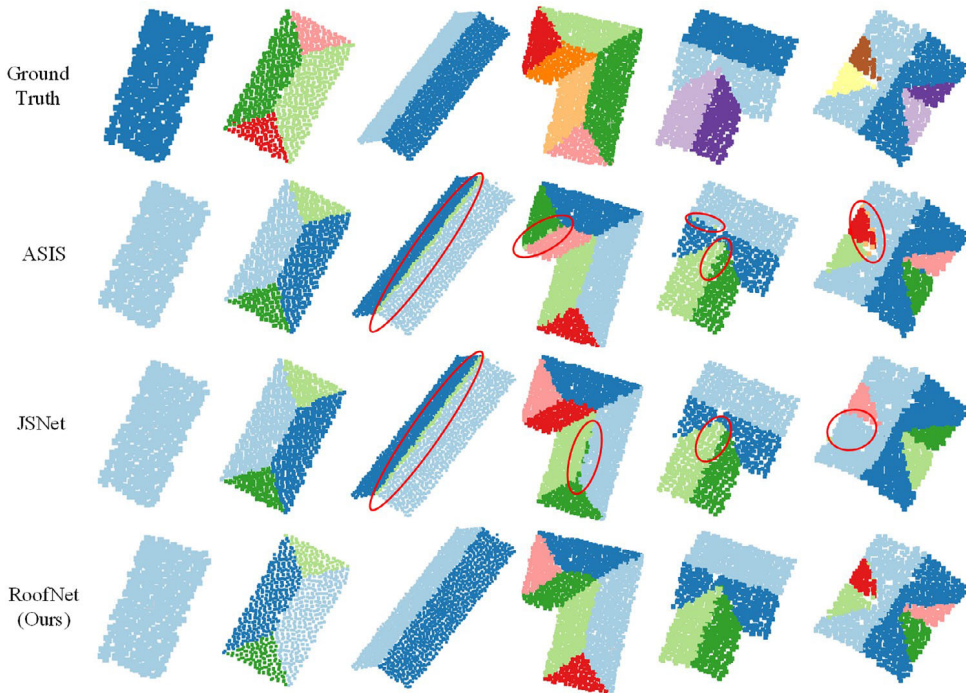


FIG. 6. Visual comparison results of ASIS, JSNet and the authors' method on the instance segmentation task. Different colours stand for different instances (that is, roof planes). The colours of the same instance in ground truth and prediction are not necessarily the same.

separate them in the instance embedding space. More instance segmentation results on the testing data-set of the RoofNTNU can be found in Appendix B in the additional supporting information.

Semantic Segmentation on RoofNTNU Data-set. The network is evaluated on the RoofNTNU data-set for the task of semantic segmentation. Both quantitative and visual results achieved from PointNet++ are then compared with the single scale grouping (SSG) as the baseline, the authors' method, ASIS as well as JSNet.

Table III illustrates the quantitative results of different methods on the semantic segmentation task. As shown, the method generates 86.7 mIoU, which significantly surpasses the PointNet++ baseline by 2.7%. Meanwhile, RoofNet also largely outperforms the ASIS and JSNet in terms of mIoU. As for the metric mAcc, RoofNet slightly performs better than baseline, but more significant improvements can be observed when comparing with ASIS and JSNet, improving 1.3% and 5.1%, respectively. Similar improvement can also be observed on metric oAcc. RoofNet surpasses the other three methods by 1.0%, 2.2% and 5.2%, respectively. As the backbone of the network is PointNet++, on the one hand, the good performance on PointNet++ indicates that it can indeed help extract useful and good-quality features. On the other hand, the fused features generated from the features fusion module can indeed further promote the learning of semantic features and are beneficial to the final semantic segmentation results.

TABLE III. Quantitative results of semantic segmentation on the RoofNet (Ours).

<i>Method</i>	<i>mAcc (%)</i>	<i>oAcc (%)</i>	<i>mIoU (%)</i>
PointNet++ (baseline)	93.8	95.4	84.0
ASIS	93.1	94.2	81.1
JSNet	89.3	91.2	66.4
RoofNet (Ours)	94.4	96.4	86.7

Highest scores achieved by our proposed method are indicated in bold.

Furthermore, considering the data imbalance issue on two roof types (corner element/4 and cross element/6), the scores on each roof type are calculated to present the performance of the RoofNet more objectively and to eliminate bias. The results are presented in Appendix A in the additional supporting information.

Similar to visual comparison results in Fig. 6, visual semantic results are also presented in Fig. 7. Again, the semantic segmentation results generated from the authors' method are better overall than other three methods. For the most common roof type hipped, the other three comparison methods do not well segment the boundaries between different pairs of geometric shapes. A similar situation happens on roof type corner element, but worse. Through the visualisation, it can be found that over/under-segmentations at the boundaries and incorrectly classified points exist in the JSNet result (row 4, column 4), including the result generated by the authors' method. However, the result is better than the others, with clear boundaries. This reflects that the improved result does benefit from the aggregation of instance embeddings, in which aggregates points belonging to the same instance yield more distinguishable and accurate boundaries. Regarding the incorrectly classified points, this situation frequently appears on results generated from ASIS and JSNet. That may be because ASIS does not fuse the semantic features, resulting in insufficient semantic features learning. Whereas, despite fusing the semantic features in JSNet, its network structure is too complex, and the learned features are too abstract, which is not beneficial to semantic segmentation.

Architecture Study

To evaluate the effectiveness of the improvements, five groups of experiments on the RoofNTNU data-set were conducted. Here the discussion only concentrates on the improved parts compared with ASIS, since ASIS has already proved its advantages with respect to semantic awareness (that is, transferring the semantic features to the instance branch) and instance aggregation (that is, aggregating the instance features into semantic branch). The key part of the baseline ASIS and details of five groups of experiments are listed in Fig. 8, where their encoders and decoders are consistent with RoofNet. In each group, the upper yellow bar represents the feature produced from the decoder of instance branch, while the lower yellow bar represents the feature produced from the decoder of semantic branch. FF stands for the FF module.

- ASIS baseline. The ASIS network is treated as the baseline (Fig. 8a), and all other groups will compare the results with the baseline.
- Add the FF module for both semantic and instance branches right after the decoders; meanwhile, remove the upper parallel feature F_S from the semantic branch and transfer the fused semantic features into instance feature space, as shown in Fig. 8b.

