



Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C

Dissertationen

Heft Nr. 851

Patrick Tutzauer

**On the Reconstruction, Interpretation and
Enhancement of Virtual City Models**

München 2020

Verlag der Bayerischen Akademie der Wissenschaften

ISSN 0065-5325

ISBN 978-3-7696-5263-5

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie der Universität Stuttgart
<http://dx.doi.org/10.18419/opus-10889>, Stuttgart 2020



Veröffentlichungen der DGK

Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften

Reihe C

Dissertationen

Heft Nr. 851

On the Reconstruction, Interpretation and Enhancement of Virtual City Models

Von der Fakultät für Luft- und Raumfahrttechnik und Geodäsie
der Universität Stuttgart
zur Erlangung des Grades
Doktor-Ingenieur (Dr.-Ing.)
genehmigte Dissertation

Vorgelegt von

Dipl.-Ing. Patrick Tutzauer

Geboren am 05.03.1988 in Herrenberg, Deutschland

München 2020

Verlag der Bayerischen Akademie der Wissenschaften

ISSN 0065-5325

ISBN 978-3-7696-5263-5

Diese Arbeit ist gleichzeitig veröffentlicht in:
Wissenschaftliche Arbeiten der Fachrichtung Geodäsie der Universität Stuttgart
<http://dx.doi.org/10.18419/opus-10889>, Stuttgart 2020

Adresse der DGK:



Ausschuss Geodäsie der Bayerischen Akademie der Wissenschaften (DGK)

Alfons-Goppel-Straße 11 • D – 80 539 München
Telefon +49 – 331 – 288 1685 • Telefax +49 – 331 – 288 1759
E-Mail post@dgk.badw.de • <http://www.dgk.badw.de>

Prüfungskommission:

Vorsitzender: Prof. Dr.-Ing. Nico Sneeuw

Referent: apl. Prof. Dr.-Ing. Norbert Haala

Korreferenten: Prof. Dr.-Ing. Helmut Mayer
Prof. Dr.-Ing. Uwe Sörgel

Tag der mündlichen Prüfung: 30.01.2020

© 2020 Bayerische Akademie der Wissenschaften, München

Alle Rechte vorbehalten. Ohne Genehmigung der Herausgeber ist es auch nicht gestattet,
die Veröffentlichung oder Teile daraus auf photomechanischem Wege (Photokopie, Mikrokopie) zu vervielfältigen

ISSN 0065-5325

ISBN 978-3-7696-5263-5

Contents

Acronyms	5
Abstract	7
Kurzfassung	9
1 Introduction	11
1.1 Motivation	11
1.2 Objectives	12
1.3 Outline	13
2 Related Work	15
2.1 Building Reconstruction and City Modeling	16
2.2 Urban Data Interpretation	18
2.2.1 Image-Based Building Understanding	18
2.2.2 Urban Object Segmentation and Classification	19
2.2.3 Cognitive Aspects in the Architectural Context	21
3 Building Models: Refinement, Demands, and Enhancements	23
3.1 Semi-Automatic Building Reconstruction	24
3.1.1 3D Cell Decomposition	25
3.1.2 Classification of 3D Cells	25
3.1.3 Radiometric Segmentation	26
3.1.4 Building Modeling	28
3.2 Human Understanding of Building Models	31
3.2.1 Development and Conduction of User Studies	31
3.2.2 Results and Application	34
3.3 Grammar-Based Enhancement and Abstraction	41
3.3.1 Building Feature Space	41
3.3.2 Perception-Based Rules	43
3.4 City Models in Virtual Reality (VR)	44
3.5 Summary for the Understanding and Enhancement of Building Models	46
4 Semantic Urban Mesh Segmentation	49
4.1 Data Preparation	51
4.1.1 Semantic Classes	51
4.1.2 Feature Vector Composition	53
4.2 Convolutional Neural Networks (CNNs)	56
4.3 Feature Based Semantic Mesh Segmentation	59
4.4 Smoothing of Segmented Triangles	60

4.5	Semantic Segmentation Results	61
4.5.1	Influence of Feature Vector Composition	63
4.5.2	Influence of Smoothing	64
4.5.3	Influence of Synthetic Data	64
4.6	Summary for the Semantic Enrichment of Textured Meshes	66
5	Building Interpretation	69
5.1	Automated Generation of Training Data Using Street View Imagery	70
5.1.1	Crawling Street View Imagery	70
5.1.2	Extraction of Building-Relevant Images	71
5.1.3	Registration of Imagery with Cadastral Data	76
5.2	Building Classification	77
5.2.1	CNN Training	79
5.2.2	Classification Results	80
5.3	CNN-Learned Features	82
5.3.1	Class-Activation Maps for Prediction Interpretability	82
5.3.2	Evaluation on Different Representation Types	84
5.3.3	Influence of Manipulated Images	87
5.3.4	Importance-Based Abstract Renderings	89
5.4	Summary for the Image-Based Interpretation of Building Facades	90
6	Conclusion and Outlook	93

Acronyms

ALS airborne laser scanning	IF <i>industrial facility</i>
ANN artificial neural network	IFC industry foundation classes
AR augmented reality	ILSVRC <i>ImageNet Large Scale Visual Recognition Challenge</i>
ASPRS <i>American Society for Photogrammetry and Remote Sensing</i>	
BIM building information modeling	k-NN k-nearest neighbors
BRep boundary representation	LiDAR light detection and ranging
BWS <i>building with shops</i>	LOD level of detail
	LUT lookup table
CAM class-activation map	MFB <i>multi-family building</i>
CE cross-entropy	ML machine learning
CGA computer generated architecture	MLP multilayer perceptron
CNN convolutional neural network	MNIST <i>Modified National Institute of Standards and Technology</i>
COG center of gravity	MRF markov random field
CPU central processing unit	MVS multi-view stereo
cuDNN NVIDIA CUDA deep neural network library	
DCNN deep convolutional neural network	NCM normalized confusion matrix
DIM dense image matching	OA overall accuracy
DL deep learning	OFB <i>one-family building</i>
DSM digital surface model	OFF <i>office building</i>
DTM digital terrain model	OGC <i>Open Geospatial Consortium</i>
	OSM Open Street Map
FCN fully convolutional network	PASCAL <i>Pattern Analysis, Statistical Modelling and Computational Learning</i>
FCS facade coordinate system	PCA principal component analysis
FL focal loss	PSB Princeton Shape Benchmark
FOV field of view	
GAN generative adversarial network	R-CNN Regional-CNN
GCNN graph CNN	RANSAC random sample consensus
GE Google Earth	ReLU rectified linear unit
GF ground floor	RF random forest
GIS geographic information system	RoI region of interest
GPS global positioning system	RPN region proposal network
GPU graphics processing unit	RT <i>residential tower</i>
GSD ground sampling distance	
GSV Google Street View	SACNN structure-aware CNN
GUI graphical user interface	

SfM Structure-from-Motion

SGD stochastic gradient descent

SGM semi-global matching

SPNH spherical neighborhood

SVM support vector machine

t-SNE t-distributed stochastic neighbor embedding

TLS terrestrial laser scanning

UAV unmanned airborne vehicle

VOC visual object classes

VR virtual reality

WP weighted precision

Abstract

With constant advances both in hardware and software, the availability of urban data is more versatile than ever. Structure-from-Motion (SfM), dense image matching (DIM), and multi-view stereo (MVS) algorithms have revolutionized the software side and scale to large data sets. The outcomes of these pipelines are various products such as point clouds and textured meshes of complete cities. Correspondingly, the geometric reconstruction of large scale urban scenes is widely solved. To keep detailed urban data understandable for humans, the highest level of detail (LOD) might not always be the best representation to transport intended information. Accordingly, the semantic interpretation of various levels of urban data representations is still in its early stages. There are many applications for digital urban scenes: gaming, urban planning, disaster management, taxation, navigation, and many more. Consequently, there is a great variety of geometric representations of urban scenes. Hence, this work does not focus on a single data representation such as imagery but instead incorporates various representation types to address several aspects of the reconstruction, enhancement, and, most importantly, interpretation of virtual building and city models. A semi-automatic building reconstruction approach with subsequent grammar-based synthesis of facades is presented. The goal of this framework is to generate a geometrically, as well as semantically enriched CityGML LOD3 model from coarse input data. To investigate the human understanding of building models, user studies on building category classification are performed. Thereof, important building category-specific features can be extracted. This knowledge and respective features can, in turn, be used to modify existing building models to make them better understandable. To this end, two approaches are presented - a perception-based abstraction and a grammar-based enhancement of building models using category-specific rule sets. However, in order to generate or extract building models, urban data has to be semantically analyzed. Hence, this work presents an approach for semantic segmentation of urban textured meshes. Through a hybrid model that combines explicit feature calculation and convolutional feature learning, triangle meshes can be semantically enhanced. For each face within the mesh, a multi-scale feature vector is calculated and fed into a 1D convolutional neural network (CNN). The presented approach is compared with a random forest (RF) baseline. Once buildings can be extracted from urban data representation, further distinctions can be made on an instance-level. Therefore, a deep learning-based approach for building use classification, i.e., the subdivision of buildings into different types of use, based on image representations, is presented. In order to train a CNN for classification, large amounts of training data are necessary. The presented work addresses this issue by proposing a pipeline for large-scale automated training data generation, which is comprised of crawling Google Street View (GSV) data, filtering the imagery for suitable training samples, and linking each building sample to ground truth cadastral data in the form of building polygons. Classification results of the trained CNNs are reported. Additionally, class-activation maps (CAMs) are used to investigate critical features for the classifier. The transferability to different representation types of building is investigated, and CAMs assist here to compare important features to those extracted from the previous human user studies. By these means, several integral parts that contribute to a holistic pipeline for urban scene interpretation are proposed. Finally, several open issues and future directions related to maintaining and processing virtual city models are presented.

Kurzfassung

Durch die ständige Weiterentwicklung von Hard- und Software ist die Verfügbarkeit von urbanen Daten vielseitiger denn je. Structure-from-Motion, dichte Bildzuordnung und Multi-View Stereo (MVS) Algorithmen sorgten softwareseitig für eine Revolution und haben die Skalierung auf große Datensätze ermöglicht. Das Ergebnis dieser Prozessierungen sind Punktwolken und texturierte Dreiecksvermaschungen ganzer Städte. Dementsprechend ist die geometrische Rekonstruktion von kompletten Stadtszenen weitgehend gelöst. Um detaillierte städtische Daten für den Menschen semantisch verständlich zu halten, ist der höchste Detaillierungsgrad allerdings nicht immer die beste Darstellung. Die semantische Interpretation urbaner Datenrepräsentationen befindet sich folglich noch in einem frühen Stadium. Es gibt viele Anwendungen für digitale Stadtszenen (z.B. Computerspiele, Stadtplanung, oder Navigation) und damit eine große Vielfalt an geometrischen Darstellungen dieser Szenen. Daher konzentriert sich diese Arbeit nicht auf eine einzige Datenrepräsentation, wie Bilder, sondern nutzt verschiedene Darstellungsarten, um mehrere Aspekte der Rekonstruktion, Anreicherung und vor allem der Interpretation von virtuellen Gebäude- und Stadtmodellen zu untersuchen. Es wird ein halbautomatischer Gebäuderekonstruktionsansatz mit anschließender Grammatikbasierter Synthese von Fassaden vorgestellt. Ziel dieses Ansatzes ist es, aus groben Eingangsdaten ein geometrisch und semantisch angereichertes Gebäudemodell zu generieren. Um das menschliche Verständnis von Gebäudemodellen zu untersuchen, werden Nutzerstudien zur Klassifizierung von Gebäudekategorien durchgeführt. Daraus lassen sich Merkmale ableiten, die spezifisch für eine bestimmte Gebäudenutzungsart sind. Dieses Wissen und die entsprechenden Merkmale können wiederum genutzt werden, um bestehende Gebäudemodelle zu modifizieren und besser verständlich zu machen. Um jedoch Gebäudemodelle aus urbanen Daten generieren oder extrahieren zu können, müssen diese zunächst semantisch analysiert werden. Daher wird ein Algorithmus für die semantische Segmentierung von urbanen Dreiecksvermaschungen vorgestellt. Explizite Merkmalsberechnung und faltungsbasiertes Lernen von Merkmalen wird kombiniert, um Dreiecksvermaschungen semantisch anzureichern. Für jedes Dreieck in der Vermaschung wird ein Multiskalen-Merkmalsvektor berechnet und in ein 1D faltendes neuronales Netzwerk (CNN) eingespeist. Sobald Gebäude aus einer urbanen Datenrepräsentation extrahiert werden können, ist es möglich, weitere Unterscheidungen auf Instanzebene vorzunehmen. Dazu wird ein Deep Learning-basierter Ansatz zur Klassifizierung von Gebäudenutzungen vorgestellt. Genauer handelt es sich hierbei um die Unterteilung von Gebäuden in verschiedene Nutzungsarten, basierend auf Bilddarstellungen. Um ein CNN für Klassifikationszwecke zu trainieren, sind große Mengen an Trainingsdaten erforderlich. Diese Arbeit präsentiert hierzu eine Pipeline für die großflächige, automatisierte Generierung von Trainingsdaten - Street View Daten werden extrahiert, nach geeigneten Trainings-Bildern gefiltert und jedes Gebäudes mit Referenzdaten aus dem Kataster, in Form von Gebäudepolygonen verknüpft. Die Klassifizierungsergebnisse der trainierten CNNs werden analysiert und zusätzlich werden Class-Activation Maps (CAMs) verwendet, um entscheidende Merkmale für den Klassifikator zuzufinden. Die Übertragbarkeit auf verschiedene Darstellungsarten von Gebäuden wird untersucht, und CAMs helfen hierbei wichtige Merkmale mit denen aus den zuvor angesprochenen Nutzerstudien zu vergleichen. Auf diese Weise werden mehrere integrale Bestandteile präsentiert, die zu einer ganzheitlichen Interpretation von urbanen Datenszenen beitragen. Schließlich werden mehrere offene Fragen und zukünftige Richtungen im Zusammenhang mit der Vorhaltung und Verarbeitung virtueller Stadtmodelle diskutiert.

Chapter 1

Introduction

1.1 Motivation

The virtual world has become an integral part of our everyday life. Whether using a computer at work or smartphones on the move, we are constantly interacting with connected devices to obtain and share information. One crucial component of that flow of information is geodata. We use online maps and route planners on a daily basis. Autonomous driving is about to become a reality in the near future. Therefore it is crucial to keep geodata and its resulting products scalable, maintainable, and, most importantly, up to date.

Recently, we see a shift of actual digital map representations from simple 2D data to 2.5D or even 3D virtual worlds. There is a variety of hardware and data sources capable of creating this content: airborne laser scanning (ALS) and terrestrial laser scanning (TLS), aerial nadir and oblique imagery, mobile mapping or handheld devices (cameras or even smartphones). Which results in a multitude of photogrammetric products - digital surface models (DSMs), (colored) point clouds, and more recently textured triangle meshes. With algorithms like semi-global matching (SGM) [Hirschmüller, 2008] and the possibility to perform dense image matching, geometric reconstruction of urban objects is quite sophisticated in the meantime. These technological developments trigger new applications and demands in the field of geoinformation and geovisualization. Additionally, it brings a new LOD to users and has simplified our lives in many regards.

However, these improvements also impose new challenges to keep detailed urban data understandable for humans. Is the highest LOD always the best representation to transport intended information? This question is even more valid when 3D city models are presented on small screens while on the go. In general, the geometry of buildings or city infrastructure is not the focus for end-user applications. The essential part is the semantic meaning that underlies the data. Semantic insights are not only valuable for planning and construction applications as required by building information modeling (BIM) or aspired urban development concepts like Smart Cities, but also provide new possibilities in enhancing the visual insight for humans when interacting with that kind of geodata.

Yet, semantic data interpretation and automatic extraction thereof is still in the early stages of development and a vivid research topic. Within this context, the photogrammetry and computer vision community benefit from the success of crowd-sourcing. Open Street Map (OSM) has proven its potential and success, giving everybody the possibility to enrich global maps in numerous ways. The recently emerged service Mapillary can be envisioned as a (partial) open source counterpart to GSV, opening new ways for volunteers to contribute imagery for mapping. Keeping cadastral and crowd-sourced map data up to date is a crucial

task in our ever-evolving urban environment. Up to now, a lot of this work is done manually. Any effort to automate these tasks is appreciated.

In addition to 3D data representations on 2D screens, immersive techniques and devices have become affordable for everyone. Besides watching movies in 3D and playing games in 3D virtual worlds, we can dive even deeper into an immersive experience with the growing availability of virtual reality (VR) devices in the meantime. In a white paper from 2017, Cisco has predicted a 20-fold increase of augmented reality (AR) and VR related internet traffic globally between 2016 and 2021 [Cisco, 2017]. Only recently, Google released a beta version for Google Maps AR walking navigation which uses a visual positioning service to locate users based on visual features from buildings and street furniture in assistance to often low accuracy global positioning system (GPS) and compass signals [Reinhardt, 2019]. Once users are visually located in the city by means of their smartphone camera, large visual direction indicator cues are overlaid into the scenery to assist navigation. Why not use VR and AR techniques in order to make urban data more accessible to broader masses?

Deep learning (DL) has revolutionized computer vision and machine learning and is now state of the art for many applications such as image classification, semantic segmentation, and natural language processing [Schmidhuber, 2015, LeCun et al., 2015]. Accordingly, DL has also found its way into photogrammetry and remote sensing, and is widely used for a variety of tasks: land cover classification [Marcos et al., 2018], building shape refinement [Bittner et al., 2018], digital terrain model (DTM) extraction [Gevaert et al., 2018], point cloud segmentation [Schmohl and Sörgel, 2019], and many more. However, as of now, massive amounts of manually labeled data are required in order for these algorithms to learn. Hence, much effort is put into sourcing this ground truth data, either commercially [Yuan, 2018] or voluntarily [Chen et al., 2019]. Automation in this context is desirable for the future as well.

The mentioned approaches either use image data or point clouds as basis for classification tasks. However, little work has been put into understanding mesh representations so far, which comes at a surprise. Textured meshes are the arguably most widely used 3D (or rather 2.5D) representation of virtual city models at the moment, namely in the form of Google Earth. Up to now, this information is purely geometric.

In order to tackle the open questions in the field of virtual city models, this thesis explores techniques to introduce a level of automatization for building reconstruction and enhancement. More importantly, algorithms for the semantic interpretation of urban data on several levels are introduced. For the interpretation and enhancement of building models, semantics are essential. The overarching goal of this work is, therefore, to make virtual city models more understandable for humans. As discussed above, there is a variety of data sources and applications within this context. Hence, this work does not focus on a single data representation such as imagery but instead incorporates various representation types for the variety of subtasks involved. Moreover, the influence of different representation types for the understanding of virtual building models is investigated.

1.2 Objectives

The overall objective of this thesis is to develop a framework for the reconstruction and enhancement of building models and the interpretation of virtual city models. Due to the variety of representation forms of urban scenes, different approaches should be investigated. In particular, the following problems are addressed:

- **Geometry of urban building models:** The focus is on the reconstruction of buildings from point clouds. A semi-automatic facade reconstruction approach takes point clouds (either from light detection and ranging (LiDAR) or DIM) and coarse building models as input. Raw point cloud data has to be processed to extract structural information. Previous work [Becker, 2011] dealt with custom data structures and could only handle single buildings. This framework should be modified and extended

so that inputs with large geospatial extent can be handled and standardized data formats can be used. Furthermore, previous work only relied on geometric information. This work should incorporate radiometric information as additional support. The approach should provide a semi-automatic pipeline to obtain reconstructed building models from point clouds, derive a grammar, complete the remainder of the building by means of this grammar, and export the final model to a standard data format [Tutzauer and Haala, 2015].

- Demands on virtual building representations: In order to understand what makes a building representation more comprehensible for humans, user studies should investigate this question. Therefore test persons were asked to participate in an extensive test on various building categories with different representation types [Tutzauer et al., 2016b, Tutzauer et al., 2016a].
- Enhancement of building models: A scheme to enhance existing virtual building representations is proposed. If semantics for buildings are known, it is possible to enhance existing virtual representations in a way that they become visually more comprehensible for humans. Specifically, the knowledge derived from the user study can be used for this task [Tutzauer et al., 2017].
- Use case of city models in VR: As mentioned in section 1.1, VR technology has found its way into the consumer market and should not only be considered a pure entertainment tool. There is potential to use this technology also for city model related tasks such as visualization, inspection, and maintenance. Hence, a use case for virtual city models in a VR environment should be provided, where users can interact with different representation types, extract information, or even manipulate geometry [Han, 2016].
- Semantic segmentation of urban textured triangle meshes: Virtual city models are currently mainly applied for visualization purposes. However, the enhancement of building models, as mentioned earlier, is only possible if buildings can be semantically identified in the virtual world. In order to extract building information from photogrammetric data, there should be an approach for semantic enrichment of existing textured meshes [Tutzauer et al., 2019].
- Framework for building use classification and CNN-learned feature investigation: The final objective of this thesis is to provide a framework for building use classification with a pipeline to automatically generate training data input for a DL approach. These training data are obtained by crawling GSV and linking it to cadastral data [Tutzauer and Haala, 2017]. With the aggregated data, it is possible to train a CNN for building use classification. Results of this classification should be interpretable in terms of decisive features for the classifier. Therefore, class-activation maps are leveraged for visual inspection of important feature areas and investigating influences of different representation types on classification results [Laupheimer, 2016, Laupheimer et al., 2018].

1.3 Outline

The remainder of this thesis is structured as follows. Chapter 2 surveys related work in the domains of building and facade reconstruction, semantic segmentation, and classification of urban scenes. Concepts for abstraction and enhancement of urban data representations are discussed as well. Those fields of research are explored in order to embed the subtasks of this work. Consecutive to related work, there are three main parts of this thesis:

- Semi-automatic building reconstruction and building enhancement: Chapter 3 outlines the demands on virtual building representations, as well as the refinement and enhancement of geometric building models. A semi-automatic approach for building reconstruction is presented in section 3.1. Using prior knowledge, building models can be abstracted or refined to make them visually better understandable. A proposal for this task is demonstrated in section 3.2.2.8 and section 3.3.

- Semantic segmentation of urban meshes: Chapter 4 outlines our proposed pipeline to semantically enhance 3D textured triangle meshes. For each triangle within the mesh, multi-scale features are computed and a baseline RF, as well as a 1D CNN, are trained for the face-wise classification into urban categories.
- Building interpretation: Classification of buildings based on street-level imagery is described in chapter 5. First, we propose a pipeline for the automated generation of DL training data (cf. section 5.1). Subsequently, the building classification process (cf. section 5.2), as well as quantitative results, are described. We then go further and investigate CNN-learned features, influences of different data representations and manipulations, and finally propose an application of CNN heatmaps.