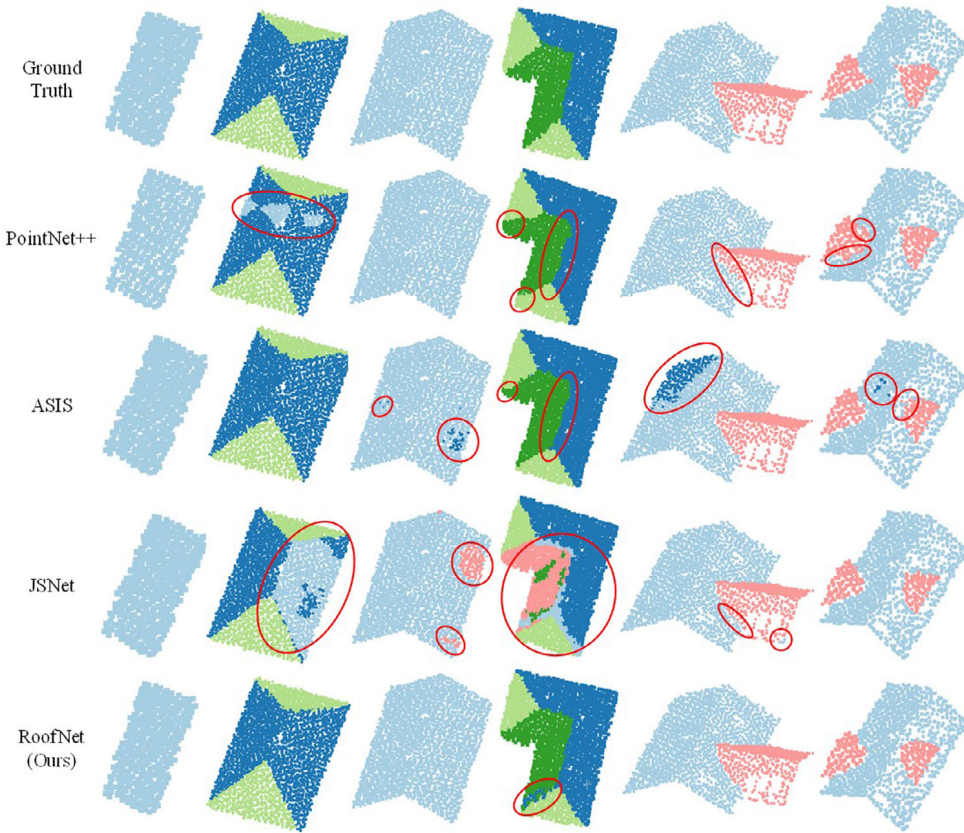


FIG. 7. Visual comparison results of PointNet++, ASIS, JSNet and the authors' method on the semantic segmentation task. Different colours represent different pairs of roof planes with the same geometric shapes.

- Remove the upper parallel feature F_S from the semantic branch and transfer the fused semantic features into the instance feature space, as shown in Fig. 8c.
- Retain two parallel features F_S of the semantic branch, but add the FF module after the upper parallel feature F_S and then transfer the fused semantic features into instance feature space, as shown in Fig. 8d.
- Keep the semantic branch and semantic awareness consistent with RoofNet, but greatly deepen the instance branch aiming to learn more high-level instance features, as shown in Fig. 8e. This idea borrows from JSNet.

Results Analysis. Three questions are discussed throughout the experiments: (1) whether the FF module needs to be applied to the instance branch (groups 1 and 2); (2) where to add the FF module to the semantics branch (groups 3–6); and (3) whether it is helpful to make instance branch deeper (group 5).

Table IV demonstrates the results for all experiments of architecture study. Comparing group 2 with the baseline, the FF modules indeed strengthen the semantic segmentation, but they weaken the instance segmentation, because integrating fused semantic features (that is,

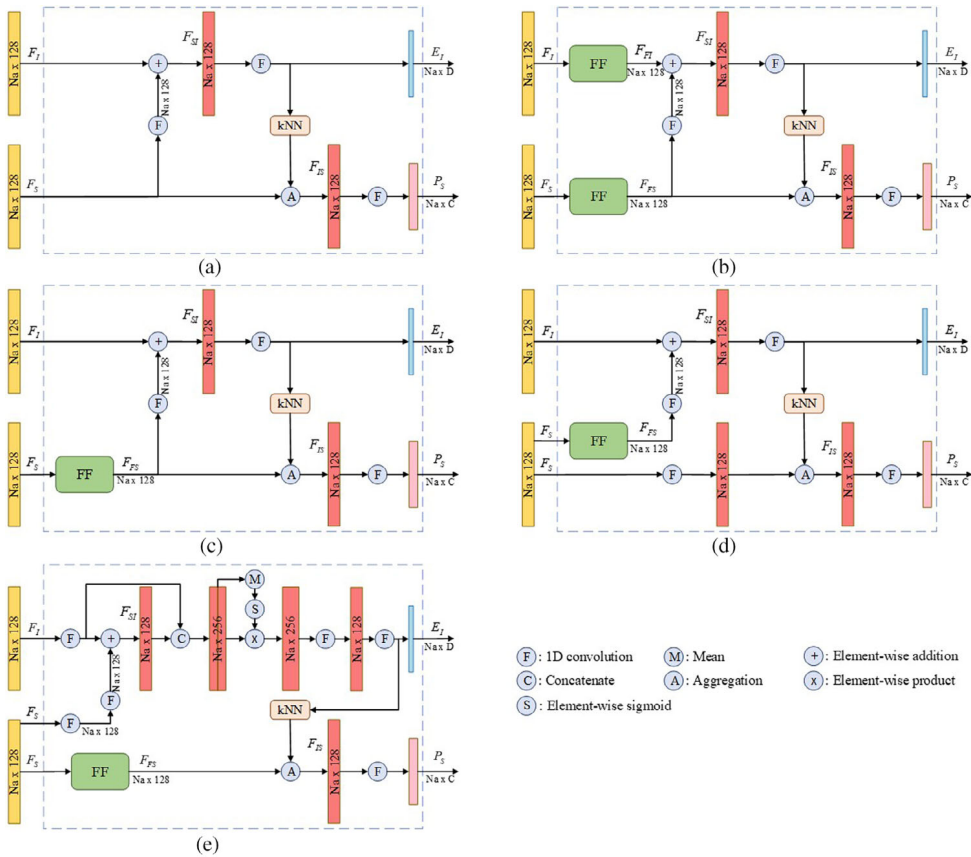


FIG. 8. Key part of baseline ASIS (a) and four groups of experiments (b–e). Their encoders and decoders are consistent with RoofNet.

points with the same semantic label would lie closer and form the roof parts) into fused instance features (that is, points with different instance labels would be as mutually exclusive as possible and form rather separated roof planes) may confuse the network (mainly is the instance branch) about what kind of features it should learn exactly. For this reason, the FF module was not added to the instance branch.

Compared with group (2), the FF module only applying to the semantics branch (that is, group 3) does benefit the overall performance of two segmentation tasks, because it could help the network to know more explicitly what features to learn for different branches, that is, penalising the over-separated semantic features on the semantics branch, as well as facilitating the learning of semantic-aware instance embeddings.

Compared with group (3), taking the FF module as a parallel semantic sub-branch and adding in the instance branch (that is, group 4) is less effective. Primarily this is because the aggregation operation (Aggregation) is directly performed on non-fused semantic features and hence cannot learn sufficient refined semantic features for each point. Therefore, the structure of the RoofNet (group 6 in Table IV) was designed in order to let the semantics branch learn refined semantic features for each point after the aggregation operation.

TABLE IV. Instance and semantic segmentation results of all experiments of architecture study on the RoofNet (Ours).

Group	Instance segmentation		Semantic segmentation	
	<i>mWCov (%)</i>	<i>mPrec (%)</i>	<i>mAcc (%)</i>	<i>mIoU (%)</i>
(1) Baseline (ASIS)	85.0	94.8	93.1	81.1
(2)	83.6	94.8	94.8	88.3
(3)	84.1	95.9	94.9	88.8
(4)	83.0	94.9	93.0	85.1
(5)	82.8	92.5	93.4	85.9
(6) RoofNet (Ours)	85.2	96.2	94.4	86.7

Highest scores achieved by our proposed method are indicated in bold.

Further, the measure of adding non-fused semantic features to the instance branch benefits the whole instance branch from the FF module because of the weights updating during the backpropagation.

The deeper network architecture (that is, group 5) is less effective for roof plane segmentation, since the complexity of roof planes' features is much lower than object features in indoor/outdoor scenes. The deeper and more complex network may lead to overfit and affect the segmentation performance instead.

Finally, even though RoofNet does not surpass all groups on both two-segmentation tasks, the final goal is to segment roof planes (or instances). Hence, a compromise was made and the network was chosen that could perform best on instance segmentation, that is, RoofNet.

CONCLUSIONS

This paper proposed an improved deep learning neural network, RoofNet, for the purpose of roof plane segmentation in ALS point clouds. The authors' network is composed of a shared encoder and two parallel decoders for extracting rich features, an FF module for fusing semantic features, and a JFL module for simultaneously facilitating the learning of semantic and instance features. Most importantly, a new roof plane training data-set, RoofNTNU, was manually established from standard ALS point clouds. The data-set includes seven types of typical roof structures and contains 930 roof samples located in Trondheim, Norway. The results of the variants study demonstrate that the design of the network is reasonable, and it achieves good performance on both instance and semantic segmentation tasks. The evaluation of the RoofNTNU data-set shows that the authors' method obtains a mean precision (mPrec) of 96.2% for the instance segmentation task and a mean accuracy (mAcc) of 94.4% for the semantic segmentation task. These results present the potential of the authors' approach to segment the roof planes in residential regions of Norway, even in some other countries of West Europe, from the most ALS point cloud data.

Since the use of kNN in the JFL module is limited by the selection of the k -value and distance metric, in future the network will be further optimised and improved to obtain more accurate roof plane segmentation results. The authors will also consider enriching the data-set to reduce the imbalance in the number of certain roof types (for example, cross element and corner element). Regarding the combination roofs (that is, roof type 7), the method currently cannot directly segment them due to their complexity and variety. The method will be further improved in future to increase the general ability of dealing with

hard cases in the real world. Furthermore, the roofs will be reconstructed after segmentation to help generate more accurate 3D building models. Last, the RoofNTNU data-set will soon be released to the public.

FUNDING

This research was supported by the Norwegian Public Roads Administration (Statens vegvesen) under the project name λ Road.

ACKNOWLEDGEMENT

Open access funding enabled and organized by ProjektDEAL.

REFERENCES

- ARMENI, I., SENER, O., ZAMIR, A. R., JIANG, H., BRILAKIS, I., FISCHER, M. and SAVARESE, S., 2016. 3D semantic parsing of large-scale indoor spaces. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1534–1543. <https://doi.org/10.1109/cvpr.2016.170>
- BALLARD, D. H., 1981. Generalizing the Hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2): 111–122. [https://doi.org/10.1016/0031-3203\(81\)90009-1](https://doi.org/10.1016/0031-3203(81)90009-1)
- DE BRABANDERE, B., NEVEN, D. and VAN GOOL, L., 2017. Semantic instance segmentation with a discriminative loss function. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 478–480. <https://doi.org/10.48550/arXiv.1708.02551>
- CHEN, L. C., ZHU, Y., PAPANDREOU, G., SCHROFF, F., and ADAM, H., 2018. Encoder–decoder with atrous separable convolution for semantic image segmentation. *Proceedings of the European Conference on Computer Vision (ECCV)*, 801–818. https://doi.org/10.1007/978-3-030-01234-2_49
- CLOUDCOMPARE. 2021. CloudCompare-Open Source project. <http://www.cloudcompare.org/> (Accessed 20th December 2021).
- COMANICIU, D. and MEER, P., 2002. Mean shift: A robust approach toward feature space analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 24(5): 603–619. <https://doi.org/10.1109/34.1000236>
- FAN, H., KONG, G. and ZHANG, C., 2021. An interactive platform for low-cost 3D building modeling from VGI data using convolutional neural network. *Big Earth Data*, 5(1): 49–65. <https://doi.org/10.1080/20964471.2021.1886391>
- FISCHLER, M. A. and BOLLES, R. C., 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM*, 24(6): 381–395. <https://doi.org/10.1016/b978-0-08-051581-6.50070-2>
- GILANI, S. A. N., AWRANGJEB, M. and LU, G., 2018. Segmentation of airborne point cloud data for automatic building roof extraction. *GIScience & Remote Sensing*, 55(1): 63–89. <https://doi.org/10.1080/15481603.2017.1361509>
- HAN, L., ZHENG, T., XU, L. and FANG, L., 2020. Occuseg: Occupancy-aware 3D instance segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2940–2949. <https://doi.org/10.1109/cvpr42600.2020.00301>
- HU, Q., YANG, B., XIE, L., ROSA, S., GUO, Y., WANG, Z., TRIGONI, N. and MARKHAM, A., 2020. Randlanet: Efficient semantic segmentation of large-scale point clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 11108–11117. <https://doi.org/10.1109/cvpr42600.2020.01112>
- JIANG, L., ZHAO, H., SHI, S., LIU, S., FU, C. W. and JIA, J., 2020. Pointgroup: Dual-set point grouping for 3D instance segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4867–4876. <https://doi.org/10.1109/cvpr42600.2020.00492>
- KADA, M., 2007. Scale-dependent simplification of 3D building models based on cell decomposition and primitive instancing. *International Conference on Spatial Information Theory*, Springer, Berlin, Heidelberg. 222–237. https://doi.org/10.1007/978-3-540-74788-8_14
- KÖLLE, M., LAUPHEIMER, D., SCHMOHL, S., HAALA, N., ROTTENSTEINER, F., WEGNER, J. D. and LEDOUX, H., 2021. The Hessigheim 3D (H3D) benchmark on semantic segmentation of high-resolution 3D point clouds and textured meshes from UAV LiDAR and multi-view-stereo. *ISPRS Open Journal of Photogrammetry and Remote Sensing*, 1: 100001. <https://doi.org/10.1016/j.ophoto.2021.100001>