In chapter 6, this thesis is concluded, and some areas for future work are proposed.

Chapter 2

Related Work

Virtual city models have been an ongoing research topic for several decades [Biljecki et al., 2015] as there are many applications for digital urban scenes: gaming, urban planning, disaster management, taxation, navigation, and many more. Consequently, there is a great variety of geometric representations of urban scenes. The most immediate format to describe urban data is point clouds. Depending on the data acquisition technique, point clouds are either a direct output (from laser scanning) or the result of photogrammetric derivation from images. The latter is the case when acquiring imagery through aerial (or terrestrial) imagery. Pushed by the advances in SfM [Snavely et al., 2006] and dense reconstruction techniques [Hirschmüller, 2008], multi-view stereo processing is the standard pipeline to produce dense point clouds nowadays [Haala, 2013]. However, for actual urban scene representations, point clouds are just an intermediate product. Most virtual 3D cities are a collection of 3D buildings given as boundary representations (BReps). Following the *Open Geospatial Consortium* (OGC) standard for 3D city models CityGML [Kolbe et al., 2005, Gröger et al., 2012], the geometric LOD of such 3D building representations can be differentiated into four degrees: (1) planar facades with flat roof structures (LOD1), (2) planar facades with detailed roof structures (LOD2), (3) facades with 3D structures (LOD3) and detailed roofs, (4) indoor models (LOD4). Since most of the existing 3D city models have been reconstructed from airborne data, the majority of 3D building models is of LOD2. [Biljecki et al., 2016b] proposed to modify the current five levels of detail in CityGML. They argue the current state is insufficient and ambiguous from a geometrical point of view. Additionally, they provide a good overview of national standards for 3D city models, as well as workflows for producing such.

Parallel to CityGML, industry foundation classes (IFC) has developed as open file format for BIM, to capture the full life cycle of a building. Since BIM is becoming more and more important, there are endeavors to link CityGML and IFC [El-Mekawy et al., 2011] which is beyond the scope of this work. In general, the formats listed here are more likely to be used by niche disciplines. Computer graphics and computer vision mostly rely on mesh representations instead of BReps for 3D scenes. Google Earth, for example, solely uses meshed point clouds for their representations. By doing so, they avoid the derivation of geometrically and semantically interpreted BReps with a defined LOD which, however, are required for all applications that go beyond pure visualizations.

There is a variety of representations for virtual city models. The question however is - *which representation is the 'best' in terms of flexibility, scalability, and comprehensibility?*

CityGML has been around for quite some time, yet it never really prevailed. This is mostly due to the fact that encoding a building, let alone an entire city, into the CityGML data structure is quite cumbersome and thus decoding and visualizing it is not really efficient either. Those drawbacks become even worse for IFC, where the temporal component comes into play additionally.

Point clouds, in terms of a data structure, are easy to handle. In principle, it is a list-like sequence of coor-

ordinates with associated color information (though color availability depends on the acquisition technique). Enhancing point clouds with semantics is straightforward - each entry in the list can be extended by a scalar that represents a specific semantic class. However, instance-level segmentation is more challenging - all related points of an instance must first be identified and then given a unique ID.

Triangle mesh representations are structured similarly. A sequence of coordinates defines all vertices occurring in the mesh. These vertices are indexed, and triangle faces are defined by vertex-index triplets. Color information is either stored per vertex or in the form of texture maps. To leverage the texture maps, for each vertex triplet defining a triangle face, additionally, 2D texture coordinates have to be stored. In order to introduce semantics to the triangle mesh representations, a semantic material (just like the scalar value for point clouds, as mentioned above) can be associated with every face. Instance-level segmentation is easier to incorporate in the mesh structure than for point clouds - essentially a container structure could be introduced, that encapsulates all face indices belonging to the same object. The point cloud and mesh comparison will be briefly taken up in the introduction for chapter 4, where our semantic urban mesh segmentation approach is motivated.

As of now, there is no definitive way to go in the representation of virtual city models, and the question remains open on what should be the standard format, especially when incorporating semantic information. Because there is no final answer yet, different aspects are investigated within this work: Geometric reconstruction (section 3.1) vs. semantic interpretation on different levels. Those levels are semantic superclasses (chapter 4) and fine-grained classification of building use (chapter 5). In addition, we consider different representation forms: Point clouds (section 3.1), BReps (section 3.1.4.2, section 3.2.1.1), textured triangle meshes (chapter 4), and imagery (chapter 5).

Deep learning has found its way into essentially all subfields of computer vision and machine learning. Consequently, the photogrammetry community has embraced these advances as well. In the following, several related work publications are using DL-based approaches, so does the thesis at hand. Hence, the history of artificial neural networks (ANNs) and principles of CNNs are briefly described in section 4.2. Whereas more specific realizations of CNN architectures are discussed in section 5.1.2.1 and section 5.1.2.3.

The following sections cover topics of related work that are relevant for this thesis. Section 2.1 is concerned with building reconstruction, mostly focused on facades of buildings. Since this is not the main focus within the presented thesis, this section is not as extensive as the following ones. The remainder of this section is concerned with city modeling and addresses grammar-based approaches for building generation and completion but also more recent work in the image domain. Finally, section 2.2 is the most extensive section, since the core of our work is located here. Within this section urban data interpretation of several aspects is covered - 2D image-based extraction of information on urban data, identifying semantic classes and objects from 3D data, but also cognitive aspects, that are part our work, too.

2.1 Building Reconstruction and City Modeling

A comprehensive survey of urban reconstruction and modeling pre-2013 has been given by [Musialski et al., 2013] They proposed a subdivision of the urban reconstruction problem into several subdomains. Inspired by this scheme, relevant related work from several of those subdomains is reported here.

Semi-automatic building reconstruction algorithms are disadvantageous in that user interaction is necessary. On the other hand, this simplifies some of the heuristics during the reconstruction process. Hence, there are several techniques relying on user input. [Nan et al., 2010] proposed an approach where users define and manipulate building blocks laid over a LiDAR point cloud. SmartBoxes, rectangular cuboids, are the basic primitive of the reconstruction. Compound SmartBoxes can model more complex, repeated structures. SmartBoxes are snapped to the point cloud data by optimizing a data and contextual term. Through several SmartBox interactions, such as drag-and-drop, grouping, and automatic continuation, building primitives

(windows, balconies, doors, or protrusions) can be reconstructed. A semi-automatic image-based facade modeling system is presented by [Musialski et al., 2012]. Facade elements that exhibit partial symmetries across the image can be grouped and edited in a synchronized manner. The user is in control of the modeling workflow but is supported by automatic modeling tools, where they utilize unsupervised clustering in order to detect significant elements in orthographic images robustly. Recently, GeoFly introduced a new online tool to semi-automatically extract coarse building models from 3D meshes [Geofly, 2018]. A building has to be manually selected, outlines of the building hull are automatically extracted and can be manually further refined. Our approach presented in section 3.1 relies on little user input as well. However, the interaction is limited to a minimum. That is, users have to manually select a facade of interest and can fine-tune some parameters (though this is not necessary), the remainder of the reconstruction and modeling process is automatic. A fully automatic pipeline to extract building models from large-scale point clouds is presented by [Nguattem and Mayer, 2017]. The approach is comprised of random sample consensus (RANSAC) plane fitting and nonparametric Bayesian clustering for the segmentation of meaningful structures. Subsequently, a polygon sweeping is used to fit templates for the buildings, where finally boolean operations clip unnecessary geometry and generate the final result. There is a large corpus of work on the automatic reconstruction of roofs and buildings from aerial imagery and 3D data [Haala and Kada, 2010]. These approaches are, however, limited to LOD2 models. That is, only simple roof structures and, more importantly, flat facades without further geometric or semantic information on facade contents are extracted.

Grammar-based approaches have been widely used in urban reconstruction and modeling. Early works within the context of generating building structures introduced the set/split grammars [Wonka et al., 2003], which operate on basic shapes in 3D and decompose them into shapes from a given vocabulary. Building on this work, computer generated architecture (CGA) Shape, a sequential grammar was introduced [Müller et al., 2006]. CGA Shape incorporates several extensions to the previous split grammar, thus providing a complete modeling system. The more recent follow-up approach CGA++ [Schwarz and Müller, 2015] overcomes several limitations inherent in grammar systems mentioned above, such as simultaneous operations on multiple shapes or leveraging contextual information. These techniques can be considered state of the art now, implemented in the commercial product CityEngine, which is heavily used in urban planning, game development, and the movie industry. Based on a set of given rules, CityEngine synthesizes virtual 3D building models in a specific level of detail and architectural style defined in the rules. Another example of a procedural software tool is Random3Dcity [Biljecki et al., 2016a]. In contrast to CityEngine, this approach uses rules and procedures to generate synthetic data sets of buildings in multiple LODs, encoded in the CityGML format. An extensive survey on procedural modeling for virtual worlds is given in [Smelik et al., 2014]. Generally, formal grammars can be applied to produce virtual 3D buildings from scratch. However, they can also be used to efficiently enhance already existing building representations, e.g., by augmenting the planar facades of LOD2 models with 3D window and door geometries [Becker, 2009] or 3D structures like stairs [Schmittwilken et al., 2006, Schmittwilken, 2011]. [Müller et al., 2007] use mutual information for symmetry detection in rectified facade images. [Wenzel et al., 2008, Wenzel, 2016] detect multiple repeated groups and determine their translations in the image. Thereby a model for compact description of repetitive structures in facades is derived using a heuristic search method. [Ripperda, 2010] uses formal grammars and reversible jump Markov Chain Monte Carlo sampling for the reconstruction of facade structures.

Since defining rules by hand requires specific expert knowledge, some approaches rely on user input. A combination of sketch-based and procedural modeling is presented by [Nishida et al., 2016]. They leverage a collection of simple procedural grammars (snippets) and use CNNs to find the snippets that describe the user sketch best. Non-photorealistic rendered artificial images are used as samples for offline CNN training. As a follow-up, they present an approach to produce procedural building models from a single image with the target building’s silhouette highlighted [Nishida et al., 2018]. CNNs are used to (a) estimate the camera parameters and building mass shape, (b) rectify a facade image and (c) derive facade and window grammars from the rectified facade image.

[Chu et al., 2016] use a DL-based approach to exploit floor plans extracted from rental ad websites and

combine it with Google Street View imagery to build 3D textured building exteriors. [Martinović et al., 2015] propose a complete pipeline from SfM-based reconstruction to semantic labeling of street-side scenes. They perform classification of 3D point clouds as provided by a multi-view-stereo reconstruction into semantic classes *window*, *wall*, *balcony*, *door*, *roof*, *sky* and *shop* using a RF classifier trained on lightweight 3D features. [Gadde et al., 2018] segment 2D images and 3D point clouds of building facades and present empirical results on all available 2D and 3D facade benchmark data sets.

Symmetries in the facade imagery can also be exploited by applying specific shape priors for energy minimization [Teboul et al., 2010, Wenzel and Förstner, 2016]. An fully convolutional network (FCN) architecture is used by [Schmitz and Mayer, 2016] to perform semantic segmentation of building facades. They use the eTRIMS database - building facade images and their corresponding pixel-wise labeling - for fine-tuning their ImageNet pre-trained network. [Mathias et al., 2015] present a three-layered approach consisting of an initial semantic segmentation, the subsequent object detection and a final layer introducing building-specific meta-knowledge by so-called weak architectural principles.

Recently, several promising deep learning techniques for the generation of synthetic data have emerged. Generative adversarial networks (GANs) are architectures consisting of two competing networks - a generator and a discriminator [Goodfellow et al., 2014a]. While the generator produces new examples for a data distribution of interest, the discriminator tries to distinguish between real data samples and fakes produced by the generator. This leads to interesting application possibilities for virtual city models. [Isola et al., 2016] show how conditional GANs, given only an architectural label map, can synthesize realistic facade textures (trained on CMP Facades [Tyleček and Šára, 2013]). They provide more examples like synthesizing urban scene from semantic labels (trained on Cityscapes [Cordts et al., 2016]). [Kelly et al., 2018] propose FrankenGAN, a method to refine coarse building mass models. Style references in the form of texture maps and mass models are fed into a cascaded set of GANs. In a first step, GAN-chains are used to generate texture and label maps for facade and roof, accordingly. Then, super-resolution GANs generate high-resolution textures for facade and roof details, and another GAN-chain is responsible for generating windows. In the last step, the texture and label maps are used to geometrically enhance the mass model (facade, window, and roofs). Building mass models with detailed geometry and textures, both being consistent over a building itself as well as its neighborhood, are the outcome of this framework.

2.2 Urban Data Interpretation

Since this thesis consists of two parts for urban data understanding, this related work section is subdivided accordingly. In section 2.2.1, we report work on interpretation and understanding for building-related content. Our work in this field is covered in chapter 5. Additionally, a core topic of this thesis is located in the 3D geodata domain in terms of scene understanding, described in chapter 4. According related work is presented in section 2.2.2.

2.2.1 Image-Based Building Understanding

The work of [Movshovitz-Attias et al., 2015] uses GSV images for the classification of storefronts, more specifically, the classification into business categories. They create an extensive training data set by propagating business category information with the help of an ontology that uses geographical concepts. For learning, they use a network based on GoogLeNet. With a top-1 accuracy of 69%, they are approximately at human-level. Similarly, [Yu et al., 2015] aim at large-scale detection of business storefronts using Multi-box, a single CNN architecture directly generating bounding box candidates and their confidence scores from image pixel input. Training data consists of two million manually labeled panoramas as provided by crowd-sourcing.

GSV can also be a means to predict socioeconomic trends [Gebru et al., 2017]. They use street-level imagery to detect motor vehicles with their make, model, and year in different neighborhoods and use this information to predict income, race, education, and voting patterns for every precinct. Trading efficiency for accuracy, they relied on deformable part models instead of a then state-of-the-art model such as Faster Regional-CNN (R-CNN) and detected 50 million cars in two weeks.

[Branson et al., 2018] adapt Faster R-CNN to catalog trees from aerial and street-level imagery. The approach consists of two modules - generation of a geographic catalog of objects for a given location and computation of fine-grained class labels for the 3D object at the given location. [Workman et al., 2017] fuse aerial and street level imagery to classify *land use*, *building function* and *building age*. The work of [Kang et al., 2018] is most similar to ours. They are using OSM to extract labels for the GSV imagery, whereas we rely on governmental data. However, our focus is not only on the classification itself. Instead, we want to investigate why specific decisions were made by analyzing CAMs and directly use this information for an application - a non-photorealistic abstract rendering.

2.2.2 Urban Object Segmentation and Classification

A significant amount of work has been done in the domain of image-based land cover/use classification with most promising results of DL based approaches in recent years. For the sake of brevity, the reader is kindly referred to extensive surveys on DL for remote sensing [Zhang et al., 2016, Ball et al., 2017].

In the 2D domain, CNNs are state-of-the-art classifiers that outperform all other approaches. Several architectures have been proposed that are tailored (and thus directly applicable) to semantic segmentation in image space [Zhao et al., 2017, Chen et al., 2018a, Long et al., 2015, Noh et al., 2015, Liu et al., 2015, Yu and Koltun, 2015]. Ideally, DL frameworks for 3D semantic segmentation would directly use 3D data (point clouds or meshes) as input. However, CNNs require Euclidean or grid-like structures. Hence, they cannot be applied to 3D data like point clouds or meshes. In case of point clouds, a natural solution is to migrate to voxel space [Hackel et al., 2016, Huang and You, 2016, Roynard et al., 2018, Landrieu and Simonovsky, 2017], which comes along with memory overhead. Therefore, much effort is put into networks that use sparse 3D convolutions [Graham et al., 2018]. This approach has been successfully applied to urban 3D point clouds [Schmohl and Sörgel, 2019]. Another common approach is to make use of well-performing semantic segmentation in image space [Boulch et al., 2017, Lawin et al., 2017, He and Upcroft, 2013]. To give an example, [He and Upcroft, 2013] segment stereo images semantically, create the point cloud, and back-project the 2D semantic segmentation results to the point cloud. Due to the non-grid like structure of point clouds, classical machine learning approaches still compete with DL approaches. [Weinmann et al., 2015] compute features on multi-scale and multi-type neighborhoods and feed a RF for supervised point-based classification. Our approach takes similar steps but operates on meshes. Furthermore, we leverage a DL approach.

Several current works deal with classification of point clouds using a DL approach like [Wu et al., 2015, Qi et al., 2017, Su et al., 2018]. PointNet++ directly operates on the point cloud [Qi et al., 2017]. The hierarchical neural network uses different abstraction levels of the point cloud for feature learning. Similar to semantic point cloud segmentation, approaches that aim to segment meshes semantically usually do not operate on the mesh directly (Pointnet++ is an exception here). Conventional approaches make a circuit to 2D image space to take advantage of well-performing semantic image segmentation. Instead of directly operating on the mesh, those approaches rather render 2D views of the 3D scene, learn the segmentation for different views and finally, back-project the segmented 2D images onto the 3D surface [Kalogerakis et al., 2017]. Main drawback of such approaches is that they do not make use of geometric information. Moreover, they suffer from occlusions in 2D space. On the contrary, the few existing approaches working on meshes directly, however, often do not use textural information and merely perform mesh segmentation. Therefore, [Valentin et al., 2013] train a cascaded boosting classifier using both geometric features (extracted from mesh) and textural features (extracted from imagery). The classifier’s output is used as unary term for a sub-sequential markov random field (MRF). [Taime et al., 2018] propose a semantic mesh segmentation

solely based on color information provided by imagery and compare their results to approaches solely based on geometric information.

To the best of our knowledge, there are little or no applications of 3D DL to urban meshes yet. Almost all approaches that directly operate on meshes are merely dealing with small-scale toy data sets such as the Princeton Shape Benchmark (PSB) [Shilane et al., 2004]. [Theologou et al., 2015] list several methodologies for mesh segmentation in the domain of computer vision. Most of the presented methods are unsupervised methods applied to small-scale toy data sets focusing on the detection of coherent parts. For example, the methods aim to detect a coherent part of a hand which might be interpreted as fingers by a human operator. It is important to note that these approaches lack in extracting explicit semantic information like ‘finger’. However, semantically enriched ground truth is provided for performance evaluation of the unsupervised methods. Commonly used features are geometric features without exception: average geodesic distance, heat kernel signature, discrete conformal factor, heat mean signature, shape diameter, gaussian curvature, dihedral angle, shape contexts, convexity, convex hull based attributes, principal component analysis (PCA) based attributes or spin images.

Graph CNNs (GCNNs) have been introduced for classification purposes [Bronstein et al., 2016] a few years ago. However, early approaches were limited to transductive classification only [Kipf and Welling, 2016]. In our opinion this is an interesting approach but not reasonable for urban meshes. We seek to learn a model that can be applied to available data without the need to carry data after the training step (inductive learning). Thereby, memory consumption is smaller during inference in case of inductive learning. More recently, GCNNs were used for (semantic) 3D shape segmentation [Yi et al., 2017, Te et al., 2018]. However, these approaches also operate on point clouds only. [Chang et al., 2018] propose the structure-aware CNN (SACNN), a neural network that uses generalized filters (univariate functions). These univariate functions are capable of aggregating local inputs of different learnable topological structures (i.e., different neighborhoods). Approximating the univariate functions with orthogonal base functions makes the univariate functions learnable. By that, SACNNs work with both Euclidean and non-Euclidean data. First results on common toy grid-like structured data sets like MNIST [LeCun and Cortes, 2010], as well as non-grid like structured data sets show that SACNNs outperform classical CNNs.

[Huang and Mayer, 2015] use imagery together with elevation data for urban scene classification by means of relative features. They use L^*a^*b color space because of the independent channel for lightness/luminance, and define vegetation - due to its comparably robust color appearance - as reference class for the relative feature computation. They propose a superpixel-like patch-wise classification scheme. Our approach for semantic segmentation (cf. chapter 4) is similar in that we are using triangles and their surrounding neighborhood for feature calculation and leverage terrain information as well. The same data basis as in [Huang and Mayer, 2015] is used in [Zhang et al., 2017], where they investigate fusion strategies of such multi-modal data in FCNs. They show, for instance, that for elevation input, shallow layers are sensitive to impervious surfaces and low vegetation. For image input, shallow layers are sensitive to small objects such as cars, whereas deeper layers are more sensitive to objects with intricate textures, covering larger image regions. Thus, they conclude, that early fusion does not lead to satisfying results, and hence, they propose a respective fusion strategy. A joint approach to refine geometry and semantic segmentation of meshes is presented by [Blaha et al., 2017]. The mesh shape is deformed with variational energy minimization, while semantic labels are updated with MRF inference. Initial semantic segmentation maps are obtained from a MultiBoost classifier. The recently published TextureNet is a promising approach for semantic segmentation of 3D meshes [Huang et al., 2018]. They define a 4-rotational symmetric field (4-RoSy) as a domain for surface convolutions. By mapping a geodesic neighborhood of a surface to two-dimensional parameterizations and defining a specific 4-RoSy convolutional filter, they are able to directly incorporate color texture maps from which features can be learned.

Most similar to our work is the semantic mesh segmentation proposed by [Rouhani et al., 2017]. They gather faces of the input 3D textured mesh (generated from a MVS point cloud) into so-called superfacets in order to reduce the calculation effort. In other words, they use segmentation as preprocessing before

performing the semantic segmentation on the superfacets. Prediction is made via a RF using geometric and photometric features. For each superfacet, the class and its similarity to the neighboring superfacet is predicted. The similarity information is used to assign weights to the pairwise-potential of a MRF and accounts for contextual information between the classes. In contrast to the work of [Rouhani et al., 2017], this work performs inference on each face instead of a bundle of faces (superfacet). However, our features are multi-scale features considering several spherical neighborhoods (SPNHs), and thus, the achieved labeling is smoothed implicitly. We refer to this as implicitly smoothed per-face classification through a larger support region. Using this implicit smoothing, treating each face as independent of all others can be avoided. We assume that this implicit smoothing reduces the need for an explicit smoothing via a MRF. Nonetheless, we perform explicit MRF smoothing (cf. section 4.4) and analyze which unary term is better suited for MRF smoothing. Compared to the work proposed by [Rouhani et al., 2017], we use more features per face and consider a more complex semantic segmentation task with more classes (ours: 8 classes, theirs: 4 (optionally +1) classes). Our features are given in table 4.2.

Our presented work adapts the approach of [George et al., 2017], which uses a multi-branch 1D CNN for 3D mesh segmentation of the PSB. To stress this point - their proposed network aims for mesh segmentation with the objective to extract coherent parts of a 3D mesh. They do not provide explicit semantic information. On the contrary, we leverage this network in order to achieve a semantic segmentation of urban meshes. For each triangle, a multi-scale feature vector is computed and serves as input for the CNN. Different scales are fed to respective branches.

The negligence of semantic mesh segmentation in the domain of urban scenes may be due to the absence of labeled benchmarks. Several mesh benchmarks exist for indoor scenes [Armeni et al., 2017, Hua et al., 2016, Dai et al., 2017] or for single objects [Shilane et al., 2004]. Available labeled urban data sets, however, are 3D point clouds [Wichmann et al., 2018, Hackel et al., 2017, Niemeyer et al., 2014] wherefore research focuses on semantic segmentation of 3D point clouds [Wu et al., 2015, Su et al., 2018]. The work presented in [Rouhani et al., 2017] is based on an unpublished labeled mesh. Therefore, we create our own labeled ground truth data (cf. section 4.1). In order to cheaply generate training data, we additionally leverage the SynthCity data set provided by [Cabezas et al., 2015] (cf. section 4.5.3).

2.2.3 Cognitive Aspects in the Architectural Context

Cognitive aspects within the area of virtual city models are a topic that has not been covered in such detail as compared to the aforementioned subjects. However, it is important, since humans are the direct recipients and consumers of virtual geodata content. Findings of Gestalt theory play an important role when dealing with man-made structures. Gestalt theory was established by [Wertheimer, 1923], the founders of Gestalt psychology. The core of Gestalt theory is the idea of grouping. Humans tend to order things (here more specifically shapes) in a structured, regular, symmetrical way and from this fact the Gestalt laws can be deduced. The most important rules are the Gestalt laws of *continuity*, *similarity*, *symmetry*, and *proximity*.

[Li et al., 2004] exploit Gestalt principles for the grouping and generalization of 2D building footprints. [Michaelsen et al., 2012] refer to Gestalt-based groupings for the detection of 2D window structures in terrestrial thermal imagery. Within the wide field of visualization approaches, [Adabala, 2009] present a perception-based technique for generating abstract 2D renderings of building facades, and [Nan et al., 2011] apply conjoining Gestalt rules for the abstraction of architectural 2D drawings.

Approaches on the human perception of geometric building representations, which are not restricted to 2D structures or 2D visualizations but, instead, are directly located in 3D space, are often developed in the context of cartography. Prominent representatives are provided by [Glander and Döllner, 2009] or [Pasewaldt et al., 2014], who use cognitive principles for generating abstract interactive visualizations of virtual 3D city models. These approaches, as well as most others dealing with perception-based abstraction of virtual 3D

cities, focus on emphasizing landmarks while generalizing and suppressing buildings that are supposed to be unimportant from a tourist’s point of view.

All the works mentioned have one thing in common: They integrate perceptual principles in their methods for the recognition, generalization, or abstraction of geometric building structures in order to reveal or emphasize building-related information. These perception-based methods, however, are all more qualitative than quantitative operations. That means quantitative statements about the degree to which the respective information can be perceived by a human, or tasks like, for example, searching for the best abstraction to achieve a certain degree of perceptibility are not supported.

Existing attempts to quantify the human perception of geometric objects are closely linked to Gestalt principles and, therefore, limited to simple 2D structures. [Cao et al., 2007] propose a probability measure to quantify the meaningfulness of groupings in cluster analysis for 2D shape recognition. [Kubovy and Van Den Berg, 2008] provide a probabilistic model of Gestalt-based groupings by proximity and similarity on regular 2D patterns. In the context of point cloud segmentation, [Xu et al., 2017] proposed a voxel-and graph-based approach using Gestalt theory. They use the perceptual grouping *laws of proximity, similarity* and *continuity* as a clustering criterion.

To the best of our knowledge, the evaluation of complex 3D building geometries with respect to their perceivable semantic information content, i.e., the quantification of perceptual insight, has not been addressed yet. Based on our user study on the human perception of building categories (cf. section 3.2), we took a first step in this direction.

There has also been recent work on cognitive aspects in the field of DL. [Kim et al., 2019] have shown that even CNNs follow the Gestalt principles to some extent. In particular, they investigated the *law of closure*, which states that human visual perception tends to complete fragmented shapes. In other words, the perception system closes gaps in order to perceive objects a whole. A shape bias case study for deep convolutional neural network (DCNN) is presented in [Ritter et al., 2017]. Their findings suggest that one shot learning models preferably categorize objects according to shape rather than color. Humans exhibit the same bias. In the field of urban data processing, this behavior is desirable because the shape of an object is arguably the strongest indicator for a certain semantic class affiliation. However, for simple shapes like planes, semantic distinction becomes hard, and thus, color information is essential. The influence of texture on building model classification is discussed in section 3.2.2.7. Additionally, we show the influence of radiometric information on semantic segmentation of urban scenes in section 4.5.

Chapter 3

Building Models: Refinement, Demands, and Enhancements

Within this chapter several aspects of virtual building models are addressed. In order to use tools for urban planning, construction monitoring, or emergency simulations, it is necessary to generate solid models from point clouds. As discussed in section 2.1, for coarse building representations, this is state of the art and can be performed in a fully automatic way. Reconstructing buildings with actual facade elements and thus generating LOD3 models, is an ongoing field of research. In section 3.1, we present an approach to reconstruct solid LOD3 models from 3D point clouds. This pipeline combines a data-driven algorithm for reconstruction (bottom-up) with a model-driven approach for grammar-based completion (top-down). The bottom-up part is responsible for the extraction of semantics, that is, building primitives like windows and doors, from point clouds and consecutively the derivation of a facade grammar. Whereas the top-down algorithm uses this facade grammar to enhance a coarse input building model geometrically and semantically. LOD3 models are the result of this pipeline and with them, semantics are introduced on a facade level - positions of walls, doors, windows, and roofs are encoded in this structure.

However, the scope of this thesis goes beyond facade semantics and seeks to obtain semantics on the building level, i.e., *which use type does a building belong to?* The generation of LOD3 is therefore the first step of introducing semantic information in virtual city models. Consequently, the next step is to generate semantic information on a higher level. To that end, it is necessary to consider the human factor. Section 3.2 introduces an extensive study on the human comprehension of building categories. The knowledge from this study can be used in various way. Most importantly, we try to capture features that are, from a human perspective, characteristic for certain building categories. One application of this knowledge is an approach for perception-based abstraction (cf. section 3.2.2.8) that retains key features and drops unimportant elements of a model.

Once essential characteristics of different building categories are extracted, this information can be further used to semantically enhance existing building representations. A pipeline for grammar-based enhancement and abstraction of building models is proposed in section 3.3. We depict a way to estimate the building use category of yet unknown building models by means of the previously extracted feature sets. This information can be used to enhance the visual appearance of a model by perceptual rules in a way that it becomes more understandable for humans. Additionally, feeding this information into category-specific rule sets allows enhancing the model's appearance, thus becoming visually better understandable. Together, this section describes three parts that contribute to the pipeline of processing raw point clouds to solid models and adapting those building representations for better perceptual insights.

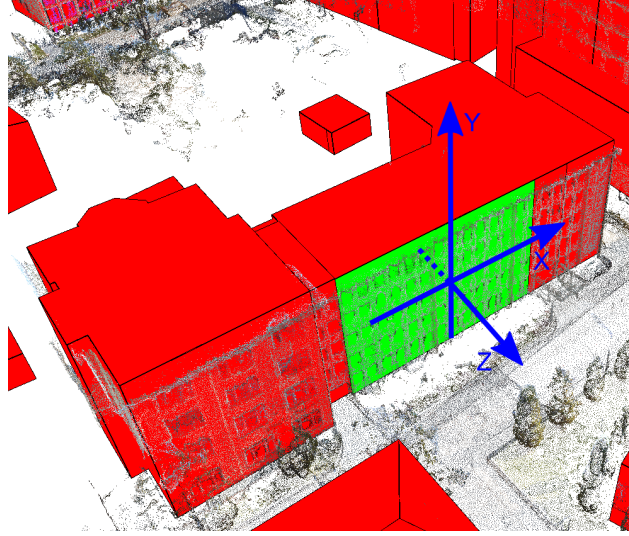


Figure 3.1: Selected facade of interest within the semi-automatic reconstruction interface. Blue axes depict the local FCS.

3.1 Semi-Automatic Building Reconstruction

The work presented here uses point cloud information to gradually modify coarse building models. In the following section the developed semi-automatic geometric reconstruction process is described. Section 3.1.3 discusses a radiometric segmentation approach for point clouds with limited geometric accuracy. Finally, section 3.1.4 briefly addresses the modeling approach as part of previous work and outlines the principles for grammar-based building generation in general since this will be subject in section 3.3.

The basis for this approach is previous work introduced by [Becker, 2011]. It was extended in several ways to be more scalable and robust. Previous work relied on custom data formats and was only feasible to handle a single facade point cloud and building model. Core of the framework is a graphical user interface (GUI) that is capable of handling two input sources: (1) large point clouds for complete city areas in the LAS¹ format. (2) a CityGML² model covering approximately the same area as the point cloud. The pipeline is semi-automatic in the way that users have to select a facade \mathcal{F} from a building of interest by clicking on it. In order to perform geometric reconstruction, it is necessary to define a buffer around the selected facade. This buffer defines a three-dimensional clip box around the facade and determines which elements of the point cloud \mathcal{P} are selected for facade reconstruction:

$$\mathcal{P}_{\mathcal{F}} = \mathbf{X}_{\mathcal{F}} - \Delta\mathbf{X}_{lower} \leq \mathbf{X}_{\mathcal{P}} \leq \mathbf{X}_{\mathcal{F}} + \Delta\mathbf{X}_{upper} \quad (3.1)$$

Where $\Delta\mathbf{X}$ is the lower and upper bound in X, Y, Z direction of the facade coordinate system (FCS), respectively. Figure 3.1 depicts a facade selected by the user in green and the axes of the local FCS in blue. As a by-product, the extracted facade points can be exported, which culminated in the tool used in the benchmarking process for [Cavegn et al., 2014]. $\mathcal{P}_{\mathcal{F}}$ serves as input for the reconstruction process which is outlined in the following.

¹LAS is a 3D file format introduced by the *American Society for Photogrammetry and Remote Sensing* (ASPRS). Originally developed for the exchange of LiDAR data, it has become a de facto standard for point clouds stemming from either LiDAR or DIM.

²CityGML is an OGC standard for representing virtual 3D cities and landscapes, encoded in an XML structure. It offers the possibility to store geometry, semantics, topology, and appearance in different LODs.

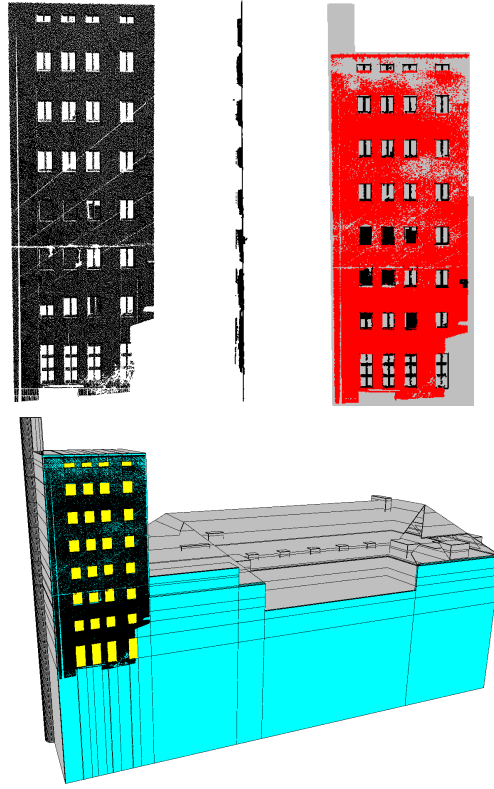


Figure 3.2: Top from left to right: TLS facade point cloud, lateral view, segmentation into *facade points* and *non-facade points*. Bottom: Result of cell decomposition with reconstructed primitives (windows) depicted in yellow.

3.1.1 3D Cell Decomposition

Initially, a plane is fitted to $\mathcal{P}_{\mathcal{F}}$. Based on this facade reference plane, the facade is segmented into actual *facade points* and *non-facade points* by distance thresholding. Subsequently, horizontal and vertical edge points located at the borders of windows and doors are searched within the *facade points*. There are four different kinds of edge points: upper, right, lower, left. An upper (*right/lower/left*) edge point is considered to be found if inside a given buffer no further *facade point* is located below (*left to/above/right to*) the query point. Testing all *facade points* whether they belong to one of the four edge-point categories delivers window edges in both horizontal and vertical direction. By means of the extracted edges and the estimated window depth, the coarse building model is divided into three-dimensional cells. The upper row of figure 3.2 depicts a TLS point cloud of a building with tower-like structure in Zürich, its lateral view, and the segmentation into *facade points* and *non-facade points*, respectively. The lower part of this figure shows the detected window edges and the resulting cell decomposition. Initially, the reconstructed cells purely have geometric information. The following section will discuss how these cells can be semantically classified, as depicted by the yellow rectangles (which are actually 3D cells) in figure 3.2.

3.1.2 Classification of 3D Cells

There are two types of 3D cells, facade cells and building cells. Building cells are not of interest, since they do not contain primitive information. They can easily be discarded as they do not border on the front facade. Facade cells can be further distinguished into wall and window segments. By analyzing the percentage of

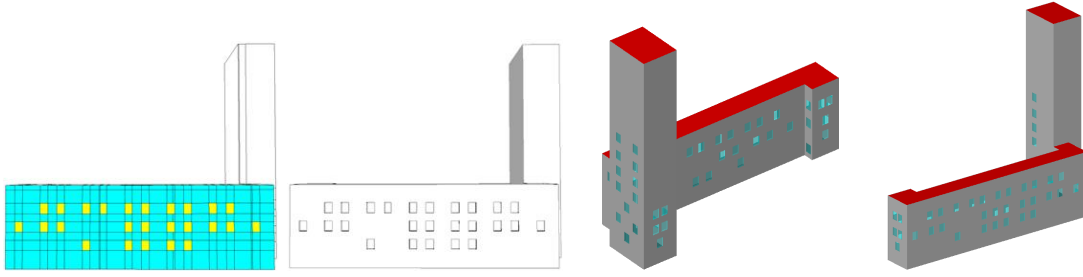


Figure 3.3: From left to right: Cell decomposition with classified cells of facade shown in the middle of figure 3.5, reconstructed model, and two different views of the enriched CityGML LOD3 model.

area covered by facade points per cell, a classification into facade and wall cells is possible. The classification is accomplished by means of a point availability map. Each pixel of this low resolution binary image defines a grid element on the facade. Black pixels in the point availability denote grid elements that contain 3D points, those are wall pixels. Whereas white pixels do not contain any 3D point measurements and are therefore considered window pixels. If the majority of a 3D cell consists of wall pixels, this cell is classified as wall cell. Otherwise, if less than 10 % of a cell consists of wall pixels, this cell is classified as window cell. This way, the complete facade can be reconstructed into wall parts and window/door parts. Figure 3.3 gives an additional example of a 3D cell decomposed fabric building with classified window and facade cells. The two subfigures on the right side depict the enriched LOD3 model generated by the grammar-based approach, which will be discussed in section 3.1.4.

3.1.3 Radiometric Segmentation

In contrast to the above-shown LiDAR examples, point clouds generated from DIM often show significantly more variance in geometric accuracy. Reflective surfaces and areas with homogeneous textures cause problems in the matching process and thus lead to noisy data [Remondino et al., 2013]. Additionally, in urban scenarios difficult viewing angles with occluded areas and acquisition platforms in rapid motion might lead to facade point clouds with clutter in depth direction. For the task of facade reconstruction, the downside of LiDAR - generating holes in areas with no proper reflexion (e.g., windows) - is actually beneficial. 3D data generated from DIM, on the other hand, leads to problems in such areas. Due to limited geometric accuracy in depth specifically, *facade points* and actual primitives might become coplanar and thus not separable from the facade anymore. Geometric reconstruction will presumably fail in these regions. However, radiometric information that is always inherent in point clouds generated from imagery can help to overcome those difficulties.

The approach for radiometric segmentation is developed under the assumption that homogeneous facade parts exhibit the same or very similar color schemes. On the one hand, this assumption is based on architectural and symmetrical aspects. On the other hand, it is often conditioned by the construction material used. Thus, the semantic segmentation tries to identify topologically coherent regions, but also those that are disjunct, yet radiometrically similar. To enable building reconstruction for point clouds with limited geometric accuracy, the pre-segmentation into *facade points* and *non-facade points* during the geometric reconstruction step has to be bypassed. Hence, the outcome of radiometric segmentation is a labeling into those two respective classes.

In the first step, RGB color information is transformed into HSV space. Due to its robustness against changes in lighting conditions, we only consider the hue channel. In order to perform a color-based clustering of the point cloud, all points are mapped to a small number of bins of a hue histogram. Each bin of the histogram can be considered a cluster. Since points belonging to the facade plane are expected to be coherent in

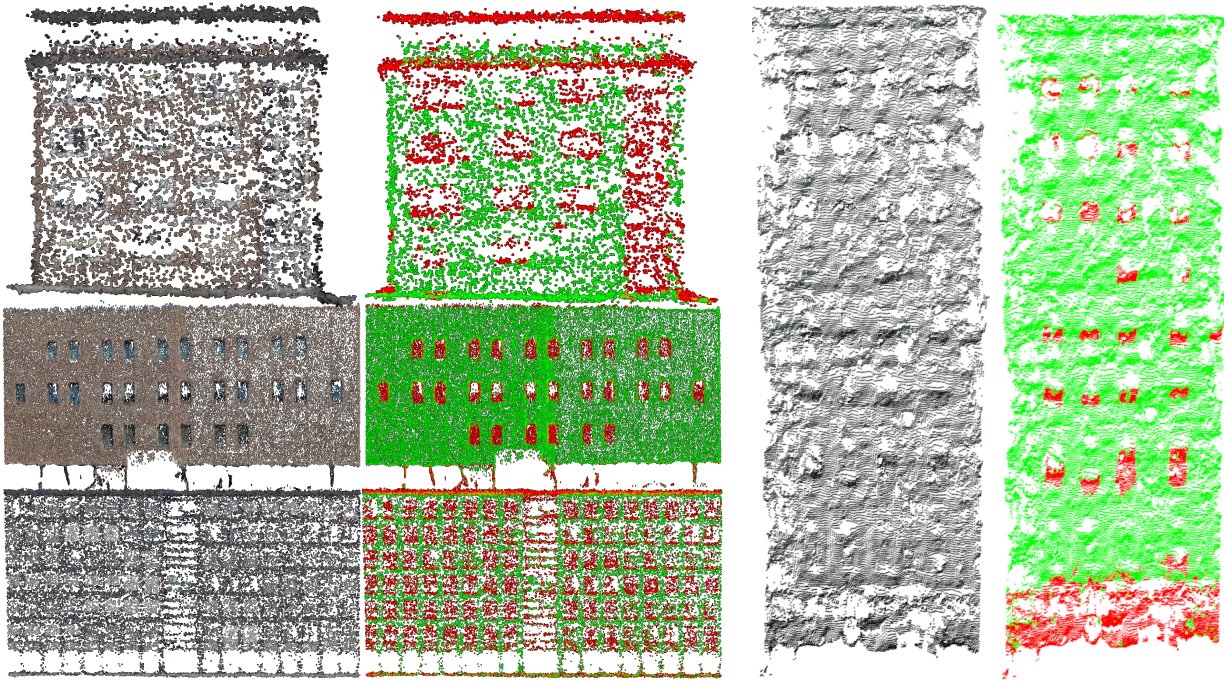


Figure 3.4: Left column: Colored point cloud from mobile mapping data. Right column: Segmentation into *facade points* and *non-facade points*.

Figure 3.5: Left: Point cloud of tower shown in figure 3.2 generated from airborne imagery. Low radiometric information and geometric accuracy. Right: Corresponding segmentation.

terms of colour and planarity, they are considered to form a large cluster and hence the number of bins is chosen to be small (3 in most of our experiments). Additionally, for every point, local features are computed by considering its local neighborhood using k-nearest neighbors (k-NN) search, with k around 10 points in our experiments. The features are local hue difference and planarity. Local hue difference is obtained by computing the hue difference of the current point to all its neighbors and extracting the median of those differences. Median is preferred over mean to give robustness against outliers in hue space. Local planarity is determined by a RANSAC based plane fit to the k-NN. The distances of all points of the local neighborhood to the plane fit is calculated. Again, the median distance is extracted and serves as planarity feature for the current point of interest. Subsequently, for each cluster (that is, each bin of the histogram) we determine its mean hue and mean planarity. If both, mean hue difference, and mean planarity per cluster are below pre-defined thresholds, homogenous facade plane points are supposed to be found. This relatively simple approach works quite well for most of the facades as shown in figure 3.4. Building primitives, roofs, as well as indents can be distinguished from the main facade plane. Figure 3.5 depicts the same building as shown in figure 3.2. This point cloud, however, was generated from oblique airborne imagery. Both, geometric accuracy and radiometric information are very limited. In this case, the segmentation quality decreases. Nevertheless, with an adapted cluster size in contrast to figure 3.4, most of the primitive outlines are still detected. If main facade plane and primitives are radiometrically too similar, however, this segmentation approach is more likely to fail.

In summary, the radiometric segmentation module can be embedded in the reconstruction pipeline and serves as a bypass of the plane fitting based pre-segmentation discussed in section 3.1.1 if the geometric accuracy is too low. Once the distinction between *facade points* and *non-facade points* is made, the pipeline can proceed to the next step of 3D cell decomposition.

3.1.4 Building Modeling

With the previously classified primitives and a corresponding LOD2 model of the building of interest, it is possible to perform a grammar-based modeling. The idea is to enhance the coarse building model and complete it in areas where no point cloud information is available. Thus, lifting the input LOD2 to an enriched version of LOD3. First, the facade is divided into floors by horizontal cutting planes. If the number of floors is stored as metadata in the CityGML data, this information could be used instead. Horizontal and vertical divisions within each floor segment the facade into disjunct tiles. A tile is either a homogeneous wall part or contains a geometry object. Thus, a sequence of wall and geometry tiles emerges. This sequence can be considered correct if it begins and ends with a wall segment and has alternating tile types in-between. In other words, the same tile type must not appear consecutively. Using a sequential cluster approach, the tiles are classified through their size and type. Different tile types with correspondingly assigned symbols are the outcome of the clustering process and the relation or rather sequence of symbols is further investigated. The objective is to compress the symbols by replacing redundant sequences by a new symbol. Thereby hierarchical relations between the tiles are detectable. These relations can be expressed by replacement rules. Additionally, production rules can be derived. Afore-gained knowledge can be used for the modeling process. The entirety of production- and replacement rules depicts the facade grammar. Section 3.1.4.1 describes the derivation of a (facade) grammar in more detail.