- KONG, D., XU, L., LI, X. and LI, S., 2014. K-plane-based classification of airborne LiDAR data for accurate building roof measurement. *IEEE Transactions on Instrumentation and Measurement*, 63(5): 1200–1214. <https://doi.org/10.1109/tim.2013.2292310>
- LANDRIEU, L. and SIMONOVSKY, M., 2018. Large-scale point cloud semantic segmentation with superpoint graphs. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 4558–4567. <https://doi.org/10.1109/cvpr.2018.00479>
- LI, L., YANG, F., ZHU, H., LI, D., LI, Y. and TANG, L., 2017. An improved RANSAC for 3D point cloud plane segmentation based on normal distribution transformation cells. *Remote Sensing*, 9(5): 433. <https://doi.org/10.3390/rs9050433>
- LI, L., YAO, J., TU, J., LIU, X., LI, Y. and GUO, L., 2020. Roof plane segmentation from airborne lidar data using hierarchical clustering and boundary relabeling. *Remote Sensing*, 12(9): 1363. <https://doi.org/10.3390/rs12091363>
- LIN, Z. H., HUANG, S. Y. and WANG, Y. C. F., 2020. Convolution in the cloud: Learning deformable kernels in 3D graph convolution networks for point cloud analysis. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 1800–1809. <https://doi.org/10.1109/cvpr42600.2020.00187>
- LIU, R., WANG, J. and ZHANG, B., 2020. High definition map for automated driving: Overview and analysis. *The Journal of Navigation*, 73(2): 324–341. <https://doi.org/10.1017/s0373463319000638>
- MAO, J., WANG, X. and LI, H., 2019. Interpolated convolutional networks for 3D point cloud understanding. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1578–1587. <https://doi.org/10.1109/iccv.2019.00166>
- NELSON, J. R. and GRUBESIC, T. H., 2020. The use of LiDAR versus unmanned aerial systems (UAS) to assess rooftop solar energy potential. *Sustainable Cities and Society*, 61: 102353. <https://doi.org/10.1016/j.scs.2020.102353>
- NGUYEN, A. and LE, B., 2013. 3D point cloud segmentation: A survey. *2013 6th IEEE conference on robotics, automation and mechatronics (RAM)*. IEEE, Manila, Philippines. 225–230. <https://doi.org/10.1109/ram.2013.6758588>
- NURUNNABI, A., BELTON, D. and WEST, G., 2012. Robust segmentation in laser scanning 3D point cloud data. *2012 International Conference on Digital Image Computing Techniques and Applications (DICTA)*. IEEE, Fremantle, WA, Australia. pp. 1–8. <https://doi.org/10.1109/dicta.2012.6411672>
- PASZKE, A., GROSS, S., MASSA, F., LERER, A., BRADBURY, J., CHANAN, G., KILLEEN, T., LIN, Z., GIMELSHEIN, N., ANTIGA, L. and DESMAISON, A., 2019. Pytorch: An imperative style, high-performance deep learning library. *Advances in Neural Information Processing Systems*, 32: 8026–8037. <https://doi.org/10.48550/arXiv.1912.01703>
- QI, C. R., SU, H., MO, K. and GUIBAS, L. J., 2017a. Pointnet: Deep learning on point sets for 3D classification and segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 652–660. <https://doi.org/10.1109/cvpr.2017.16>
- QI, C. R., YI, L., SU, H. and GUIBAS, L. J., 2017b. PointNet++: Deep hierarchical feature learning on point sets in a metric space. *Advances in Neural Information Processing Systems*, 5105–5114. <https://doi.org/10.48550/arXiv.1706.02413>
- QIN, Z., YU, F., LIU, C. and CHEN, X., 2018. How convolutional neural networks see the world – A survey of convolutional neural network visualization methods. *Mathematical Foundations of Computing*, 1(2): 149. <https://doi.org/10.3934/mfc.2018008>
- REN, M. and ZEMEL, R. S., 2017. End-to-end instance segmentation with recurrent attention. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 6656–6664. <https://doi.org/10.1109/cvpr.2017.39>
- ROTTENSTEINER, F., SOHN, G., JUNG, J., GERKE, M., BAILLARD, C., BENITEZ, S. and BREITKOPF, U., 2012. The ISPRS benchmark on urban object classification and 3D building reconstruction. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences I-3*, 1(1): 293–298. <https://doi.org/10.5194/isprsannals-i-3-293-2012>
- SHAO, J., ZHANG, W., SHEN, A., MELLADO, N., CAI, S., LUO, L., WANG, N., YAN, G. and ZHOU, G., 2021. Seed point set-based building roof extraction from airborne LiDAR point clouds using a top-down strategy. *Automation in Construction*, 126: 103660. <https://doi.org/10.1016/j.autcon.2021.103660>
- SIRANEC, M., HÖGER, M. and OTCENASOVA, A., 2021. Advanced power line diagnostics using point cloud data – possible applications and limits. *Remote Sensing*, 13(10): 1880. <https://doi.org/10.3390/rs13101880>
- TARSHA-KURDI, F., LANDES, T. and GRUSSENMEYER, P., 2007. Hough-transform and extended ransac algorithms for automatic detection of 3D building roof planes from lidar data. *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, 36: 407–412. Retrieved from <https://halshs.archives-ouvertes.fr/halshs-00264843/document>