3.1.4.1 Grammars for Buildings

The concept of grammars is generic and can be used for a variety of tasks. The general concept is therefore commonly addressed as formal grammar. A formal grammar consists of symbols (alphabet) and a set of production rules to generate content (syntax). In a formal language, this content would be a string. In our case, it is 3D objects. The notation of a grammar is as follows:

$$G = (N, T, P, S) \quad (3.2)$$

Where N is a set of non-terminal and T a set of terminal symbols. Those two sets are disjunct and terminals cannot be further decomposed. Whereas non-terminals can be replaced and expressed in terms of a set of terminals. S is called axiom, a non-terminal defining the start point. P describes the set of production rules which are probabilistic and context-sensitive in our case. They can be expressed as follows:

$$id : lc < pred > rc : cond \rightarrow succ : prob \quad (3.3)$$

Where id is a consecutive rule number, lc is the left context, $pred$ is the predecessor symbol, rc is the right context, $cond$ is a condition under which the rule is applied, $succ$ is the successor symbol, and $prob$ is a certain probability for the rule to be applied. Within the context of generating building structures the so-called Split-Grammar [Wonka et al., 2003] and later as a continuation, the CGA Shape Grammar was developed [Müller et al., 2006]. Later in this work, such a grammar will be used to define specific rule sets for each building category and will then be used to generate enhanced versions of the input building (cf. section 3.3). Figure 3.6 shows an exemplary floor with different windows. Each window is investigated on its geometry, and thereby instances with same spatial extents are grouped, leading to the notation w_i , with $i = 0, 1, 2$, denoting that there were three window types found. They represent terminal symbols of a grammar (cf.

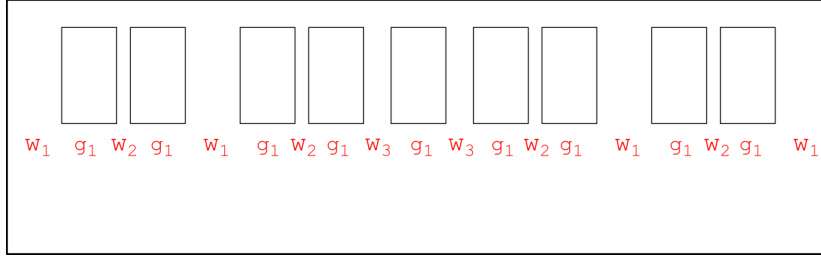


Figure 3.6: Exemplary floor of a facade with geometry (window) tiles denoted as g_i and wall tiles denoted as w_i . This facade string can be further compressed by merging sequences (cf. equation (3.4))

equation (3.3)). If we denote the floor in figure 3.6 as Fl_0 and derive repetitive patterns as Ψ_i , then the sequence can be described as follows:

$$\begin{aligned}
 Fl_0 &\rightarrow w_1 \ g_1 \ w_2 \ g_1 \ w_1 \ g_1 \ w_2 \ g_1 \ w_3 \ g_1 \ w_3 \ g_1 \ w_2 \ g_1 \ w_1 \ g_1 \ w_2 \ g_1 \ w_1 \\
 Fl_0 &\rightarrow w_1 \ \Psi_2 \ w_3 \ g_1 \ w_3 \ \Psi_2 \ w_1 \\
 &\text{with} \\
 \Psi_2 &\rightarrow \Psi_1 \ w_1 \ \Psi_1 \\
 \Psi_1 &\rightarrow g_1 \ w_2 \ g_1
 \end{aligned} \tag{3.4}$$

In this case, the terminals are given by $T = \{w_1, w_2, w_3, g_1, g_2\}$ and the non-terminals $N = \{W, G, \Psi_1, \Psi_2\}$. From those collections productions rules for a facade grammar can be inferred:

$$\begin{aligned}
 p_0 : F &\rightarrow W* \\
 p_1 : W : cond &\rightarrow WGW \\
 p_2 : G : cond &\rightarrow \Psi_i : P(x|p_2) \\
 p_3 : G : cond &\rightarrow g_i : P(x|p_3) \\
 p_4 : lc < W > rc : cond &\rightarrow w_i : P(x|p_4)
 \end{aligned}$$

p_0 horizontally partitions the whole facade F into a set of floors, where $W*$ denotes multiple repetitions of wall tiles. Rule p_1 reflects the vertical split of a floor into tiles. The previously detected structures can be transferred to a production rule of the form p_2 . Non-terminal geometry tiles G are thus replaced by structures Ψ_i . Similarly to p_2 , p_3 and p_4 are instantiation rules. They define production rules for geometry terminals g_i (p_3) and wall terminals w_i (p_4), respectively. Rules p_2, \dots, p_4 contain a conditional probability, where x denotes the position on the facade and p_4 additionally incorporates context-sensitivity as depicted in equation (3.3). For further details on this grammar-based concept the reader is referred to [Becker et al., 2008, Becker and Haala, 2009, Becker, 2011].

3.1.4.2 Synthetic Facades and Model Completion

The facade grammar gathered from the data driven reconstruction can be used to model the remaining facades of a building synthetically. Production rules are applied for each of the already determined number of floors. Respective starting point is the center of the facade since main appearance features are frequently located there. This way, first the middle, and subsequently the remaining facade parts are modeled with new tiles from left to right. If there are multiple production rules possible for a new tile, a corresponding rule is selected probability-based (*prob* statement in equation (3.3)). The final output is a CityGML LOD3

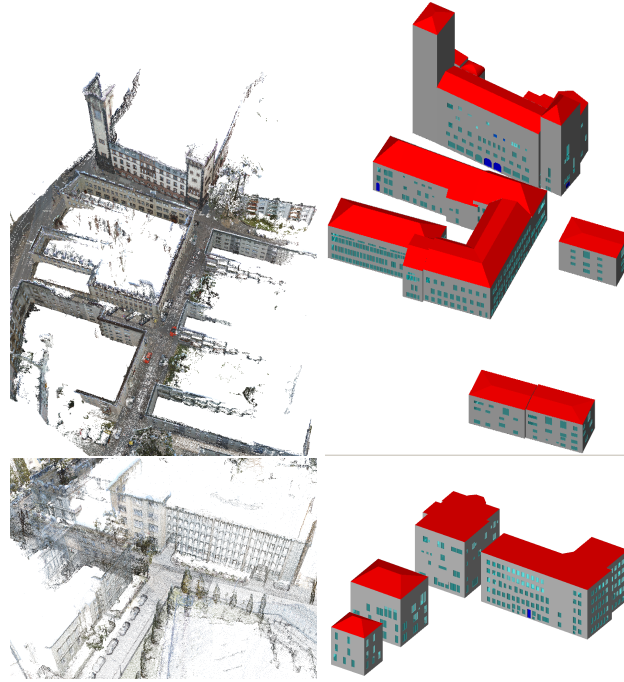


Figure 3.7: *Left column:* Point cloud inputs to the reconstruction and modeling approach. *Right column:* Respective CityGML results after grammar-based completion. Roofs are depicted in red, facade in gray, windows in semi-transparent turquoise, and doors in blue.

model as shown in figure 3.7. Using CityGML has several advantages: (1) Semantic information is explicitly tagged as such in the XML based format. (2) Due to the hierarchical data structure, there is a direct link between each wall and its associated primitives. (3) If the input model to the algorithm is geolocated, the output will be as well. Even more, as shown in figure 3.8, the CityGML model can be directly exported to Google Earth for instance. To conclude, in this section we presented an approach that uses point clouds from LiDAR or DIM together with a coarse building model in CityGML LOD2 format for a semi-automatic approach to lift the input model from LOD2 to an enriched LOD3 version.



Figure 3.8: Examples of geo-located CityGML buildings, produced by the semi-automatic facade reconstruction and modeling approach.

3.2 Human Understanding of Building Models

The symbiosis of a geometrically reconstructed as well as semantically interpreted building model is a powerful basis for further, more abstract processing. In the previous section we focused on the geometric reconstruction and relied on grammars for the completion of buildings. Within this section we gather information on how humans actually perceive buildings. Besides providing geometric information on the represented buildings, virtual 3D cities can also serve as medium to visually communicate urban- or building-related semantic information. In this case, 3D representations should enable users to comprehend the respective semantics fast and intuitively without wasting mental workload on non-relevant information. The degree of insight that people obtain via the visual communication of semantics strongly depends on what kind of geometric 3D building representations are used. Geometric 3D representations which fit people's visual habits and urban legibility can help to achieve a quick and accurate understanding of urban spatial information. Due to the multitude of different sensors, algorithms and modeling concepts used for acquiring and processing geodata in urban areas, virtual 3D cities can be based on various data types and ways of modeling (e.g. unstructured 3D point clouds, meshed surfaces, textured or non-textured volumetric 3D models with different levels of detail and abstraction). However, the question *Which of these geometric 3D representations is, given a context, best suited to enable a maximum understanding of the information that is intended to be transmitted?* is still an open problem.

Depending on the application and the requirements going along with it, the provision of virtual 3D cities may involve considerable investments with respect to costs, time and expertise for data acquisition and processing. Thus, it is highly unsatisfactory that it is not known beforehand whether the desired degree of understanding can be reached by means of the generated virtual 3D building representations, or whether a smaller solution would have been sufficient. Questions like these are of special relevance for developers of systems that work with 3D virtual cities (e.g. 3D navigation systems, virtual reality applications, computer games etc.). The basis for all that - profound knowledge on the human's ability to understand semantics from 3D building structures - is still missing. This work provides an important first step by dealing with the identification of perceptual aspects which are relevant for the understanding of semantic information inherent in geometric 3D building structures.

Generally, it depends on the application as to which specific building-related semantic information needs to be understood by the user. Semantic issues of interest may be: building category, architectural style, historical relevance, state of preservation etc. Out of these, we will exemplarily address the semantic issue *building category* which covers basic semantic information: Being able to quickly understand the category of buildings when moving through virtual 3D cities means support for various applications (e.g. navigation, real estate management, or spatial marketing) as it will help users to orient themselves and enable intuitive and efficient exploration. The following section presents a user study we developed and conducted in order to reveal the required knowledge about how a human understands building categories from geometric 3D building representations. In more detail, we will focus on two questions:

1. Which representation type is for which building category the most suitable?
2. Which geometric building properties and structures are relevant for the perceptibility of a particular building category?

Moreover, we will demonstrate how the derived knowledge about perceptually relevant geometric structures can be applied to improve the interpretability of 3D building abstractions.

3.2.1 Development and Conduction of User Studies

The data basis and the setup of the user study are described in the following section. Applied evaluation metrics are presented consecutively. The results of the study as well as a first application scenario

showing how the derived knowledge can be used for perception-aware abstraction processes are presented in section 3.2.2.

3.2.1.1 Data Basis and Setup

The category of a building is reflected in both geometric building properties (among others: building size, roof shape, size, number and arrangement of windows) and textural information. In order to separate these influences in the best possible way, the following representation types are used within the study: (a) untextured LOD3 models for analyzing isolated influences of geometric building and facade properties, (b) textured meshes/LOD2 models (from Google Earth (GE)), as well as (c) images from Google Street View for analyzing influences of textural information. Each of the three representation types are shown to the user in a way that at least two facades per building are visible. To avoid the building's environment having influence on the user's decision, only the building itself is shown; the environment of the building is not represented. Within the study, users have to classify buildings into six characteristic building categories extracted from the ALKIS feature catalogue [ADV, 2018]:

- *one-family building* (OFB)
- *multi-family building* (MFB)
- *residential tower* (RT)
- *building with shops* (BWS)
- *office building* (OFF)
- *industrial facility* (IF)

The buildings to be classified are randomly taken from German cities (mostly Stuttgart), i.e., between 15 and 20 candidates of each building category are selected. For all these candidates LOD3 models have been modeled manually. For 60 % of the buildings, additionally, textured meshes/LOD2 models from Google Earth and/or images from Google Street View are provided. Table 3.1 gives examples of the building categories and representation types presented to each user. The user study is conducted as an online survey for the test person's convenience as well as faster evaluation reasons. At the beginning of the survey, some general information about the user is obtained, namely:

- Gender
- Age
- Graduation
- Subject of study
- Nationality
- Previous experiences in 3D virtual reality worlds (computer games, Google Earth, CAD modeling etc.)

After the general information, users have to perform the actual building category classifications. All in all, 165 different building representations are tested. The representations are shown to the users in random order. After the classification of each representation, the test persons have to rate their level of certainty (reaching from *Very Uncertain* to *Very Certain* in 5 selection options). Based on the self-assessment for each classification, a relation between user correctness and certainty can be examined. This metric can give further information about whether the user is aware of being wrong in the current classification.





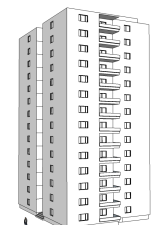
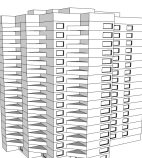

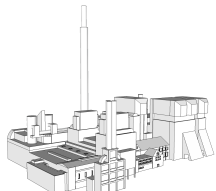
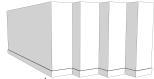


	LOD3 models				Textured Meshes	Images (GSV)
OFB						
MFB						
RT						
BWS						
OFF						
IF						

Table 3.1: Different building categories and representation forms presented to the users.

building footprint	number of floors
floor height	total building height
number of windows per facade	mean window surface area
window-to-wall-surface ratio	number of entrances
mean entrance surface area	number of balconies
mean balcony surface area	different window arrangement in ground floor
different window sizes in ground floor	different window shapes in ground floor
different ground plan in ground floor	roof complexity

Table 3.2: Building features used for the study.

3.2.1.2 Evaluation Metrics

The actual reference category for each model is obtained by extracting the type of use from the digital city basic chart and 3D data from the City Surveying Office of Stuttgart. To compare differences between the user’s classification and the actual ground truth, all surveys are evaluated, and typical classification quantities such as confusion matrix, commission/omission errors and user’s/producer’s accuracy are computed. Moreover, in order to obtain deeper knowledge on the user’s perception, for each building category, the ground truth buildings are compared to the classified buildings. Aiming at quantifiable results, this comparison is based on computing geometric building properties inherent in LOD3 models.

The properties given in table 3.2 are evaluated: The *window-to-wall-surface* ratio is given as the quotient of mean window surface area divided by the mean facade area (wall surface minus windows, doors etc.). Considering the property *different appearance of the ground floor (GF) as compared to the remaining floors*, 4 different aspects are analyzed: *different arrangement, size and shape* of windows in GF, as well as *different ground plan in GF than in other floors*. Each of these 4 aspects can take either the value 1 (different) or 0 (equal). Thus, the 4 mean values, which are computed for all representatives of a building category, express the degree of geometric difference between ground floor and remaining floors. Considering the property *different roof types*, we discriminate between five different roof shapes: flat, saddle, hipped, monopitch and complex. Correspondingly, a roof complexity value ranging from 1 (simple) to 5 (complex) for each building category is computed as the weighted mean, with the weights being the occurring amount of each roof type within the class.

Based on these metrics, the discrepancy between ground truth and the user’s perception is investigated. As a first step within this evaluation, the ground truth data is analyzed. For each building presented to the user in the test, the above-stated features are determined. Since every building has been labelled into one of the 6 building categories presented in section 3.1, it is possible to calculate mean values of the features for each building category. These values can be considered representative for the respective category.

In a second step, the 6 building categories are set up again, however, this time they are computed as *as-perceived*. Meaning, that for each category, the entirety of all buildings classified into the respective class by all users is registered. Then again the mean values for each feature are computed, representing the *as-perceived* or *as-expected* features for each category. This way, a comparison between actual properties of a building class and as they are expected by the users is possible (cf. section 3.2.2.4).

3.2.2 Results and Application

This section first presents overall results of the users’ classification (cf. section 3.2.2.1). Based on these results, we performed evaluations on the entirety of all users (cf. section 3.2.2.2) and different groups of users (cf. section 3.2.2.3) before deriving concrete knowledge on the users’ perception of building categories in section 3.2.2.4. This knowledge can be divided into metrics for the reference (cf. section 3.2.2.5) and the *as-perceived* classes (cf. section 3.2.2.6). Subsequently, in section 3.2.2.7 we investigate the influence of the

		Ground Truth						
		OFB	MFB	RT	BWS	OFF	IF	Sum Class
Prediction	OFB	1483	59	1	69	1	2	1615
	MFB	475	2462	137	460	62	8	3604
	RT	6	377	2042	95	103	1	2624
	BWS	23	222	87	1626	493	57	2508
	OFF	18	237	513	265	1983	64	3080
	IF	11	3	4	77	142	2172	2409
	Sum GT	2016	3360	2784	2592	2784	2304	15840

Table 3.3: Confusion matrix for building classification.

	Producer Accuracy [%]	User Accuracy [%]	Commission Error [%]	Omission Error [%]
OFB	73.6	91.8	8.2	26.4
MFB	73.3	68.3	31.7	26.7
RT	73.3	77.8	22.2	26.7
BWS	62.7	64.8	35.2	37.3
OFF	71.2	64.4	35.6	28.8
IF	94.3	90.2	9.8	5.7

Table 3.4: Classification metrics obtained from confusion matrix.

different representation types. Finally, a first application of the obtained knowledge, namely perception-based abstraction, is presented in section 3.2.2.8.

3.2.2.1 Classification Results of User Study

In total, 96 test persons have participated in the user study. The participants' age ranges from 18 to 73 with a mean of 24.8 years. The majority of the participants are students from Germany and abroad. In the following, we will first evaluate the classification results based on the entirety of all users. Afterwards, the results will be evaluated with respect to different groups of users.

3.2.2.2 Evaluation Based on the Entirety of All Users

The producer accuracy is given as the ratio of correctly classified buildings with regard to all ground truth buildings in this class. However, user accuracy is more interesting for this work – it is the fraction of correctly classified buildings with respect to all buildings classified to the current class. Commission errors correspond to buildings that were classified to a particular class, yet are actually belonging to another. Omission errors are buildings that actually belong to the ground truth class but were classified to a different category. The results can be seen in table 3.4.

Obviously, *one-family buildings* and *industrial facilities* could be identified best with both over 90% user accuracy. Users have most difficulties with the classes *office building*, *building with shops* and *multi-family building* which is indicated by user accuracies between 64.4% and 68.3%. Reasons for that will be further explained in section 3.2.2.4.

Besides the classification result, for each building the users should also rate their certainty for the particular decision. For 22 buildings the correct classification result was below 50%, with a mean correctness of 32.4% for these buildings. However, the mean certainty value for the same buildings is 3.78, which translates to a

certainty level of closely to *Certain*. This reflects the issue of users often not even being aware of currently misinterpreting data. Even more: The user might feel certain in his wrong classification. Therefore, it is necessary to use derived knowledge about the difference between perception/expectation and reality to optimize the building representation for the user's needs.

3.2.2.3 Evaluation Based on Different Groups of Users

In the following, we analyze whether different groups of users come to different classification results. The participants of the study have been quite homogeneous with respect to age (90 % between 18 and 30 years), graduation (over 90 % higher education entrance qualification, Bachelor or Master), and subject of study (over 95 % engineering studies). However, clearly separable user groups of meaningful size can be identified with respect to gender (71 % male, 29 % female), the users' origin (38.5 % German, 61.5 % foreign) as well as the users' previous experience with 3D virtual reality worlds (75 % experience, 25 % no experience). Thus, the user study is additionally evaluated with respect to the latter three properties. For this purpose, the same accuracy measures as in the previous section have been determined. This time, however, for the different user groups separately. Significance tests in form of Students t-tests are carried out to search for significant differences in the classification results between those user groups.

Evaluation based on gender of users: 71 % of the participants were male, 29 % female. To examine whether there are gender-specific differences in the way of how humans perceive building categories, the results of both groups have been evaluated separately and compared to each other. The analysis shows no significant differences between male and female users.

Evaluation based on origin of users: To investigate influences of the user's origin on the classification results, an evaluation based on the user groups *German* and *foreign* has been performed. 38.5 % of the users in the survey are from Germany, complementary 61.5 % of the users have another nationality, distributed all over the world. Since all building models presented in the survey are located in Germany, and architectural construction for equal building types might vary throughout the world, this distinction seems eligible. However, tests on features in each building category did not reveal any significant difference between foreign and German users.

Evaluation based on users' previous experience: Further, the factor of self-assessment with regards to previous experiences in 3D virtual reality worlds is examined. 75 % of the test persons stated that they have previous experience in this subject, whereas 25 % stated they don't. However, the results for this subject are somewhat ambiguous, since experience in the topic of 3D virtual reality worlds could be interpreted quite widespread. Tests unveiled no significant difference between users with previous experience and novices.

As no significant differences in the classification results of the aforementioned user groups can be identified, all subsequent evaluations and interpretations in the following section are based on the entirety of all participants.

3.2.2.4 Derivation of Knowledge on Building Perception

Based on the findings described in the previous section, we go one step further and try to derive coherences between the perceptibility of the building categories and several properties of the 3D representations. To find answers to the questions questions posed earlier (*Which representation type is for which building category the best?* and *Which geometric building properties and structures are relevant for the perceptibility of a particular building category?*), we proceed as follows: first, we extract geometric dependencies, i.e., dependencies between the perceptibility of a building's category and the building's geometric properties. The goal is to derive geometric building properties and structures which are relevant or essential for the perceptibility of a specific building category. Following this goal, we first analyze the geometric properties of the building

categories' representatives of our ground truth (cf. section 3.2.2.5). Afterwards, the same analysis is done for building categories *as-perceived* by the users (cf. section 3.2.2.6). In section 3.2.2.7, the perceptibility with respect to different representation types is analyzed.