- ULVI, A., 2021. Documentation, three-dimensional (3D) modelling and visualization of cultural heritage by using Unmanned Aerial Vehicle (UAV) photogrammetry and terrestrial laser scanners. *International Journal of Remote Sensing*, 42(6): 1994–2021. <https://doi.org/10.1080/01431161.2020.1834164>
- VARNEY, N., ASARI, V. K. and GRAEHLING, Q., 2020. DALES: a large-scale aerial LiDAR data set for semantic segmentation. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, 186–187. <https://doi.org/10.1109/cvprw50498.2020.00101>
- WANG, C., JI, M., WANG, J., WEN, W., LI, T. and SUN, Y., 2019a. An improved DBSCAN method for LiDAR data segmentation with automatic Eps estimation. *Sensors*, 19(1): 172. <https://doi.org/10.3390/s19010172>
- WANG, X., LIU, S., SHEN, X., SHEN, C. and JIA, J., 2019b. Associatively segmenting instances and semantics in point clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 4096–4105. <https://doi.org/10.1109/cvpr.2019.00422>
- WANG, Y., JIANG, T., LIU, J., LI, X. and LIANG, C., 2020. Hierarchical instance recognition of individual roadside trees in environmentally complex urban areas from UAV laser scanning point clouds. *ISPRS International Journal of Geo-Information*, 9(10): 595. <https://doi.org/10.3390/ijgi9100595>
- WANG, W., YU, R., HUANG, Q. and NEUMANN, U., 2018. Sgpn: Similarity group proposal network for 3D point cloud instance segmentation. *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2569–2578. <https://doi.org/10.1109/cvpr.2018.00272>
- WANG, Y., PYÖRÄLÄ, J., LIANG, X., LEHTOMÄKI, M., KUKKO, A., YU, X., KAARTINEN, H. and HYYPPÄ, J., 2019c. In situ biomass estimation at tree and plot levels: What did data record and what did algorithms derive from terrestrial and aerial point clouds in boreal forest. *Remote Sensing of Environment*, 232: 111309. <https://doi.org/10.1016/j.rse.2019.111309>
- WICHMANN, A., AGOUB, A. and KADA, M., 2018. RoofN3D: Deep learning training data for 3D building reconstruction. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42(2): 1191–1198. <https://doi.org/10.5194/isprs-archives-xlii-2-1191-2018>
- WU, W., QI, Z. and FUXIN, L., 2019. Pointconv: Deep convolutional networks on 3D point clouds. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 9621–9630. <https://doi.org/10.1109/cvpr.2019.00985>
- XU, Y., YAO, W., HOEGNER, L. and STILLA, U., 2017. Segmentation of building roofs from airborne LiDAR point clouds using robust voxel-based region growing. *Remote Sensing Letters*, 8(11): 1062–1071. <https://doi.org/10.1080/2150704x.2017.1349961>
- YAN, J., SHAN, J. and JIANG, W., 2014. A global optimization approach to roof segmentation from airborne lidar point clouds. *ISPRS Journal of Photogrammetry and Remote Sensing*, 94: 183–193. <https://doi.org/10.1016/j.isprsjprs.2014.04.022>
- YANG, B., WANG, J., CLARK, R., HU, Q., WANG, S., MARKHAM, A. and TRIGONI, N., 2019. Learning object bounding boxes for 3D instance segmentation on point clouds. *Advances in Neural Information Processing Systems*, 32: 6740–6749. <https://doi.org/10.48550/arXiv.1906.01140>
- YE, X., LI, J., HUANG, H., DU, L. and ZHANG, X., 2018. 3D recurrent neural networks with context fusion for point cloud semantic segmentation. *Proceedings of the European Conference on Computer Vision (ECCV)*, 403–417. https://doi.org/10.1007/978-3-030-01234-2_25
- YI, L., KIM, V. G., CEYLAN, D., SHEN, I. C., YAN, M., SU, H., LU, C., HUANG, Q., SHEFFER, A. and GUIBAS, L., 2016. A scalable active framework for region annotation in 3D shape collections. *ACM Transactions on Graphics (ToG)*, 35(6): 1–12. <https://doi.org/10.1145/2980179.2980238>
- ZHANG, C., FAN, H. and KONG, G., 2021. VGI3D: an interactive and low-cost solution for 3D building modelling from street-level VGI images. *Journal of Geovisualization and Spatial Analysis*, 5(2): 1–16. <https://doi.org/10.1007/s41651-021-00086-7>
- ZHANG, J. and LIN, X., 2017. Advances in fusion of optical imagery and LiDAR point cloud applied to photogrammetry and remote sensing. *International Journal of Image and Data Fusion*, 8(1): 1–31. <https://doi.org/10.1080/19479832.2016.1160960>
- ZHANG, L., LI, Z., LI, A. and LIU, F., 2018. Large-scale urban point cloud labeling and reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 138: 86–100. <https://doi.org/10.1080/19479832.2016.1160960>
- ZHANG, Z., HUA, B. S. and YEUNG, S. K., 2019. Shellnet: Efficient point cloud convolutional neural networks using concentric shells statistics. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 1607–1616. <https://doi.org/10.1109/iccv.2019.00169>
- ZHAO, H., JIANG, L., JIA, J., TORR, P. H. and KOLTUN, V., 2021. Point transformer. *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 16259–16268. <https://doi.org/10.1109/iccv48922.2021.01595>

- ZHAO, L. and TAO, W., 2020. JSNet: Joint instance and semantic segmentation of 3D point clouds. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(7): 12951–12958. <https://doi.org/10.1609/aaai.v34i07.6994>
- ZOLANVARI, S. M., RUANO, S., RANA, A., CUMMINS, A., DA, SILVA, R. E., RAHBAR, M. and SMOLIC, A., 2019. DublinCity: Annotated LiDAR point cloud and its applications. *arXiv preprint arXiv:1909.03613*. <https://doi.org/10.48550/arXiv.1909.03613>

SUPPORTING INFORMATION

Additional Supporting Information may be found in the online version of this article.

Résumé

La segmentation de plans de toits est une étape essentielle dans le processus de reconstruction 3D de bâtiments à partir de nuages de points de laser aéroporté à balayage. Les approches existantes s'appuient sur une intervention humaine pour sélectionner les paramètres d'entrée appropriés pour différents jeux de données, ou bien elles ne sont pas automatiques ni efficaces. Pour résoudre ces problèmes, un système multitâche est proposé pour segmenter simultanément les objets (c'est-à-dire les plans de toits individuels) et leur sémantique (c'est-à-dire les groupes de plans de toits ayant des formes géométriques similaires) dans les nuages de points. PointNet++ est utilisé comme système de base pour extraire des caractéristiques robustes dans la première étape. Ensuite, les caractéristiques issues de la branche sémantique sont ajoutées à la branche objets pour l'apprentissage de l'intégration d'objets. Un module de fusion de caractéristiques est ensuite ajouté à la branche sémantique pour acquérir des caractéristiques plus discriminantes à partir du système de base. Afin d'augmenter la précision des prédictions sémantiques, les caractéristiques sémantiques fusionnées des points appartenant au même objet sont fusionnées. Enfin, un algorithme de décalage moyen est appliqué aux fusions d'objets pour produire les prédictions. En outre, nous avons créé un nouveau jeu de données de toits (appelé RoofNTNU) où les nuages de points laser servent de données d'entraînement pour produire une segmentation automatique et plus générale. Les expériences menées sur ce jeu de données montrent que notre méthode obtient des résultats de segmentation prometteurs : une précision moyenne de 96,2% pour la tâche de segmentation d'objets et une exactitude moyenne de 94,4% pour la tâche de segmentation sémantique.