3.2.2.5 Metrics of Building Categories Reference

The geometric building features introduced in section 3.2.1.2 are evaluated for each ground truth category (cf. table 3.5(right part)). Based on that, the building categories are tested whether they are significantly different in their geometric features. For that purpose, multiple significance tests are performed for each building feature's class mean. In the following, we list some significant characteristics for each building category within the ground truth:

- **One-Family Buildings** have a significantly smaller footprint than all other categories besides *building with shops*. The total building height, the number of floors and the number of windows are smaller than in all other classes.
- For **Multi-Family Buildings** the total number of floors is significantly higher than for *one-family buildings* and *industrial facilities*, yet lower than for *residential towers* and OFF. Accordingly, the total number of windows is higher than for *one-family buildings* but lower than for *residential towers* and *office buildings*. *multi-family buildings* only differ in few features from *building with shops*, hence the more important they are. The mean window surface is significantly smaller than for BWS. Related thereto, a different arrangement, size, and shape of windows on ground floor and a different ground floor itself as compared to the remaining floors is significantly more important for *building with shops* than for *multi-family buildings*.
- The most important feature of **Residential Towers** is the total number of floors, which is significantly higher than for all other building categories. Apart from *multi-family buildings*, to which no significant difference is detected, the total amount of balconies is higher than in all other categories.
- To distinguish **Buildings With Shops** from the rest, the most important features are different arrangement, size and shape of windows on ground floor as well as different ground floor itself in comparison to the remaining floors. These properties are significantly higher than in all other categories.
- Two features are salient for **Office Buildings**: The total amount of windows per facade, and the number of floors is significantly higher than for all other categories (except *residential towers*). Moreover, the mean entrance surface area is significantly higher than for OFB, MFB and RT. To distinguish *office buildings* from *buildings with shops*, a higher number of windows per facade as well as a higher amount of floors is characteristic. Accordingly, the ground floor and first floor resemble each other more in contrary to BWS.
- For **Industrial Facilities** the footprint is the predominant feature because it is significantly higher than in all other categories. The window-to-wall-surface ratio is lower than for *multi-family buildings*, *residential towers*, *buildings with shops* and *office buildings*.

Table 3.5: Geometric properties of the building categories as given in the ground truth (right part), and as classified by users in the study (left part).

As Classified						Ground Truth						
OFB	MFB	RT	BWS	OFF	IF	OFB	MFB	RT	BWS	OFF	IF	
115.30	238.41	573.01	697.51	868.26	10812.58	148.69	252.92	328.95	432.80	480.00	912.60	Footprint(m ²)
2.10	4.00	15.50	3.70	5.90	2.50	3.10	5.80	6.90	5.70	6.30	5.00	# Floors
3.30	2.94	2.73	3.84	4.09	24.63	8.65	5.43	3.55	6.78	6.61	11.10	Floor Height (m)
9.58	14.68	43.67	17.61	24.94	59.60	18.25	22.98	24.19	26.69	28.11	36.78	Total Height (m)
8.30	27.80	110.80	34.90	96.80	23.00	22.20	48.70	61.00	56.50	59.90	59.30	# Windows Per Facade
1.33	1.93	1.94	3.58	4.94	10.17	3.47	2.31	3.84	4.52	3.84	6.19	Ø Window Surface Area (m ²)
16.10	30.00	29.10	52.20	127.90	10.10	25.90	32.50	56.90	53.60	52.50	61.70	Window/Wall Surface Ratio (%)
1.70	1.40	1.20	1.60	1.40	5.30	1.60	2.20	1.40	2.30	2.30	2.80	# Entrances
4.13	2.99	3.53	24.90	7.93	13.33	4.99	7.35	8.39	9.86	8.97	13.48	Ø Entrance Surface Area (m ²)
0.20	2.20	9.50	1.10	0.10	0.00	0.90	2.60	3.00	2.00	2.10	1.50	# Balconies
1.04	2.06	4.24	10.08	3.42	0.00	1.80	3.82	4.36	3.30	3.34	3.88	Ø Balcony Surface Area (m ²)
0.67	0.18	0.36	0.91	0.53	0.53	0.53	0.49	0.46	0.54	0.49	0.53	Different Window Arrangement in GF
0.27	0.18	0.36	0.91	0.41	0.47	0.30	0.38	0.39	0.47	0.44	0.47	Different Window Sizes in GF
0.33	0.18	0.36	0.82	0.41	0.47	0.30	0.40	0.38	0.46	0.41	0.49	Different Window Shapes in GF
0.13	0.06	0.21	0.55	0.53	0.40	0.20	0.29	0.28	0.36	0.32	0.40	Different Ground Plan in GF
2.70	2.70	1.00	2.30	1.00	1.40	2.30	1.90	1.80	1.80	1.70	1.40	Roof Complexity

3.2.2.6 Metrics of Building Categories as Perceived/Expected by Users

As done before for the ground truth data, mean features are computed, this time based on the total amount of buildings all users classified into the respective class (cf. table 3.5(left part)). To compare the ground truth data with the results from all users, a significance test for the differences in all corresponding features is computed – this way discrepancies in the user’s perception or expectation and ground truth can be revealed.

The most important findings in this evaluation are:

- For **One-Family Buildings**, there is no difference in perception and ground truth.
- For **Multi-Family Buildings**, a different arrangement of windows on the ground floor as well as a different ground floor itself in comparison to the remaining floors of the buildings is expected. Additionally, in the users’ perception, *multi-family buildings* have a higher number of floors.
- To classify a building as **Residential Tower**, for users, the number of floors can be less and the total height lower in comparison to ground truth. However, a single floor height is expected to be higher than for the ground truth.
- **Buildings With Shops** are considered to have a higher number of floors than in reality.
- For **Office Buildings** users are expecting a higher number of balconies.
- **Industrial Facilities** are expected to have more windows per facade and a bigger number of floors, too.

3.2.2.7 Findings Based on Building Representation Type

By separating the evaluation into geometric and textural representation types, their impact onto the classification results can be measured. For 60 % of the models at least two different representation types for the same building are available. The mean correctness for untextured LOD3 models is at 69.2 %. Whereas a slightly higher correctness could be achieved for the textured meshes/LOD2 models from Google Earth with 75.4 %. However, the most accurate classification result with 79.3 % is based on the images from Google Street View. To determine whether the results actually differ from each other, again significance tests for the differences between geometric and textured representations results have been performed. The difference between untextured LOD3 models and textured meshes/LOD2 models from Google Earth is not significant but there is a significant difference between the geometric representation and images from Street View. One

reason for the superior correctness obtained for Street View representations could be the viewpoint of the models. As exemplarily shown in the last column of table 3.1, all images are captured looking slightly upwards and thus resembling the human perspective. These findings are one of the reasons that encouraged us to carry out the work in chapter 5.

For building categories that are easily separable from the rest like *one-family building* and *industrial facility*, a geometric representation is sufficient in the majority of cases. Particularly for buildings belonging to somewhat more ambiguous categories like *building with shops*, *office building*, and *multi-family building*, additional textural information improves the classification results.

3.2.2.8 Perception-Based Abstraction

The knowledge derived in section 3.2.2.5 and section 3.2.2.6 describes geometric 3D building properties and structures which are characteristic for a specific building category. Many applications that require users to move around in virtual 3D cities can benefit from this perceptual knowledge. It is even more critical that the abstracted building representations still contain those geometric properties and structures which are essential for perceiving the correct category of the buildings when the virtual 3D city consists of abstracted, geometrically simplified buildings. This scenario becomes more and more frequent when applications are visualized on small screens, such as smartphones and tablets. In the following, we will show how such perceptual knowledge can be embedded in a 3D abstraction process. Effects on the perceptibility of the buildings' categories will be demonstrated based on representative examples.

In a preprocessing step, information about perceptual relevance is attached to the respective 3D structures to provide semantically enriched building representations as input for the abstraction process. Based on [Nan et al., 2011], we create different abstractions of buildings based on human perception. [Nan et al., 2011] applied different Gestalt rules to drawings of facades which helped them to group drawing elements and to represent them by other elements. We extended this idea to the three-dimensional blocks formed by the facade elements and use it for abstracting given buildings. During this process, we used the Gestalt laws of proximity, regularity, and similarity to group blocks together and represented the results by larger blocks. The preservation of geometric properties and 3D structures, which are essential for perceiving the correct building category, is ensured by translating them into geometric constraints as restrictions for the abstraction process.

Figure 3.9 (a), (d), (g) depict the original building models, respectively followed by two different results of the abstraction process. For the first abstraction, parameters have been chosen based on features that are important for the user to correctly classify a building. The second abstraction is completely free, meaning that no restrictions were made during the abstraction. In figure 3.9 (a) a *residential tower* is depicted. Windows have been merged over two floors for both abstractions here. As a consequence, the total building height appears to be smaller and the number of floors decreases with increasing single floor height at the same time. This exactly corresponds to the findings made for the users' expectation of the category *residential tower* (cf table 3.5). The important feature *balcony* is maintained in the first abstraction, the second abstraction, however, drops it. This way, model (b) retains the appearance of a residential building, whereas model (c) is more neutral and, thus, could also be interpreted as an *office building*.

Figure 3.9 (d) depicts a model belonging to the class *building with shops*. The first abstraction incorporates the properties learned to be important for *building with shops* as mentioned in section 3.2.2.5. The ratio of the window size between the ground floor, the first floor and the remaining floors is preserved, as well as the arrangement of the windows. The second model is a free abstraction. As a result of the abstraction process, both models (e) and (f) have merged dormers. However, the window shapes and distribution have changed. For example, (e) retains smaller windows in the upper floor, while (f) has a merged window front. This merged window front destroys the building's original property of having significantly bigger windows

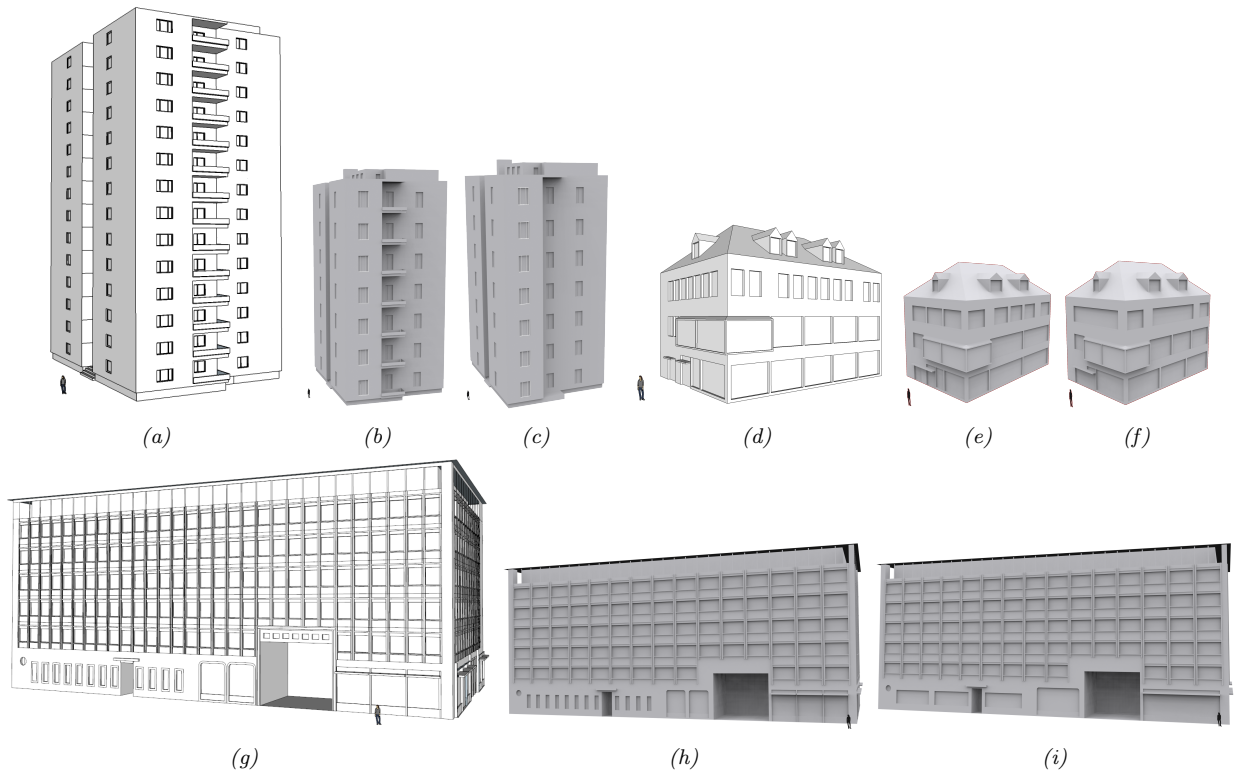


Figure 3.9: Examples for perception-guided abstraction. White buildings ((a), (d), (g)) depict the original input models to the algorithm. Gray models in the middle ((b), (e), (h)) show abstractions based on features that are important for the user to classify into the respective correct category. Gray models on the right ((c), (f), (i)): ‘Free’ abstraction without restrictions.

in the lower floors than in the remaining floors, which was detected to be an important feature of BWS, though.

Figure 3.9 (g) shows the example of an *office building*. The abstracted model (h) keeps the characteristic structure of the ground floor but merges windows in the upper floors, thus still closely resembling the original. Model (i), though, merges windows and entrances in the ground floor. As a consequence, the model might rather be perceived as *building with shops* than *office building*.

3.3 Grammar-Based Enhancement and Abstraction

Procedural modeling is an efficient technique to generate topologically correct 3D building models in large quantities. A prominent procedural software tool is **CityEngine** by ESRI. In this section we present an approach that can identify an unknown building model and generate an semantically enhanced version of it by leveraging the knowledge obtained in the previous section. This can be achieved by designing and applying category-specific rule sets. Figure 3.10 gives an overview of the proposed workflow which will be further explained in the following subsections.

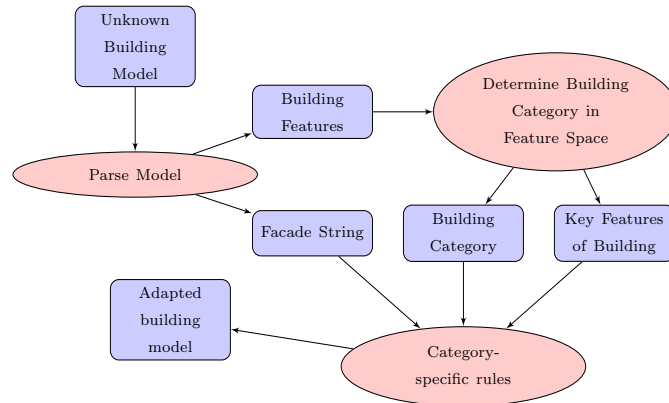


Figure 3.10: Workflow of the category-specific building enhancement.

3.3.1 Building Feature Space

Within our first user studies we designed features that can be used to describe each individual building (cf. section 3.2.1.2). Those properties can either be directly derived from the model geometry, such as building height and footprint, or they are calculated by considering building-related semantic aspects, such as number of floors, entrances, windows and balconies.

However, since CityGML is often used in the geographic information system (GIS) community, we utilize this data format and derive the features now in an automated manner. Since CityGML incorporates semantic information, we can access the related geometry of building primitives (such as windows and doors). This way, we can parse the facade for its geometry and semantics and encode this information into a XML file containing the essential information to describe the building. Figure 3.11 shows an excerpt of such a generated XML file. Note, that the parsed building in this example actually has 22 floors, for better visibility we only show a truncated version.

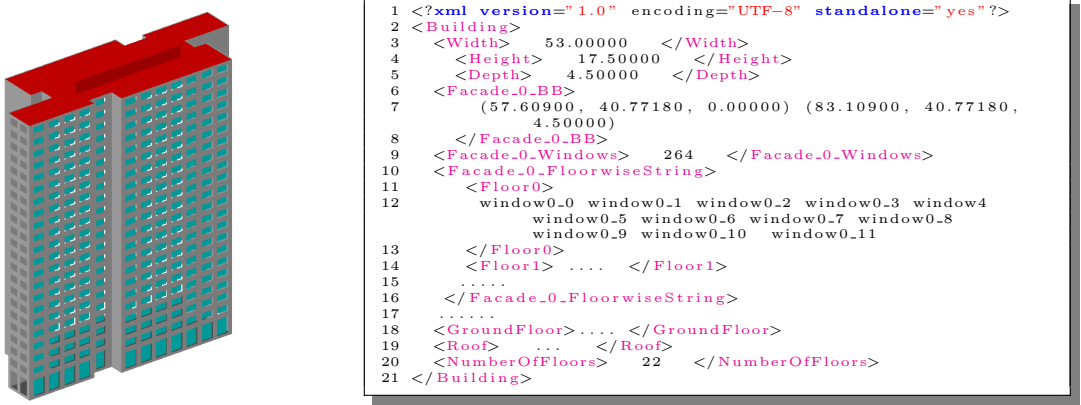


Figure 3.11: Input CityGML Model (left) and an excerpt of the resulting parsed XML (right).

3.3.1.1 Feature Space

Taking into account the mentioned features in section 3.2.1, we define an embedding for every building model. This means, we can transform our building into a feature representation and describe similarities of buildings by means of distance in a feature space. We performed a t-distributed stochastic neighbor embedding (t-SNE) to give a better impression of the discrepancies between ground truth and as-perceived building categories in our previous tests. t-SNE is a technique for dimensionality reduction, especially suitable for data of high dimensions [Maaten and Hinton, 2008]. In a pre-processing step, PCA is used to initially reduce the dimensionality of input data. Subsequently, t-SNE maps the input to either a 2D or 3D visualization. In our case, the embedding is 16-dimensional, since we have a total number of 16 features describing a building. In total, we map 12 different instances, describing the mean building of each class from the ground truth and *as-perceived*, respectively. Figure 3.12 shows the t-SNE mapping, every coloured point denotes a building category instance, blue lines are linking ground truth and the mean instances as classified by users. From this figure, some of our findings become also visually clear. *One-family buildings* and *industrial facilities* could be classified quite well, accordingly they have small distances in the t-SNE mapping. However, users had problems to distinguish between *office buildings* and *buildings with shops*, correspondingly the distances are larger and entanglements occur. To determine the building category of a new building now, we first have to parse it as described above. Once the essential properties are extracted, we can then perform a classification that can be considered a supervised learning approach. Our trained classes are the mean ground truth building instances, depicted as X_{gt}^i , with i corresponding to each of the six building categories. A 1-Nearest Neighbor (1-NN) classifier can then be used to classify each new unknown building sample. To avoid the curse of dimensionality we also implemented a dimensionality reduction using PCA and then searching for the nearest neighbor category. However, in our examples there was no difference between 1-NN on original features and using PCA as pre-processing.

If buildings are quite different from what we identified as mean building of each category, that leads to a large distance in feature space. In consequence, a building actually belonging to category A gets classified as belonging to category B , due to a smaller feature space distance. This might be problematic for the subsequent building category-specific adaption, since wrong rules might be applied. However, since CityGML is widely used by surveying offices, the building category is often contained in the XML structure. We take advantage of that fact and whenever the building usage is found in the parsing process we pull the information from there and do not need to perform any feature space processing.

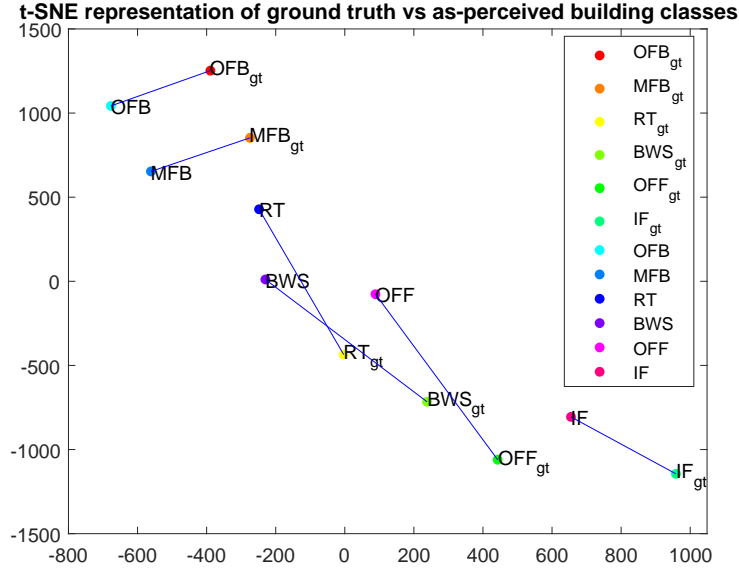


Figure 3.12: t-SNE representation of the mean buildings for each category. Blue lines denote the link between corresponding ground truth and *as-perceived* instances. Axes x and y denote the feature space and thus have no units.

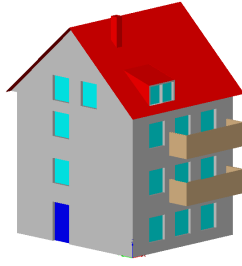
3.3.2 Perception-Based Rules

This section addresses the process of enhancing buildings in a way, that they become visually more distinctive for humans in terms of their type of use. In section 3.3.2.1, the properties and purpose of building category-specific rule sets are discussed, and section 3.3.2.2 shows how those rules can be used to either enhance existing buildings or use the rule sets to create completely new building category-specific instances. The aforementioned tasks can be accomplished by means of procedural modeling with the underlying concept of grammars, which was previously discussed in section 3.1.4.1.

3.3.2.1 Building Category-Specific Rule Sets

As discussed in 3.2.1, every building has a set of properties that qualify it belonging to a certain use type. To generate virtual representations of buildings, which are better to understand for humans, we need to refine or abstract, in general - adapt, those buildings in a certain manner. Thus, specific rule sets were designed for each building category. These rule sets incorporate geometric and semantic constraints we extracted from our previous user studies by relating the aforementioned features of ground truth buildings with the as-perceived classifications. Some of those rules are:

- *One-Family Building*: At least one visible entrance.
- *Multi-Family Building*: Higher number of floors than OFB; keep balconies if they occur in original.
- *Residential Tower*: If not existent, add balconies.
- *Building With Shops*: Ground floor significantly different from remaining floors (window sizes, shape, arrangement).
- *Office Building*: Very high window-to-wall-surface ratio.

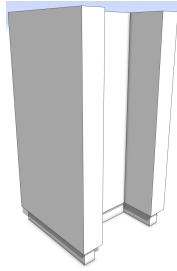


(a) Multi-Family Building Model in CityGML.

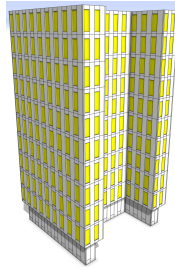


(b) Multi-Family Building Model generated from CGA Shape Grammar Rules.

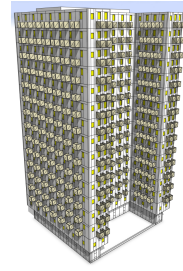
Figure 3.13: Transfer of a CityGML Multi-Family House (a) to CGA Rules generated version (b).



(a) Coarse model of a building tower.



(b) Enhanced building with generic Office Building rules.



(c) Enhanced building with generic Residential Tower rules.

Figure 3.14: Application of building category specific rules.

- *Industrial Facility*: Very high floor height.

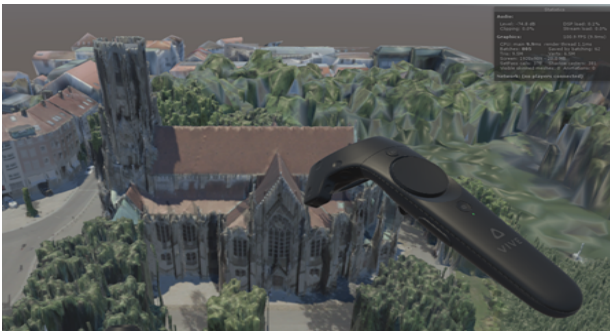
Figure 3.13 shows an example of a transferred CityGML representation of a *Multi-Family-Building* (cf. figure 3.13a) to the CGA Rule based representation (cf. figure 3.13b). Balconies are maintained, whereas positions of windows and doors on the front-facing facade are loosely coupled.

3.3.2.2 Generation of new instances

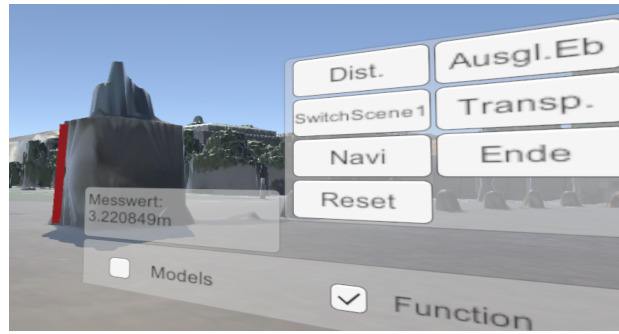
We use the described rule sets from the previous section to either generate use type specific buildings from scratch or to adapt existing ones. For the latter one, it is important to maintain the key characteristics of the building (see section 3.2.1). Therefore, we feed the essential elements of the parsed XML into the modeling process. As a first proof of concept, we used coarse building models as shown in figure 3.14a and enhanced them with category specific rule sets. We then ported the results into a VR environment and initially only displayed one of the representations depicted in figure 3.14b and figure 3.14c. We asked the subjects to classify that building into one of the six pre-defined categories introduced in section 3.2.1. First results have shown, that the *as-perceived* categories coincide with the building categories that were intended to be imparted. However, more extensive actual user studies based on this initial prototypic test have to be conducted in the future in order to support those findings.

3.4 City Models in Virtual Reality (VR)

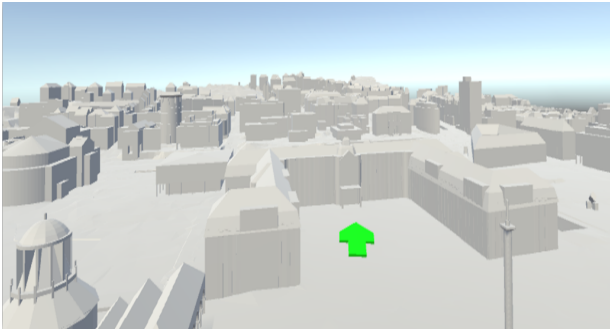
Tremendous developments in hard- and software have boosted the sector of virtual reality in the last years. Decreasing costs and increasing computational capabilities have led to the fact, that meanwhile even smart-phones serve as VR devices. Apart from that, stationary computer graphic card-powered devices like Oculus



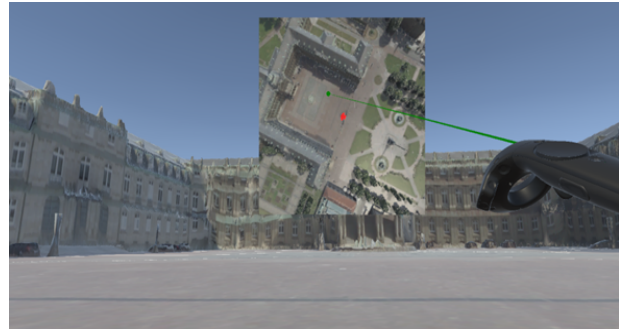
(a) Johannes church at Stuttgart, Feuersee with HTC Vive controller visible in application.



(b) Canvas with function overview and showcase of the measurement tool. Red line indicates measured distance, output value is displayed in *Messwert* field.



(c) Untextured CityGML LOD2 model displayed in flight mode. Green arrow indicates flight direction.



(d) Minimap inlay during pedestrian mode, located at Schlossplatz, Stuttgart.

Figure 3.15: VR environment for photogrammetric data.

Rift or HTC Vive and even completely autonomous ones like the Mixed Reality focused Microsoft HoloLens have hit the consumer market and enable deeply immersive experiences. With the availability of commercial VR systems, new opportunities have opened up in terms of data presentation. We see this as a chance for our community to get an actual feel for triangle meshes and BRep models derived from aerial/unmanned airborne vehicle (UAV), terrestrial imagery or LiDAR. This way, data results can be made more attractive to a broader crowd. Realities.io³, a startup with its roots in Tübingen, Germany, is one such example. The initial use case was to use photogrammetry to capture archaeological excavation scenes and display them in VR. Their platform has since evolved to a Silicon Valley VR company with broad media coverage. In the meantime, they deliver photorealistic VR models for a variety of indoor and outdoor scenes, captured by means of laser scanning and photogrammetric (UAV) imagery.

Our focus is not on models that are as beautiful and photorealistic as possible though, but on an efficient and easy way to present research results. In the future, VR will not only serve as an environment to inspect, but also edit data - to, for example, smooth out imperfections in triangulated meshes - and save the edits back to the actual input data. However, this is not the only area where we see potential benefits of VR. Within the context of this work, we conceptualized a framework in Unity that gives users the possibility to move around in an urban environment and switch between different building representation types [Han, 2016]. Our approach is designed for the HTC Vive system and offers the user several in-game functions: Locomotion via teleport and 2 different flight modes (*Superman* and *Bird View*), Mini-Map for orientation and quick navigation, switching between different building representation types, distance measurement tool, calculation of best-fitting plane in selected area and prototypic grabbing, rotating and scaling of building primitives such as windows. Figure 3.15 (b) exhibits a use case for the measurement tool, Figure 3.15(d)

³<http://realities.io/>

depicts navigation and orientation using the minimap. Due to the room-scale capabilities of the HTC Vive system the user can walk around in the real world and cover equal distance units in VR. However, when trying to cover larger distances in the virtual world the most efficient and human-friendly way of locomotion is pointing at a location and directly teleporting there. Additionally, we implemented two flying modes for large scenes. In *Superman* mode, one controller serves as 3D joystick to navigate through air within a fixed range of velocity to prevent from motion sickness. *Bird View* mode imitates the movement of birds, obviously. Here, the user mimics wings with his or her arms. By flapping the arms with the game controllers in hand it is possible to gain speed, tilting the arms just like a plane leads to flying a curve. We implemented this framework as a testbed to visualize content produced by means of photogrammetry and we want to use it to study the influence of different representation types of urban data on users. Thus, it is possible to toggle between different 3D data types for the same urban area. The current setup contains three different representations types: untextured LOD2 (cf. Figure 3.15(c)), textured LOD2 models for selected buildings (from Sketchup 3D Warehouse) and a textured 2.5D triangle mesh (cf. Figure 3.15(a)). The first two are not too difficult to manage in terms of central processing unit (CPU) and graphics processing unit (GPU) load. However, the textured mesh contains massive amounts of triangles in its highest level of detail. Therefore, we have to make use of a LOD concept, where only those parts very close to the camera are rendered in the highest LOD. This way, a sufficiently high frame rate can be maintained at all times to ensure a smooth VR experience. Extensive tests on performance with regards to complexity of objects, occlusion culling, amount of meshes and textures, different types of batching, and LOD group optimization have been performed and can be found in [Han, 2016]. This presented VR framework cannot only be used for the visualization of different data representations but also as a test bed to let users switch between different versions of the same building to examine whether a perception-based adapted building model enhances the level of insight to better conceive a building's use type. Some first proof of concept was discussed in section 3.3.2.2.

3.5 Summary for the Understanding and Enhancement of Building Models

In summary, section 3.1 presents a semi-automatic approach for building reconstruction in urban areas. We provide a framework that allows standardized data formats as input and output. The framework relies on an adaption of the facade reconstruction and modeling approach of [Becker, 2011], for more details the reader is referred to this work. An approach for radiometric segmentation to bypass the classification into *facade points* and *non-facade points* is proposed. Furthermore, it is possible to export a directly geo-referenced, synthetically enriched building model in CityGML format. By selecting a facade of interest, users are able to reconstruct the primitives of this facade. Subsequently, a facade grammar can be inferred from the primitives. This grammar is applicable to the remainder of the building, thus synthetically modeling the whole building in the style of the source facade. Currently, only one facade per building can be selected to infer a grammar from and apply it to the building. An extension would be, to allow grammar extraction for several facades and have a probabilistic approach choose rules for the remaining facades or even surrounding buildings.

With the aim of deriving knowledge on the human's ability to understand semantics from 3D building structures, we presented a user study on the user's comprehension of building categories based on different 3D building representations in section 3.2. Within the study, the users were asked to classify consecutively presented single building representations into different building categories. During the whole classification process, the users additionally had to rate their level of certainty. Analyses of the user study reveal clear coherences and dependencies between the correctness of classifications and the model representation type. In general, it is conducive to have textural information for buildings: The overall classification accuracies for textured meshes/LOD2 models from Google Earth and images from Google Street View are 75.4% and 79.3% and, thus, significantly higher than the classification accuracy of untextured LOD3 models, which

lies at 69.2%. Particularly for buildings that are belonging to somewhat more ambiguous categories like *building with shops*, *office building*, and *multi-family building* additional textural information improves the classification results. However, for building categories that are easily separable from the rest like *one-family building* and *industrial facility*, a geometric representation in form of a LOD3 model is sufficient in the majority of cases.

The classification accuracy of LOD 3 models mainly depends on whether the building models show properties that have been detected as perceptually relevant for the respective building category. Examples for such perceptually relevant geometries and structures are the occurrence of balconies for *residential tower*, the different appearance of ground floor and remaining floors for *building with shops*, or the high windows-to-wall-surface ratio for *office building*. As these properties are not inherent in all representatives of the categories mentioned, users sometimes experience difficulties to distinguish between *building with shops*, *multi-family building* and *office building*. Moreover, the majority of the users is not even aware of their misinterpretations which makes perception-adapted building representations an even more important issue. Therefore, it is crucial to guide the representation based on features that are significantly characteristic for the respective building category. The knowledge gathered in the investigation of ground truth features and the significant features as-perceived or expected by users can then be used to generate virtual 3D models that support and improve the correct perception of building categories.

As a first application, we demonstrated how such knowledge about the human's perception of building-related semantic information can be used for the perceptually adapted abstraction of 3D building models. Characteristic properties of building structures that turned out to be essential for the perceptibility of a certain building category are maintained during the abstraction process whereas structures that are unimportant or even obstructive are simplified to a much higher extent or even totally neglected. Doing so, the perceptibility of the building category can be preserved even in abstracted building representations.

Having considered buildings so far on an individual basis only, investigations on how the human perception of a building's category is influenced by the building's environment should be part of future work. To retrieve information about the impact of different building representations on the users' way of navigating through virtual 3D environments could be one scope for further experiments. The perceptual knowledge gained therefrom can be embedded into a formal grammar. Such a grammar-based system could additionally allow to modify, emphasize, add or remove perceptually relevant structures in a targeted manner in order to automatically generate models that can be classified more easily into the respective correct building category.

One step in this direction is outlined in section 3.3. We presented an approach to adapt 3D building models in a way, that they become visually more comprehensible for humans. The CityGML format is used to parse the building for essential features. If the building use type is not contained in the CityGML file, we perform a feature space transformation and determine the most likely category there. Based on findings from section 3.2, we designed building category-specific rules that incorporate perceptual constraints. Feeding essential building properties from the parsing process into the category-specific rule sets leads to semantically more comprehensive representations of input buildings. However, building models that are geometrically quite different from their class mean will lead to misclassifications. More sophisticated machine learning approaches and more input building samples could be used to tackle this issue and will be further addressed in chapter 5. Yet, if the building category is embedded in the CityGML file, we can directly access the information from there. For future work, the presented concept could be extended to not only modify single buildings, but consider the architectural neighborhood of a building and model interrelations between buildings. This way, a holistic semantically consistent 3D Virtual City could be accomplished. To verify the pursued approach, an actual user study should be set up. This can either be an extended version of the proof of concept mentioned above or by using a web platform for 3D content and using crowd-sourcing for the evaluation.

In section 3.4, we proposed a framework for the visualization and immersion of urban virtual city data in

VR. We see VR as a real opportunity for the community to not only make textured meshes and BRep models generated by means of photogrammetry and computer vision more accessible to the public but also more productive for the community itself. Our presented VR application has been widely used for public relations events of the institute and the whole study course and was enthusiastically received by users (children as well as adults). Promising use cases in the future are immersive measurement tools, on-the-fly adaption of (building) models and mesh cleaning operations.

Chapter 4

Semantic Urban Mesh Segmentation

The previous chapter described approaches to reconstruct and refine the geometry of building models. In the past, photogrammetric generation of point clouds was mainly performed by imagery from nadir cameras or delivered by ALS. As a consequence, point clouds often lacked information on facades of buildings in urban scenes due to complex infrastructure and occlusions. This was the main motivation for grammar-based model completion described in section 3.1.4.2. In the meantime, the state of the art in hardware and software has evolved. A combination of nadir and oblique camera systems, along with larger sensor sizes, is commonly used for urban scenarios nowadays. Such new hardware, coupled with the development of sophisticated MVS algorithms led to quasi-complete urban point clouds and meshes. Consequently, there is no longer a great need for grammar-based approaches to synthesize unobserved building sides and procedural modeling is today mainly used for the automatic generation of synthetic cities for cinematic visualizations or game environments. We therefore shift our focus in this chapter away from the geometric aspects discussed in section 3.1 and concentrate on the previously rather neglected semantics.

In section 3.3, we described an approach to geometrically modify buildings to make them more comprehensible for humans. These enhancements are only possible, however, if buildings can be detected in a 3D virtual world beforehand. It is therefore essential to semantically fully comprehend the data basis, that is, the virtual city model. This chapter describes an approach for the semantic enrichment of a textured triangle mesh and, thus, the extraction of geometries of different classes such as building facades, roofs, vegetation, or impervious surfaces.

In photogrammetry and computer vision, point clouds are usually used as data basis for further processing, such as primitive extraction (see section 3.1) or semantic segmentation. However, point clouds are actually not a final product of the photogrammetry pipeline. Dense clouds usually have a huge information redundancy and are therefore often downsampled to maintain data handling capabilities and decrease data storage impact. A triangle mesh on the other hand is a more compact representation that explicitly provides topology (see also chapter 2). Color can be either provided vertex-wise or per triangle. For the latter case, usually texture maps are provided. Mapping applications from large technology vendors such as Google Maps or Apple Maps similarly provide very detailed meshes in their 3D views in the meantime. This even extends to VR applications with real-world meshes of astonishing quality ¹. Naturally, the textured triangle mesh seems to be the 3D geoproduct of the future. However, to the best of our knowledge, only very little effort has been put into the semantic processing of meshes. Therefore, we address the task of semantic urban mesh

¹<https://vr.google.com/earth/>

segmentation. In order to perform further tasks of the pipeline for building interpretation and enhancement presented within this thesis, it is necessary to actually identify buildings in the input data.

As stated earlier, virtual city models are an integral part of our daily lives and applications like navigation, urban planning or computer games base on 2D and 3D geodata. Map applications - no matter if they are crowd-sourced like Open Street Map or governmental - contain geometric information organized in several layers. Each layer corresponds to a specific object category like for example *building*, *road*, *waterbody*. For this reason, map applications contain semantic information to some extent in an implicit manner. Additional and explicit semantic information on the other hand is often still lacking.

Colored triangle meshes have become the standard representation of virtual city models for 2.5D and 3D geodata [Boussaha et al., 2018]. In contrast to point clouds, meshes provide high-resolution textural information and explicit adjacency information. Meshes are the de-facto standard/deliverable in photogrammetric product pipelines since they require less storage than point clouds. Chapter 5 is concerned with image-based classification of building facades into different categories. This serves as a tool to semantically enrich map data. In order to classify individual buildings in a mesh in more detail, however, it is necessary to locate potential buildings first.

DL methods are state of the art for semantic segmentation in image space [Zhao et al., 2017, Chen et al., 2018a, Long et al., 2015, Noh et al., 2015, Liu et al., 2015, Yu and Koltun, 2015]. Point-based classifiers like PointNet++ [Qi et al., 2017] achieve good results on 3D point clouds. However, to the best of our knowledge, little work focuses on a DL framework that directly operates on meshes so far. In particular, the field of urban data represented as mesh has hardly been explored. Therefore, we address the task of semantic mesh segmentation - attaching semantic classes such as *building mass/facade*, *roof*, *impervious surface*, *green space*, *mid and high vegetation*, *vehicle*, *chimney/antenna* and *clutter* to each face. For this purpose we train a 1D multi-branch CNN. Ordinarily, CNNs are an end-to-end learning approach operating on regularly structured input data. Meshes, however, are not necessarily regularly structured. In order to make CNNs accessible to meshes, we initially calculate a multi-scale feature vector for each face. Consequently, we artificially achieve regularly structured input data: each face is represented by a normalized feature vector. By these means we end up with a hybrid model that combines explicit feature calculation and convolutional feature learning. Each branch of the 1D CNN is fed with the feature vector of corresponding scale (cf. section 4.3).

Labeled ground truth data are essential for supervised learning algorithms. Therefore, we manually attach a semantic label to each face of an urban mesh. Nevertheless, since generating training data is time consuming and expensive, especially for semantic segmentation tasks, we propose to use synthetically generated urban meshes as support for little real world data. The big advantage of synthetically generated urban meshes is that the label is already explicitly attached to each face.

Data preparation including labeling and feature calculation is described in section 4.1. In section 4.5, we evaluate our segmentation algorithm on both synthetic and real-world data and compare results to a semantic segmentation achieved by a default RF. Moreover, we compare the two classifiers' results after being smoothed by a MRF. In summary, the contributions of this chapter are: (1) Provision of ground truth data by manually labeling a 2.5D mesh of a real-world urban area (cf. section 4.1). (2) Calculation of a multi-scale feature vector for each face of the large-scale urban mesh (cf. section 4.1.2). Thereby, the subsequent semantic segmentation can be achieved by state-of-the-art classifiers (CNNs). Furthermore, the approach is independent of the geometric structure of the mesh: 2.5D or 3D meshes can be processed. (3) In section 4.2, a brief history and introduction to CNNs is given, since we make use of DL techniques in this chapter, as well as the following one. (4) We perform semantic mesh segmentation by training a tailored multi-branch 1D CNN (cf. section 4.3). We compare results to a RF baseline. (5) Results of the semantic segmentation are further smoothed by using a MRF (cf. section 4.4 and section 4.5.2). (6) An extensive ablation study is performed in order to determine most influential features (cf. section 4.5.1). (7) The use of synthetic data is investigated, as we assume additional data will support the performance of training and inference.

1.	<i>building mass/facade</i>	9.28 %
2.	<i>roof</i>	6.34 %
3.	<i>impervious surface</i>	5.67 %
4.	<i>green space</i>	5.97 %
5.	<i>mid/high vegetation</i>	63.38 %
6.	<i>vehicle</i>	0.83 %
7.	<i>chimney/antenna</i>	0.31 %
8.	<i>clutter</i>	8.22 %

Table 4.1: Label distribution of the ground truth labeled mesh.

4.1 Data Preparation

Our real world data set is obtained by fusing ALS data and aerial oblique imagery. Both are captured simultaneously for the purpose of monitoring a lock in Hessigheim [Cramer et al., 2018]. The capturing UAV is a RiCopter multi-copter platform equipped with a Riegl VUX-1LR LiDAR and two Sony Alpha 6000 oblique cameras. The LiDAR point cloud is used for creating a 2.5D mesh (also known as DSM mesh). The 2.5D textured triangle mesh generation is performed with software SURE from nFrames [Rothermel et al., 2012]. Although we use a 2.5D mesh, the applicability of the pipeline to a 3D mesh is exactly the same. A ground sampling distance (GSD) of 5 cm is used for meshing. The mesh is textured with images captured by the two Sony Alpha 6000 cameras. SURE provides a LOD mesh model. We use the second most detailed level in order to retain a good trade-off between computation effort and preserving details in the mesh. This mesh consists of ~ 3 million faces. In a first step, we split the available mesh into 24 tiles with dimensions of approximately 100×100 m each (cf. figure 4.1). In a second step, we fuse some of the tiles for validation and testing. In total, four tiles with varying spatial extents are held out of this training set - one serves as validation and three as test tiles. We chose the splits in such a way that the label distribution is similar in each data split. The distribution of our classes is as shown in table 4.1. Figure 4.1 outlines the splitting of our mesh into train set, validation set and test set.

Labeling is done manually. In more detail, students are given explicit instructions on how to label the mesh and distinguish between different semantic classes. For the labeling process itself, the students use Autodesk 3ds Max 2019. Each semantic class is defined by a material, that is, a specific RGB triple. The labeling process is defined as assigning a semantic material to each face of the textured input mesh. We are aware of the fact that manual labeling is an error-prone endeavor and that operators might miss or misclassify some faces during labeling due to occlusions and finely triangulated areas. Missed faces receive the label -1 (*Invalid*) and are filtered before classifier training. Although the manually attached labels are counter-checked by a different group of students, errors cannot be avoided as depicted in figure 4.6. This label noise might complicate the training procedure and certainly affects performance evaluation. In total the manual labeling needed approximately 300 person-hours.

4.1.1 Semantic Classes

Our considered classes are inspired by the ISPRS 3D semantic labeling contest [Niemeyer et al., 2014]: *building mass/facade*, *roof*, *impervious surface*, *green space*, *mid and high vegetation*, *vehicle*, *chimney/antenna* and *clutter*. *Building mass* and *roof* are mutually exclusive, since roof extraction is a quite common task. *Impervious surface* includes rocky areas, streets, sidewalks, parking lots and other man-made surfaces. In accordance with the *closed world assumption*, class *clutter* gathers all faces that do not match the other class labels. Figure 4.1 shows an extract of the real-world data set.