Zusammenfassung

Die Segmentierung der Dachebene ist ein wesentlicher Schritt im Prozess der 3D-Gebäuderekonstruktion aus luftgestützten Laserscanning-Punktwolken (ALS). Die bestehenden Ansätze verlassen sich entweder auf menschliches Eingreifen, um geeignete Eingabeparameter für verschiedene Datensätze auszuwählen, oder sie sind nicht automatisch und effizient. Um diese Probleme anzugehen, wird ein verbessertes punktweises Multitasking-Netzwerk vorgeschlagen, um gleichzeitig Instanzen (d. h. einzelne Dachebenen) und Semantik (d. h. Gruppen von Dachebenen mit ähnlichen geometrischen Formen) in Punktwolken zu segmentieren. PointNet++ wird als Backbone-Netzwerk verwendet, um im ersten Schritt robuste Merkmale zu extrahieren. Dann werden die Merkmale aus dem Semantikzweig dem Instanzzweig hinzugefügt, um das Lernen von Instanzeinbettungen zu erleichtern. Danach wird dem Semantikzweig ein Merkmalsfusionsmodul hinzugefügt, um diskriminierendere Merkmale aus dem Backbone-Netzwerk zu erhalten. Um die Genauigkeit semantischer Vorhersagen zu erhöhen, werden fusionierte semantische Merkmale der Punkte, die zu derselben Instanz gehören, zusammen aggregiert. Schließlich wird ein Mean-Shift-Clustering-Algorithmus auf Instanz-Einbettungen angewendet, um die Instanz-Vorhersagen zu erzeugen. Darüber hinaus erstellen wir einen neuen Dachdatensatz (genannt RoofNTNU), indem wir ALS-Punktwolken als Trainingsdaten für die automatische und allgemeinere Segmentierung verwenden.

Experimente mit unserem neuen Dachdatensatz zeigen, dass unsere Methode vielversprechende Segmentierungsergebnisse erzielt: die mittlere Genauigkeit (*mPrec*) von 96,2 % für die Segmentierungsaufgabe und die mittlere Genauigkeit (*mAcc*) von 94,4 % für die semantische Segmentierungsaufgabe.

Resumen

La segmentación de planos de tejados es un paso esencial en el proceso de reconstrucción de edificios en 3D a partir de nubes de puntos de escaneo láser aerotransportado (ALS). Los enfoques existentes se basan en la intervención humana en la selección de los parámetros de entrada apropiados para diferentes conjuntos de datos, que no son automáticos ni eficientes. Para abordar estos problemas, se propone una red de puntos multitarea mejorada para segmentar simultáneamente instancias (es decir, planos de tejado individuales) y semántica (es decir, grupos de planos de tejado con formas geométricas similares) en la nube de puntos. PointNet++ se utiliza como red vertebradora para extraer características robustas en el primer paso. Luego, las funciones de la rama semántica se agregan a la rama de instancia para facilitar el aprendizaje de las inserciones de instancia. Después de eso, se agrega un módulo de fusión de características a la rama de semántica para adquirir características más discriminatorias de la red vertebradora. Para aumentar la precisión de las predicciones semánticas, se agregan las características semánticas fusionadas de los puntos que pertenecen a la misma instancia. Finalmente, se emplea un algoritmo de agrupamiento de cambio medio en incrustaciones de instancias para producir las predicciones de instancias. Además, establecemos un nuevo conjunto de datos de tejado (llamado RoofNTNU) tomando nubes de puntos ALS como datos de entrenamiento para una segmentación automática y más general. Los experimentos en nuestro nuevo conjunto de datos de tejado muestran que nuestro método logra resultados de segmentación prometedores: la precisión media (*mPrec*) del 96,2 % en la tarea de segmentación de instancias y la precisión media (*mAcc*) del 94,4 % en la tarea de segmentación semántica.

摘要

屋顶面片分割是从机载激光点云 (ALS) 重建三维建筑过程中必不可少的重要步骤。现有的方法要么依赖于人工干预来为不同的数据集选择合适的输入参数，要么它们不是自动和高效的。为了解决这些问题，本文提出了一种改进的多任务逐点网络，能够同时分割屋顶ALS点云中的实例（即单个屋顶平面）和语义（即具有相似几何形状的屋顶平面组）。PointNet++首先被用作主干网络来提取鲁棒的特征。接着，将语义分支上的特征添加到实例分支中，以促进实例嵌入特征 (*embeddings*) 的学习。之后，将特征融合模块添加到语义分支中，以从主干网络中获取更多有判别力的特征。为了提高语义预测的准确性，将属于同一实例的点的融合语义特征聚合在一起。最后，在实例嵌入特征上采用均值漂移 (*mean-shift*) 聚类算法来产生最终的实例预测。此外，通过将ALS点云作为训练数据，本文创建了一个新的屋顶数据集（称为RoofNTNU），用于自动和更通用的屋顶面片分割。在新屋顶数据集上进行的实验表明，本文提出的方法取得了令人鼓舞的分割结果：实例分割任务的平均精确度 (*mPrec*) 为96.2%，语义分割任务的平均准确度 (*mAcc*) 为94.4%。

Appendix A

Due to data imbalance issue on two roof types (corner element/4 and cross element/6), that may affect the testing performance of the RoofNet. Hence, we additionally and separately compute the scores on each roof type to present the performance of the RoofNet more objectively and to eliminate the bias. From the table, we can see our RoofNet does not perform too badly on roof types 4 and 6, even though the amount of these two roof types are far less than others (38 and 40, respectively) and the complexity of them is relatively higher than others. That reflects our RoofNet is robust and does not heavily rely on the amount/scale of the dataset. However, the surprising finding is that RoofNet does not perform well on decomposed roof type 7 (i.e., combination). That might be because for our RoofNet, some decomposed roof parts still belong to hard cases. Their special geometric structures are not sufficiently learned by our network, probably due to the quite small proportion of these special geometric structures in the entire dataset.

TABLE I. Quantitative comparisons among different roof types on two segmentation tasks.

<i>Roof Type</i>	<i>Instance Segmentation</i>				<i>Semantic Segmentation</i>	
	<i>Cov (%)</i>	<i>WCov (%)</i>	<i>Prec (%)</i>	<i>Rec (%)</i>	<i>Acc (%)</i>	<i>IoU (%)</i>
Type1	100	100	100	100	100	100
Type2	95.2	95.2	98.1	100	97.1	92.8
Type3	97.7	97.8	94.5	100	98.1	98.8
Type4	82.6	82.2	94.4	90.2	79.8	81.4
Type5	94.7	94.9	98.7	100	97.3	93.1
Type6	85.6	85.2	100	93.8	98.3	95.9
Type7(decomposed)	68.6	68.5	95.1	74.8	93.3	76.7

Appendix B

In this section, we list all roof planes segmentation results evaluated on the testing dataset of RoofNTNU. To visualize them beautifully, all roofs are sorted by their types and then present. Please note that we do not present segmentation results of *combination* roofs (roof type 7) because they have been decomposed as data augmentation.