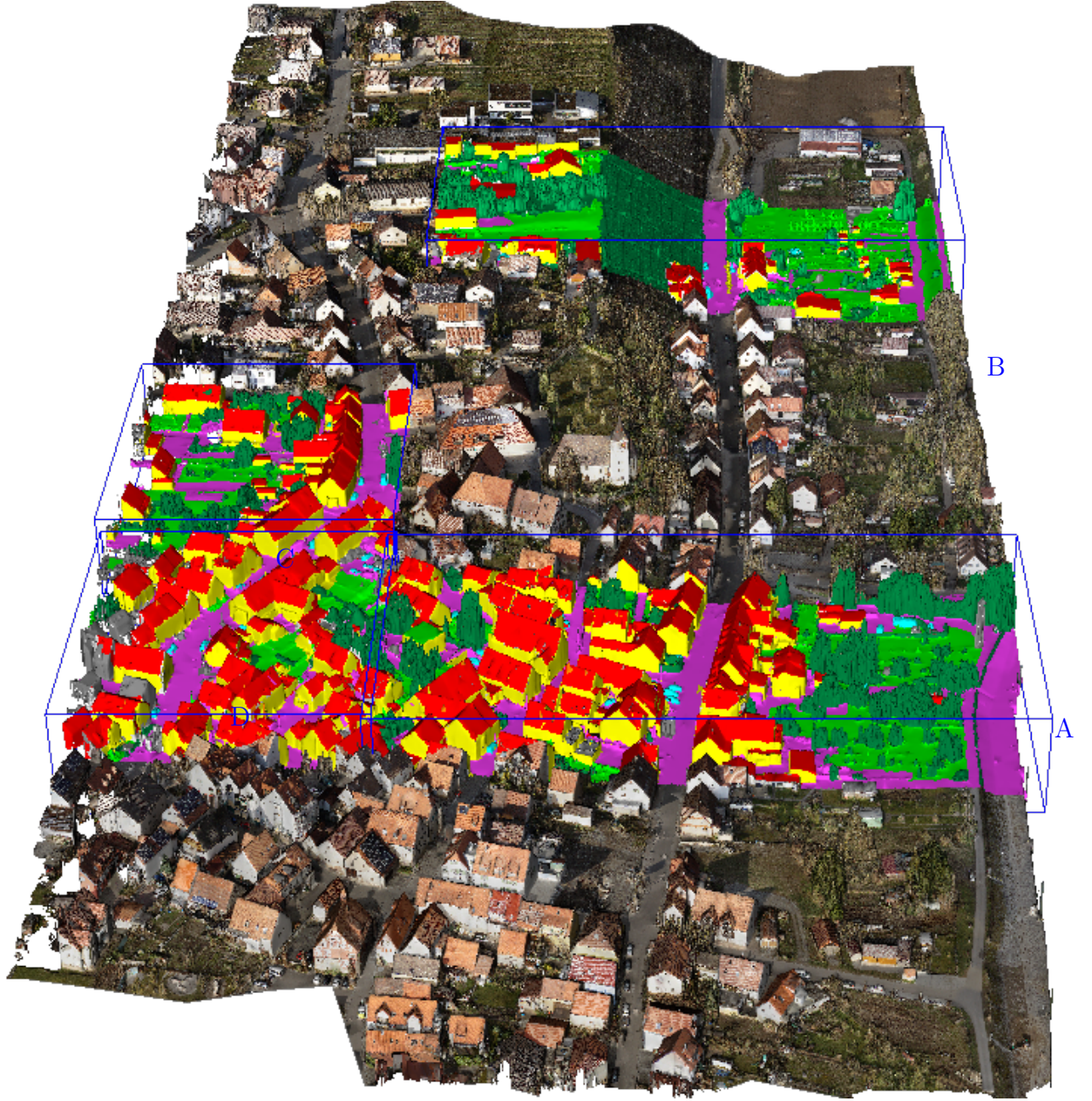


Figure 4.1: Subset of the urban mesh in Hessigheim, close to Stuttgart. The blue bounding boxes depict tiles that are used for validation and testing (mutually exclusive), represented semantically labeled here. The following class color code and enumeration is used throughout the paper: 1. *building mass/facade* (yellow), 2. *roof* (red), 3. *impervious surface* (magenta), 4. *green space* (light green), 5. *mid and high vegetation* (dark green), 6. *vehicle* (cyan), 7. *chimney/antenna* (orange) and 8. *clutter* (gray). Remaining tiles are used for training. The 2.5D triangle mesh was obtained by fusing ALS geometry with textures from UAV oblique imagery.

	per Vertex	per Face	per SPNH
geometric	Laplacian	Area	Covariance Features
	Mean Curvature	Normal	Verticality
	Gauß Curvature	Voronoi	Horizontality
	Valance	Verticality	Inclination
	Dihedral Angle	Horizontality	Face Density
		Inclination	Max Vertical Difference
radiometric		Vertical Difference	σ_{nDSM}
		n_{DSM}	
	-	RGB/HSV Hist	RGB/HSV Hist
	-	Level 1	Level [0,1,2]
	-	Median RGB/HSV	Median RGB/HSV

Table 4.2: Calculated features for each face. SPNH-based features are computed for SPNHs of 0.5 m, 1.0 m, 1.5 m. Histogram levels 0, 1, and 2 use 9, 20 and 64 bins respectively to map color values originally discretized in 256 steps.

4.1.2 Feature Vector Composition

Prior to training the CNN, for each face of the mesh a multi-scale feature vector is calculated. To do so, we implemented a pipeline that parses wavefront OBJ files with their texture maps. There are two modes for parsing: *texture* and *label*. The former is responsible for parsing the actual textured mesh and generates features for each face. Features can be distinguished into geometric and radiometric features (cf. table 4.2). To calculate either of them, a local neighborhood for each face has to be established. We do this by computing the center of gravity (COG) for every triangle. Thus, we essentially obtain a point cloud equal to the number of triangles in the mesh. This point cloud serves as input for a k-d tree. For each face, multiple SPNHs are queried in the k-d tree - in our experiments we use SPNHs of radii 0.5 m, 1.0 m and 1.5 m. All points within the respective neighborhood are selected. If the number of selected candidates is below a threshold (e.g. five points), the selection is extended to the required minimum number of points, regardless of their distance to the query point, that is, the COG of the considered face. Table 4.2 lists the features that are calculated for every face.

Vertex features are calculated by means of [Zhou, 2018]. We will briefly introduce some of the features and would like to refer the reader to the reference for further details. Since we work with triangle meshes, vertex features account for 3 or 9 components in the feature vector each, depending on whether the feature is scalar or vectorial, respectively. We obtain the *Laplacian* for each vertex i within the current face f . *Mean Curvature* H is defined as the mean of the two principal curvatures κ_1 and κ_2 being the maximum and minimum of the normal curvature, respectively:

$$H = \frac{1}{2}(\kappa_1 + \kappa_2) \quad (4.1)$$

Accordingly, *Gauß Curvature* K is defined as:

$$K = \kappa_1 \kappa_2 \quad (4.2)$$

Valance defines the number of edges incident to the current vertex, hence its degree of connectivity. *Dihedral Angle* of vertex i represents the maximum dihedral angle of vertex i . A dihedral angle of vertex i is the angle between two faces that are adjacent to an edge that incidents at vertex i .

Face features are straight-forward. *Area* is the area spanned by the triangle. *Normal* depicts the normal vector \mathbf{n} computed from the cross-product of the vertices. The Voronoi area of each corner of the the face is encoded in *Voronoi*. *Verticality*, *Horizontality* and *Inclination* are computed for the face itself, as well as for

each SPNH. For the latter case, we extract mean, median and standard deviation. *Inclination* ι is defined as:

$$\iota = \arccos \frac{\mathbf{n} \cdot \mathbf{v}}{\|\mathbf{n}\| \cdot \|\mathbf{v}\|} \quad (4.3)$$

with vertical vector $\mathbf{v} = [0 \ 0 \ 1]^T$. *Horizontal* is simply $|n_z|$, i.e. the vertical component of the face normal \mathbf{n} . Correspondingly, *Verticality* is defined as $1 - n_z$. The difference in Z between lowest and highest vertex within the triangle is described by *Vertical Difference*. To obtain terrain-specific features, we incorporate DTM information for the geolocated mesh area. Hence, n_{DSM} describes the height above ground terrain for all 3 vertices and the COG. [Kölle et al., 2019] showed that this feature is essential in the case of point cloud classification. [Rouhani et al., 2017] also encourage some kind of elevation feature, which is no height above ground, however, and therefore less expressive.

For each SPNH, similar to [Hackel et al., 2016], covariance features are calculated: *Linearity*, *Planarity*, *Variation*, *Curvature*, *Omnivariance*, *Anisotropy*, *Eigenentropy* and *Sum Of Eigenvalues*. The *Face Density* is defined as the total number of faces within the neighborhood, divided by the distance to the farthest face in the SPNH. The higher the face density, the greater the accumulation of triangles in this area. Similarly to the face feature *Vertical Difference*, the maximum vertical range for the complete neighborhood is captured by *Max Vertical Difference*. This feature gives information about the vertical extent of the current neighborhood. To inject additional terrain information, we calculate $\sigma_{n_{DSM}}$ - a measure for the vertical geometry variation within the neighborhood.

To obtain radiometric features, we extract texture patches for each triangle. The predominant color information for each face is captured by its median value. We use RGB color information, as well as its HSV color space transformed pendant, in order to ensure lighting independent features. We capture these medians for each SPNH as well. However, to obtain more color feature expressiveness, we calculate several histograms. The histograms are depicted by different levels in table 4.2. Each level depicts a different histogram bin discretization, with 9, 20, 64 bins for level 0, 1 and 2 respectively. We use these three discretization steps in order to obtain different levels of radiometric information granularity. However, the more bins, the sparser the histograms and thus the less meaningful information within the feature vector. Calculating three different histograms gives us more flexibility in crafting the feature vector, since all of them can be omitted easily in the final feature vector composition. In general, we expect from the radiometric features that they are beneficial in distinguishing classes that are geometrically similar. In particular in the case of *green space* and *impervious surface*. The features of all data splits (train, validation and test set) are normalized according to statistics of the train set. All features are zero-centered and the standard deviation is normalized to one. Formally speaking, by calculating feature vectors, the mesh segmentation basically turns into a data-point-based classification task. In total, for every face we calculate 749 features for each SPNH. However, only a subset of the feature vector is affected by the chosen neighborhood. Vertex and face features as given in table 4.2 only refer to the current face of interest and thus are independent of their neighbors. We are aware of the fact that many features need a great amount of training data. [Theodoridis and Koutroumbas, 2008] state that at least five training examples are required for each dimension in the representation. Our training set consists of 1.74 million (valid) faces and, thus, greatly exceeds this requirement. By calculating a large number of features, we are able to make an ablation study and investigate on which features are important (cf. section 4.5). Moreover, it is worth mentioning that the amount of features can be highly reduced when color histogram features are replaced by explicit 2D branches to the network.

Figure 4.2 gives a visual impression of several scalar features for Tile D, in terms of a point cloud representation, where each point depicts the COG of a face and the color coding depicts the magnitude of a specific feature. In the first row, the impact of features *Horizontal* (middle) and *Verticality* (right) is depicted. *Horizontal*, as expected, is very high on the ground plane and very low on vegetational structures, where all faces are blue. *Verticality* is complementary to that, showing high values on facade planes. Another good indicator for the ground plane is $\sigma_{n_{DSM}}$ (second row, left), with consistently very low values in ground terrain regions. *Face Density* (second row, middle) is very high in vegetational regions, where the

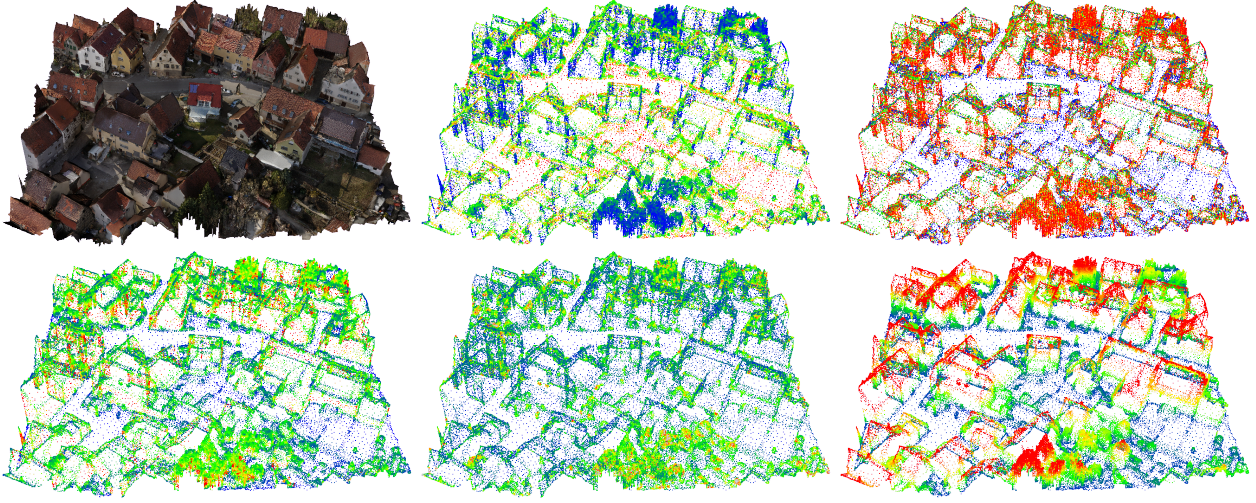


Figure 4.2: Visualization of influence of different features on Tile D. Each point of the point cloud depicts the COG of a face and the scalar value associated to the point depicts the magnitude of a feature. The color bar goes from blue (low feature value) to red (high feature value). First row from left to right: Textured mesh, *Horizontality*, *Verticality*. Second row: σ_{nDSM} , *Face Density*, *Vertical Difference*.

meshing process produces a large amount of small triangles to approximate complex tree, plant, bush, or hedge structures. Finally, *Vertical Difference* (second row, right) denotes the maximum difference of vertical components within one triangle. This feature nicely captures roof structures (and similarly large vegetation) with very high values, whereas planar structures show consistently low values.

Processing data input in *label* mode handles the semantically labeled mesh as input. In this case, the OBJ file is parsed to extract just one RGB triplet for each face, since the labeled mesh only uses materials instead of texture patches to encode semantic classes. Using a data set specific lookup table (LUT), the semantic label for each face can be retrieved from the RGB triplet. The 1-to-1 correspondence for faces between the textured and labeled mesh is accomplished via identical COGs. Figure 4.3 gives an impression of the above-mentioned SPNHs. The current face of interest for which the feature vector was generated is depicted in pink and the according SPNH (in this case level 2 with a radius of 1.5 m for both examples) is highlighted in green. To give an impression of the actual values within the feature vector we report them briefly in table 4.3. The table nicely shows the numerical differences of the features that are visually recognizable in figure 4.3. It is clearly conceivable through the *Inclination* that the main direction of the faces for the vegetation example is vertical and the *Face Density* is much higher than for the roof example. The *Face Area* for the face of interest in contrast, is much larger for roof example on the left.

	Roof example	Vegetation example
<i>Face Area</i>	0.395 m ²	0.004 m ²
<i>Median Inclination</i> of the neighborhood	45°	90°
<i>Face Density</i> in the SPNH	20.2 $\frac{\text{faces}}{\text{m}}$	68.7 $\frac{\text{faces}}{\text{m}}$

Table 4.3: Exemplary numerical values for some features of the face of interest and their SPNH, referring to the roof example in figure 4.3 on the left and the vegetation example, correspondingly on the right of figure 4.3.



Figure 4.3: Submeshes of SPNH level 2 (radius 1.5 m) depicted in green. The pink dots mark the current face of interest, for which the SPNH-dependent feature calculation is performed.

4.2 Convolutional Neural Networks (CNNs)

Artificial neural networks have long been a recurring topic of research. To keep this section concise, the history and basics of ANNs are roughly outlined, followed by a quick introduction to CNNs. The presented notes here cannot and do not want to be exhaustive but should only give the reader an introduction and point of reference for the techniques used in this thesis. For an in-depth review on this subject, the reader is kindly referred to [Nielsen, 2015, Goodfellow et al., 2016].

Pioneer work in this field ranges back as early as the 1940s, where Warren McCulloch and Walter Pitts introduced the first mathematical model for an artificial neuron [McCulloch and Pitts, 1943]. This idea was taken up in the 1950s by Frank Rosenblatt who established the perceptron concept [Rosenblatt, 1958]. In essence, the single-layer perceptron is a linear classifier for binary decisions of the form:

$$y = f(\mathbf{x}) = \begin{cases} 1 & \text{if } (\sum_{i=1}^n w_i x_i) + b > 0 \\ 0 & \text{otherwise} \end{cases} \quad (4.4)$$

where x_i are the elements of a feature vector \mathbf{x} with length n and the weight vector \mathbf{w} of the same length. Bias b is a term to translate the decision boundary and is independent of the input. In this case, the activation function for the perceptron is the unit step function. This became the first model being capable of learning weights based on the associated categories of input data. However, with the publication of [Minsky and Papert, 1969], the initial hype about the topic flattened out. They pointed out the limitations of perceptrons and showed that it is impossible to learn the XOR function, hence not being able to approximate non-linear functions.

With the introduction of backpropagation [Rumelhart et al., 1988], a second wave of research in ANNs commenced. Backpropagation is a method to calculate gradients in order to obtain the previously mentioned weights. Hence, it became possible to train multilayer perceptrons (MLPs). An MLP is the extension of the perceptron model, consisting of at least three layers: input, hidden and output layer. Please note that the term MLP might be misleading - the model does not consist of one single perceptron per layer. Instead, multiple perceptrons are organized in one layer. The MLP, or feedforward neural network, is a universal function approximator [Hornik et al., 1990]. f^* is the function to be approximated. In general, a classifier defines a mapping $y = f^*(\mathbf{x})$ as shown in equation (4.4). With notation here in accordance to [Goodfellow et al., 2016], the neural network defines a mapping

$$\mathbf{y} = f(\mathbf{x}; \boldsymbol{\theta}) \quad (4.5)$$

where θ are the parameters of the MLP to be learned and resulting in the best function approximation of the actual function f^* .

Backpropagation uses the chain rule to iteratively compute gradients for each layer within the MLP. In order to learn, that is, to adapt its weights, the MLP needs a loss function. The loss function describes the discrepancy between prediction results $\hat{\mathbf{y}}^{(i)}$ and ground truth labels $\mathbf{y}^{(i)}$, with i being the current sample. When dealing with image classification, $\mathbf{x}^{(i)}$ refers to a specific image, $\mathbf{y}^{(i)}$ its respective ground truth class, and $\hat{\mathbf{y}}^{(i)}$ the predicted class. A commonly used loss function for classification tasks is the cross-entropy (CE) loss:

$$J(\theta) = \frac{1}{N_i} \sum_i^{N_i} L_i(\mathbf{y}^{(i)}, \hat{\mathbf{y}}^{(i)}) = -\frac{1}{N_i} \sum_i^{N_i} \sum_c^{N_c} \mathbf{y}^{(i)}(c) \log \hat{\mathbf{y}}^{(i)}(c) \quad (4.6)$$

Here, $\mathbf{y}^{(i)}$ and $\hat{\mathbf{y}}^{(i)}$ are one-hot encoded and therefore the inner summation is over all potential classes N_c . The outer summation is over all images N_i within the current (mini-)batch. In order to tune the weights according to the loss function, an optimizer is necessary. The optimizer minimizes the objective function $J(\theta)$, where θ , as stated above, are the parameters of the model. $J(\theta)$ is minimized by updating the parameters of θ in the opposite direction of the gradient of the objective function. The gradient of the objection function is

$$\nabla_{\theta} J(\theta) \quad (4.7)$$

Gradient descent is the most established optimizer [Cauchy, 1847]. For gradient descent, the parameters of the model are updated as follows:

$$\theta = \theta - \eta \cdot \nabla_{\theta} J(\theta) \quad (4.8)$$

Where η is the learning rate and defines the step size of the gradient updates in order to reach a (local) minimum. η is arguably one of the most important hyperparameters [Bengio, 2012]. If η is too large, the (local) minimum might be missed due to too large update steps. If the learning rate is too small, weight updates are only marginal and the training process is slow. There are three different forms of gradient descent: (1) batch gradient descent - error for all samples within the training set are calculated, but the model weights are only updated after all samples have been evaluated. Hence, weight updates occur only once per epoch. Computationally, this method is very efficient due to the low frequency of updates and the necessity to hold the complete data set in memory. In general, this variant tends to smooth the error gradient and thus might produce stable learning curves. However, this smoothing also means, that the learned weights are very prone to overfitting. In contrast, (2) stochastic gradient descent (SGD) calculates the loss for every sample in the data set and accordingly updates weights after each sample [Bottou, 2013]. This is computationally quite expensive. Performance-wise, very frequent updates give a good impression on the learning progress and might lead to faster learning. On the other hand, updating the model so frequently leads to noisy gradient signals and thereby larger variance in the learning process. (3) Mini-batch gradient descent is a mixture of the previous variants. The complete data set (batch) is splitted into several chunks (mini-batches), for each mini-batch the model error is calculated and weights updated accordingly. This combines advantages of the afore-mentioned variants - more frequent updates than for gradient descent but computationally more efficient than SGD, thus, local minima can be avoided and convergence is more robust than using SGD. The mini-batch size is a hyperparameter and can be tweaked for different training runs. Large mini-batch sizes generally produces smoother learning curves, but converge slowly and have a lack of generalization ability [Keskar et al., 2016]. Small mini-batch sizes tend to converge more quickly, however the training process might be noisier. Please note that mini-batch gradient descent equals to SGD when setting the mini-batch size to 1.

[Fukushima, 1980] introduced the concept of the neocognitron, a hierarchical MLP inspired by the human visual primary cortex. They used this network to recognize handwritten digits. [Fukushima, 1980] was the basis for the convolutional neural network proposed by Yann LeCun [LeCun et al., 1998] for the very same task, handwritten character recognition. The algorithm was trained on the *Modified National Institute of*

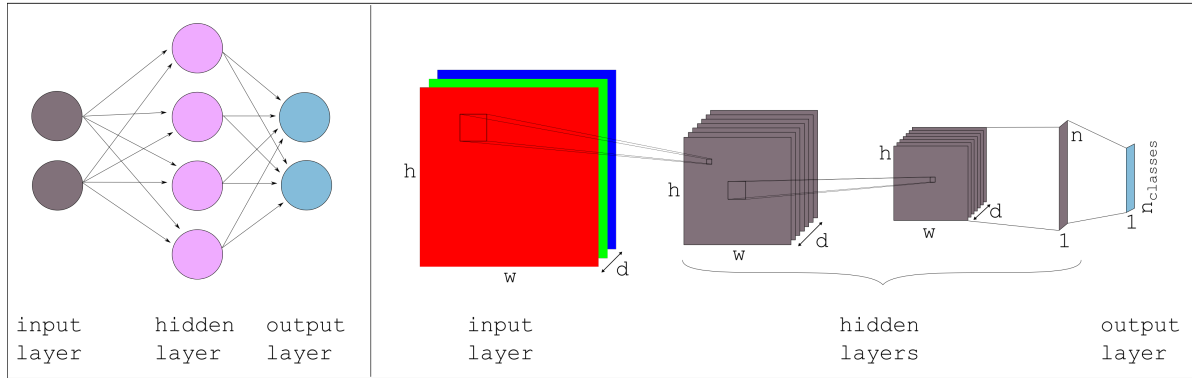


Figure 4.4: Left: Example for a fully connected neural network. Right: Example for a CNN. The input layer can be considered an ordinary image with width w and height h and three channels (red, green, blue), hence its depth $d=3$. The square located on the red layer depicts the receptive field, a hyperparameter, also known as filter size. Two consecutive convolutional layers are depicted as stack of gray activation maps, also known as feature maps. The depth of the volume is equivalent to the number of chosen filters for the convolutional layer, which again, like the filter size, is a hyperparameter. After the two convolutional layers follows a fully-connected layer with size n and the final fully-connected layer, the output layer, generates the class scores.