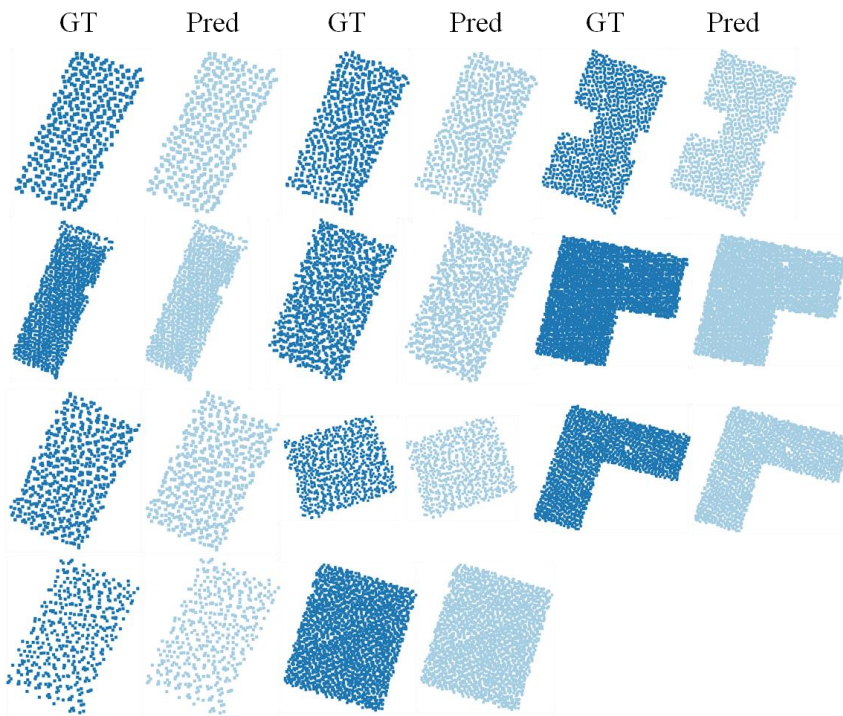


FIG. 1. Visual instance segmentation results of *flat* roofs (roof type 1) on testing dataset of RoofNTNU. The colors of the same instance in ground truth and prediction are not necessarily the same.

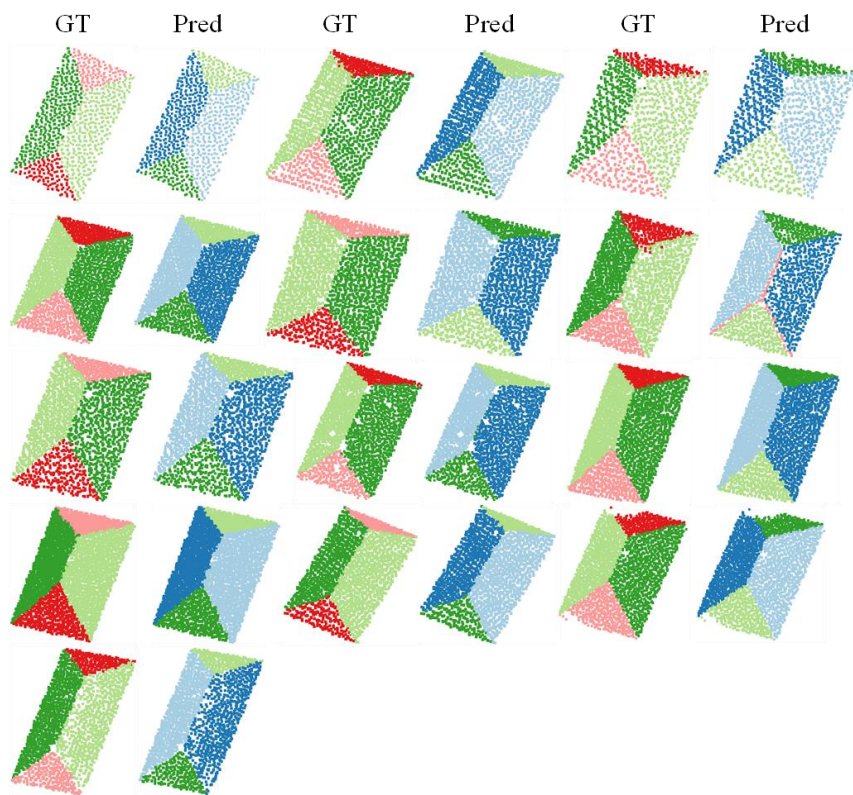


FIG. 2. Visual instance segmentation results of *hipped* roofs (roof type 2) on testing dataset of RoofNTNU. The colors of the same instance in ground truth and prediction are not necessarily the same.

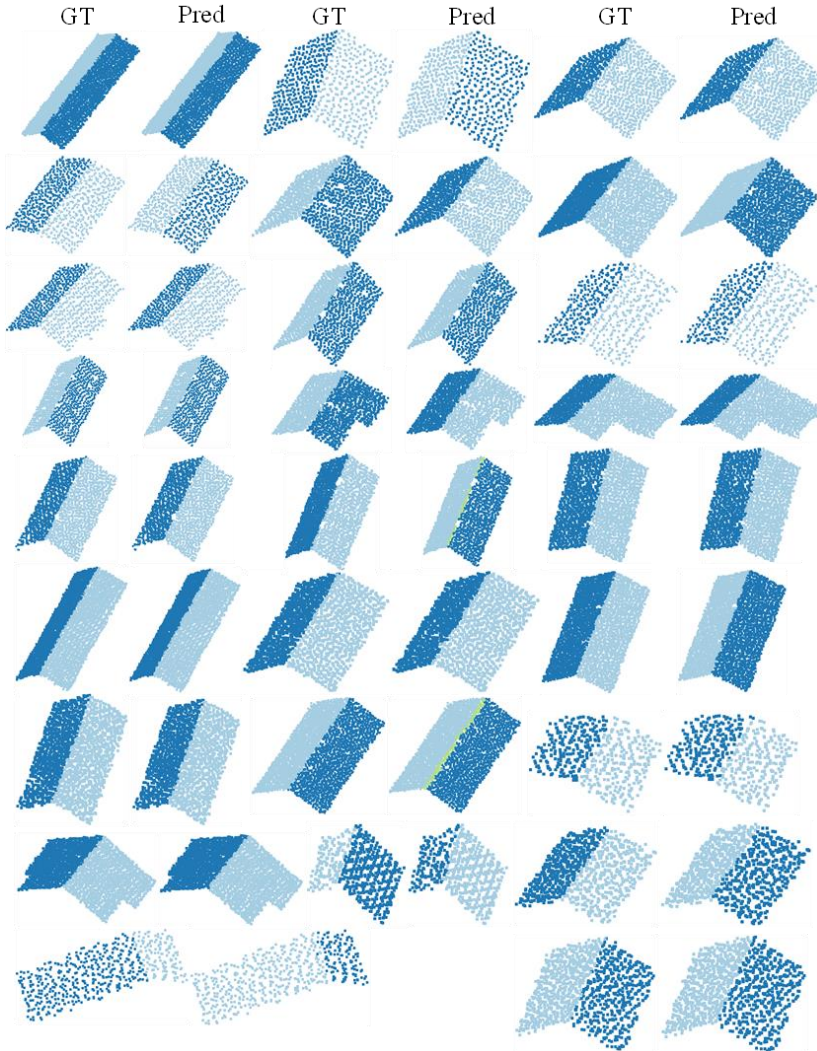


FIG. 3. Visual instance segmentation results of *gabled* roofs (roof type 3) on testing dataset of RoofNTNU. The colors of the same instance in ground truth and prediction are not necessarily the same.

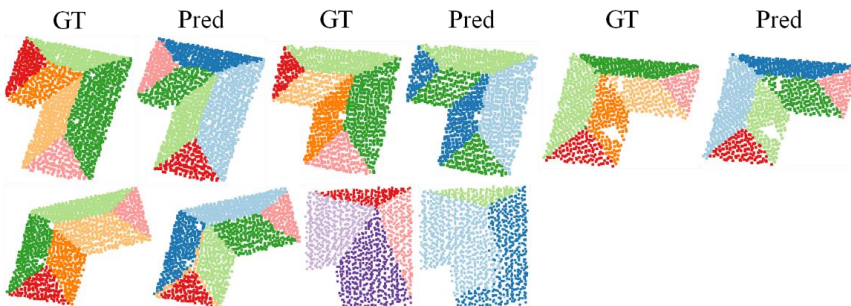


FIG. 4. Visual instance segmentation results of *corner-element* roofs (roof type 4) on testing dataset of RoofNTNU. The colors of the same instance in ground truth and prediction are not necessarily the same.

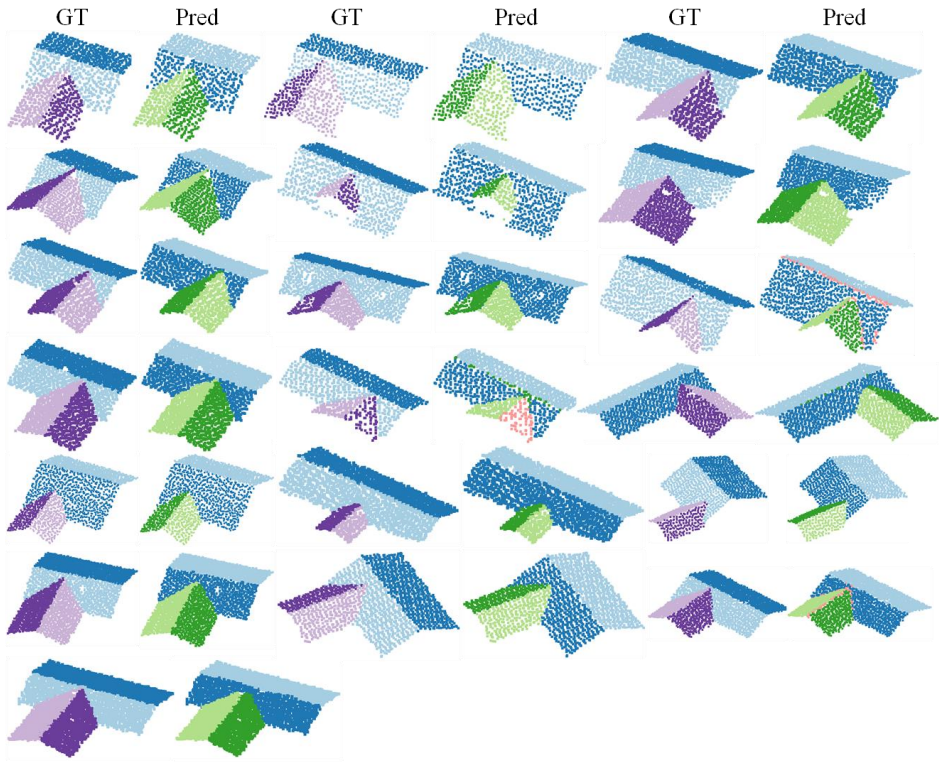


FIG. 5. Visual instance segmentation results of *T-element* roofs (roof type 5) on testing dataset of RoofNTNU. The colors of the same instance in ground truth and prediction are not necessarily the same.

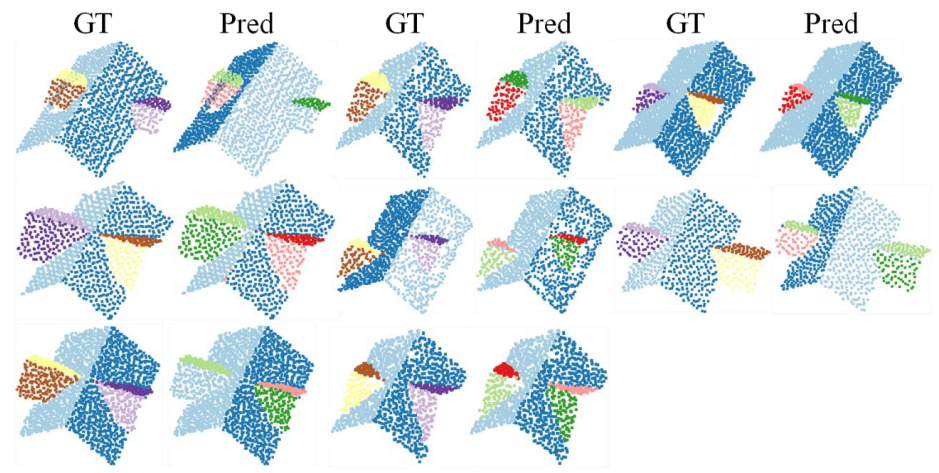


FIG. 6. Visual instance segmentation results of *cross-element* roofs (roof type 6) on testing dataset of RoofNTNU. The colors of the same instance in ground truth and prediction are not necessarily the same.