Standards and Technology (MNIST) database using backpropagation (which was not used for the neocognitron back then). Classification of digits based on MNIST is considered the **Hello World** for machine learning (ML) approaches nowadays. The most significant breakthrough for DL in recent time was in 2012, when [Krizhevsky et al., 2012] proposed AlexNet, an eight-layer CNN architecture inspired by LeCun’s earlier work. With AlexNet, they lowered the top-5 error on the *ImageNet Large Scale Visual Recognition Challenge* (ILSVRC) more than 10 %. ILSVRC, the most established classification challenge in computer vision, contains 1000 different image categories to recognize. Up until the introduction of AlexNet, shallow learning algorithms were state of the art and won ILSVRC, with a then reasonable top-5 error rate of 25 %. AlexNet lowered the error rate to 15.3 % and paved the way for DCNNs, with ILSVRC winning architectures that had an increasing number of layers year after year. In hindsight, the idea of CNNs introduced in the late 1980s proved to be successful. However, back then the lack of computational power (CPU, and, more importantly, GPU) to store and learn millions of parameters, and handle large amounts of training input, prevented further advances beyond MNIST classification. Convolutional layers are the main concept differentiating CNNs from regular ANNs. Within a regular neural network, all layers are fully connected, commonly referred to as dense layers. Figure 4.4 visually compares both systems. The right side of the figure depicts the concept of local connectivity. Instead of connecting all neurons in one layer to all layers of the previous layer, each neuron in a convolutional layer is only connected to a local region of the previous volume of activations. The volume is depicted with width w , height h and depth d in figure 4.4. Convolution is performed by computing dot products between the filters and the above-mentioned local regions in the input. Hence, by reshaping the input and filters, the convolution in one layer can be defined as large matrix multiplication. The convolution is followed by a non-linearity layer consisting of an activation function. Often, notations do not explicitly state the non-linearity layer but assume that this is directly coupled with the convolutional layer. Activation functions are responsible for the decision whether a neuron should fire (be activated) or not. Without the activation function, all ANNs would only perform a linear transformation based on weights and bias, $w \cdot x + b$, as depicted in the upper branch of equation (4.4). The activation function thus introduces non-linearity and gives the network the capability to approximate non-linear functions (and thereby becoming an universal function approximator). Early works used tanh or sigmoid as activation function for hidden layers. State of the art in the meantime, however, is rectified linear unit (ReLU), proposed to use for

restricted Boltzmann machines in [Nair and Hinton, 2010] and then later picked up by [Krizhevsky et al., 2012] within AlexNet:

$$f(x) = \max(0, x) \quad (4.9)$$

ReLU is computationally very efficient to implement, lowers the vanishing gradient problem and enables sparse activations [Glorot et al., 2011]. Pooling layers are used in-between convolutional layers to reduce spatial dimensions, i.e., the representation of the features maps. This decreases the amount of parameters in the network and simultaneously reduces overfitting. Usually, max-pooling is applied, in the most common form with a filter size of 2x2 and an according stride of 2. Which means, the maximum of the 4 values within each filter is extracted. This effectively downsamples the input by a factor of 2, indicated in figure 4.4 by the reduced size of the second feature map stack. The following fully-connected layers are responsible for higher order reasoning and the last layer generates the class scores. For classification tasks, softmax is usually used as activation function for the last layer. In-between dense layers it is common to use dropout as a regularization method to further reduce overfitting and improve generalization [Srivastava et al., 2014]. The dropout layer randomly deactivates a fraction d_r of the neurons, thereby enforcing the network to learn more robust features.

The advantages of CNNs over ordinary, fully connected neural networks, are manifold. Local spatial coherence is exploited. The convolution operation takes a local neighborhood into consideration. Neighboring pixels contain meaningful information, which in turn enables the concept of parameter sharing. The idea of the underlying assumption is as follows - if a specific feature performs well on location (r_0, c_0) , within an image, for instance, it is likely to perform accordingly well on location (r_1, c_1) . Hence, the neurons within each feature map use the same set of weights and bias. To give an example: if a convolutional layer has a receptive field of size $F=5$, depth $D=10$ and operates on a 3-channel input image, then each feature map shares one set of filter weights, that is, $10 \times 5 \times 5 \times 3$ weights plus 10 biases, accounting for 760 parameters in total. All neurons within one feature map accordingly use a set of $5 \times 5 \times 3$ weights plus 1 bias. This drastically reduces the number of parameters as compared to using fully connected layers. Convolutions can be efficiently implemented on GPUs, as will be picked up again in section 5.1.2.1, and hence are much faster to train. Pooling layers additionally reduce the number of parameters and provide a measure against overfitting.

In the following section we will make use of the here introduced concepts and leverage a 1D CNN to perform semantic segmentation of textured triangle meshes based on the features we depicted in section 4.1.2. Chapter 5 makes use of special types of CNNs, which will be discussed in section 5.1.2.1 and section 5.1.2.3.

4.3 Feature Based Semantic Mesh Segmentation

Given a textured triangle mesh $\mathcal{M} = (\mathcal{V}, \mathcal{E}, \mathcal{F})$, with $\mathcal{V}, \mathcal{E}, \mathcal{F}$ its vertices, edges and faces respectively, our goal is to process \mathcal{M} in a way that every face $f \in \mathcal{F} = \{1, \dots, N_f\}$ is assigned to a semantic label $l_c \in \mathcal{L} = \{l_1, \dots, l_{N_c}\}$. Here, N_f is the total number of faces within the mesh of interest and N_c is the number of semantic target classes. In our most fine-grained case, N_c equals 8. To do so, we pursue a supervised learning scheme. As described in section 4.1, we set up a training data set, which contains multi-scale feature vectors $X_f^{multi} \forall f \in \mathcal{F}$ and an associated label for each face l_c^f . For many photogrammetry and remote sensing related classification and regression tasks, RFs are still state of the art [Weinmann et al., 2015, Kölle et al., 2019]. Thus, we use the concatenated multi-scale feature vectors to train a RF classifier as a baseline. The concatenation of all SPNHs (level 0, 1 and 2) considers the fact that some features do not depend on the SPNHs (cf. table 4.2). Therefore, the concatenated feature vector X_f^{RF} has a length of 1901 (instead of 2247, 749 features times 3 branches, when naively stacking). We perform a grid search for parameters *amount of trees* and *depth of trees* to obtain the optimal parameter configuration for the RF. Details can be found in section 4.5.

To explore the potential of CNNs for semantic mesh segmentation, we adapt the architecture of [George et al., 2017]. The gist is described in section 2.2.2. This network consists of three input branches. Each branch is fed with the respective feature vector of the SPNH, depicted on the left in figure 4.5. The feature vectors per SPNH, X_f , have a dimension of 749×1 (with the first dimension denoted as N_x), whereas the feature vectors of [George et al., 2017] are slightly larger with dimensions of 800×1 . Although using more features, their implementation only considers geometric features. The first convolutional layer *conv0* of the original model has a filter size F of 15×1 . However, our implementation differs here. In our opinion, the filter size should be as large as X_f itself in order to capture context within the whole feature vector. If filter size of *conv0* is very small, only local correlations within the feature vector X_f can be captured (and connected). For image processing tasks, where pixels are the features, this is not a problem. Adjacent pixels can be considered semantically correlated. In our approach, however, this does not hold true since we are constructing the feature vector ourselves. The initial arrangement of single features within X_f is de facto random, but fixed after initialization. In order to decouple the local feature correlation, we use a kernel with the same dimensions as X_f (i.e. N_x) with a stride $S = 1$ and padding $P = \lceil \frac{N_x}{2} \rceil$. In addition, we define an unnormalized Gaussian weighting function \mathbf{W}_G with $\mu = 0$ and $\sigma = 0.4$ in the range $[-2, 2]$, discretized in N_x steps. Using the mentioned σ results in weights very close to 1 near the maximum of the weighting function. This normal distribution is centered at $\lceil \frac{N_x}{2} \rceil$. The output of *conv0* is element-wise multiplied by \mathbf{W}_G .

The Gaussian weighting reinforces the importance of those activations that were generated by the incorporation of the majority of features. In other words, central parts of the feature map are multiplied with values close to 1, whereas the tails of the feature map are less important, since increasingly less features contributed to the convolution and hence are multiplied by values closer to 0. The padded convolution is needed as otherwise feature maps would consist of merely one value (for each kernel). This would be a massive feature embedding. Here, we use padded convolution instead of fully connected layers as parameter sharing is only possible for convolutional layers. We use leaky ReLU as activation function and incorporate batch normalization. After each convolutional layer, we perform max pooling with $F = 2$, $S = 1$ and $P = 0$. The three scale branches are merged using depth concatenation, resulting in a dimension of 17856 (output dimensions of each branch: [93, 64]). Subsequently, the feature maps are passed through two dense layers with a dropout layer in-between, with the output size of the last dense layer being N_c . Finally, the output is passed through the softmax layer and delivers class probabilities in the range $[0, 1]$. To deal with the imbalanced classes in the training set, we use focal loss (FL) [Lin et al., 2017]. This loss was originally introduced for object detectors to tackle the extreme foreground-background class imbalance. Focal loss is defined as:

$$\text{FL}(p_t) = -\alpha_t (1 - p_t)^\gamma \log(p_t) \quad (4.10)$$

where p_t is the predicted probability of the ground truth label. α_t is an add-on to the original FL and contains the ground-truth-class-specific weight (lower class frequency means higher weight). γ is a non-trainable hyperparameter and was empirically found to work best for $\gamma = 2$ in the original paper. Intuitively, if $\gamma = 0$, FL becomes the cross-entropy loss (see equation (4.6)). Initially, we used a weight-scaled cross-entropy loss with weights obtained from the class distribution of the training samples. However, FL consistently produced better results.

With the network architecture depicted in figure 4.5, we perform several runs with different hyperparameter settings for learning rate, batch size, L2 regularization, and optimizers. We report our results in section 4.5.

4.4 Smoothing of Segmented Triangles

Generally, our real world is spatially smooth, that is, adjacent faces are more likely to belong to the same semantic class than to others. This observation can be used as prior knowledge in order to spatially smooth

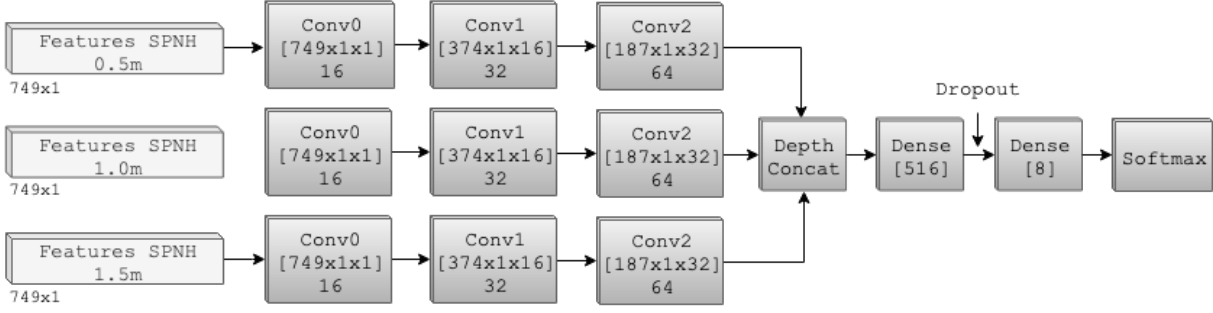


Figure 4.5: Architecture of the multi-branch 1D CNN. After each convolutional layer (input dimensions denoted in square brackets), a max pooling with $F = 2$ and $S = 2$ is performed. 16, 32 and 64 denote the number of used kernels.

the semantically enriched mesh. With increasing LOD the importance of smoothness increases because of higher intra-class variability. To give an example, more details will be visible and there will be higher spectral variability in high-resolution imagery when LOD increases. Due to the SPNH dependence of our feature vectors we already implicitly introduced the smoothness assumption to some extent. In order to explicitly smooth the achieved labeling, we employ a MRF. For each face, a label smoothing is performed. This smoothing is based on the local neighborhood defined by the value of feature *Face Density*. For each face, we define the Markov blanket \mathcal{N}_f dependent on the feature *Face Density* instead of using the mathematically correct Markov blanket consisting of adjacent faces only. As the fallback, we use a bundle of 5 closest faces. While increasing robustness, computation time remains almost the same. We report our results in section 4.5.

4.5 Semantic Segmentation Results

In this section we discuss the results obtained from the RF and CNN-based semantic segmentation. Both, CNN and RF, are trained on the training set partially shown in figure 4.1. Tests are performed on a machine with a NVIDIA GeForce GTX 1080 Ti GPU, 64 GB RAM (12 kernels) and PyTorch 1.0.

Table 4.4 registers achieved accuracies and inference times for all test tiles of both classifiers. The shown RF results correspond to a RF consisting of 250 trees, each having a depth of 25. These parameters have been picked by a grid search algorithm as they offer a good trade-off between accuracy and training time. To give an example, increasing amount of trees by 5, comes along with an increased training time (inference time) of 68 min (3s) while merely increasing accuracy by 0.016 %. For Tile A, RF trains 6.6 h (for best parameter configuration) and achieves an accuracy of 79.009 %. Subsequent MRF makes the result visually more appealing but decreases accuracy to 78.578 %. The best performing 1D CNN configuration uses a feature vector that is composed of all geometric features but only uses *Median HSV* per face and *Hue Hist Level 0* per SPNH as textural features. We use SGD as optimizer, a batch size of 50, and an initial learning rate of 0.001. Additionally, we tested the recently introduced optimizer AdaBound [Luo et al., 2019], a hybrid optimizer that is considered to unite benefits of the adaptive optimizer Adam [Kingma and Ba, 2014] and the traditional, static SGD. At the beginning of the learning phase, AdaBound behaves like Adam and transitions to SGD over time. Our training runs showed, that although AdaBound consistently decreased the loss faster than SGD, the actual learning plateau for both was the same, while the generalization gap for AdaBound was slightly larger. Hence, we stuck to traditional SGD.

The training time is less than 15 min. Inference time for Tile A is 14.69 sec and the achieved accuracy is at 79.868 %. A visual impression of the semantic segmentation result for Tile A is given in figure 4.6. Subsequent MRF smoothing slightly decreases the result (79.732 %), which will be further discussed in section 4.5.2.

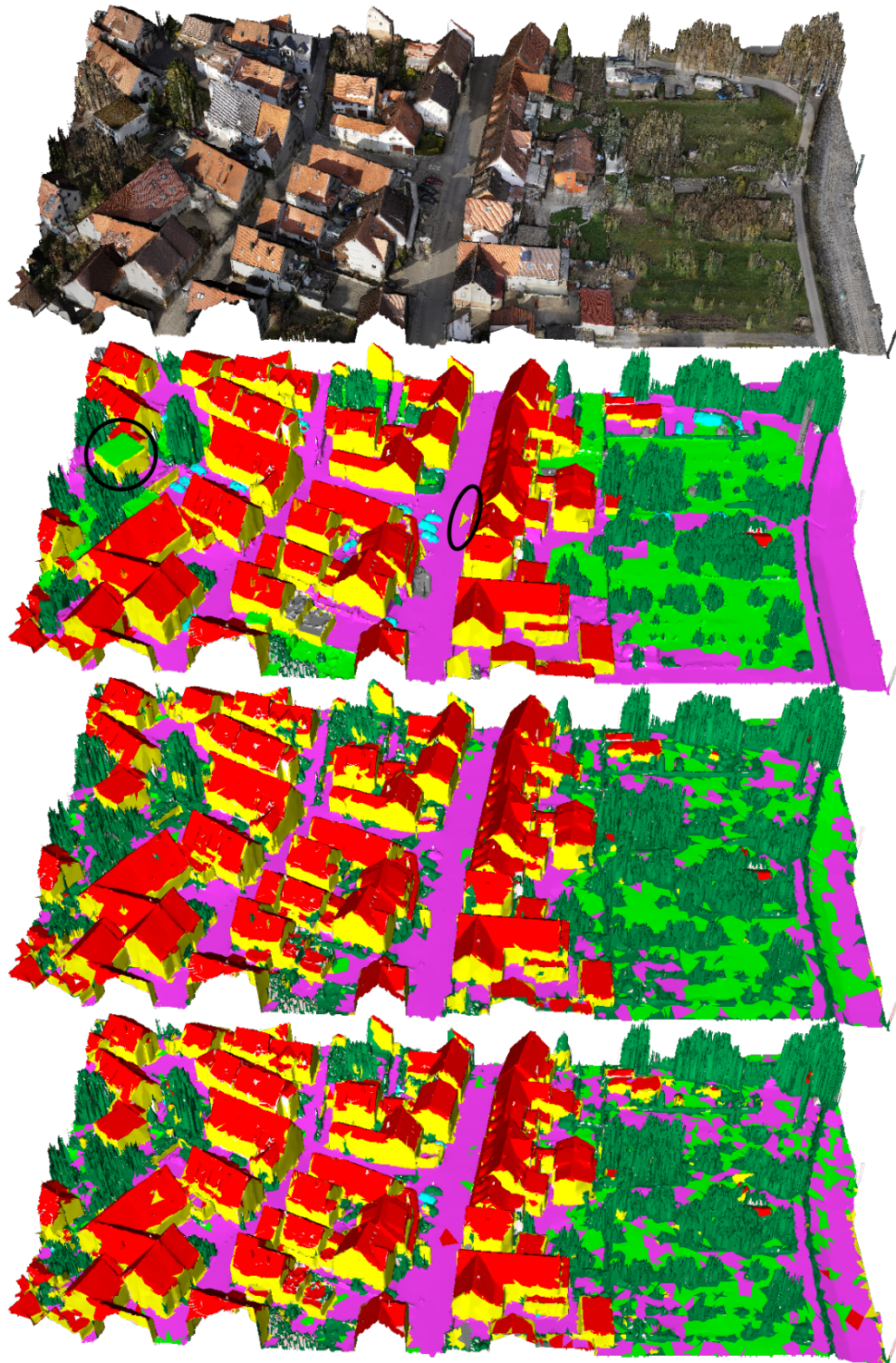


Figure 4.6: From top to bottom: Textured, ground truth labeled Tile A, RF prediction, 1D CNN prediction. In the ground truth you can see label noise: a flat roof (*black circle*) is labeled as *green space* and one face on the street is labeled as *building mass/facade* (*black ellipse*). Please note that the RF visually might indicate better performance on *green space*. Refer to table 4.5 for actual results. Both classifiers consistently predict the actual correct label for the flat roof and have problems on the street.

Tile	RF		1D CNN	
	RF	RF_{MRF}	CNN	CNN_{MRF}
Tile A 295884 faces	79.009 45.83	78.578 –	79.868 14.69	79.732 –
Tile B 226168 faces	76.584 36.14	76.513 –	76.425 11.43	76.824 –
Tile C 133932 faces	72.366 23.50	72.270 –	74.761 6.83	74.929 –
Tile D 116883 faces	48.559 19.82	48.279 –	48.336 5.82	47.766 –
Weighted Arithmetic Mean	72.543	72.298	73.209	73.213

Table 4.4: Prediction accuracies (in %, first row) and inference times (in s, second row) per test tile. There is no time depicted for the MRF smoothing as its processing time is independent on provided predicted label distribution. The weighted arithmetic mean is given for accuracies only. Best performances are marked in bold.

We are aware of the fact that overall accuracy is not as expressive as other classification metrics such as per-class precision and recall when dealing with highly imbalanced data sets. Since we experienced similar behavior throughout all our experiments, we therefore report per-class precision and recall in Table 4.5 once and limit ourselves to overall accuracy (OA) for the remaining tests. Using OA still has its justification because usually train and test splits are chosen in a manner that they contain similar class distributions. Both, RF and CNN achieve recall values close to 80 % for the *roof* class. This result is encouraging, since roof and building extraction is an important task in geospatial applications. Metrics for *building mass/facade* are slightly worse even though visual results imply that most building parts within the test areas are actually covered. It is noteworthy that the union of classes *vehicle* and *chimney/antenna* make up only approximately 1 % of the whole training set. Nevertheless, the 1D CNN is able to detect those classes to some extent.

Class		1. <i>building mass/facade</i>	2. <i>roof</i>	3. <i>impervious surface</i>	4. <i>green space</i>
Precision	1D CNN	62.237	77.110	66.418	40.997
	RF	64.613	77.933	71.213	43.622
Recall	1D CNN	63.600	78.206	36.756	14.446
	RF	58.767	78.063	30.433	12.740
Class		5. <i>mid/high vegetation</i>	6. <i>vehicle</i>	7. <i>chimney/antenna</i>	8. <i>clutter</i>
Precision	1D CNN	85.676	53.526	38.235	3.257
	RF	82.771	46.875	0.000	3.448
Recall	1D CNN	96.877	9.199	2.317	1.666
	RF	97.468	8.262	0.000	1.403

Table 4.5: Per-class precision and recall (in %) for predictions of the 1D CNN (*top row*) and RF (*bottom row*) for Tile A with best performing feature configuration (all geometric features but only *Median HSV* per face and *Hue Hist Level 0* per SPNH as textural features).

4.5.1 Influence of Feature Vector Composition

As described in section 4.3, our full feature vector is 749-dimensional. In the following, we refer to this as $X_{i,full}$. Training on $X_{i,full}$ delivers good results in terms of the predicted class coverage, however suffers from smoothness. Due to the sparseness of the histogram features, especially for level 1 and 2, we examined different feature vector compositions and their influence on predictions. Table 4.6 lists training runs with

Features Dropped	OA/WP in [%]
Config 1: None	74.9/71.4
Config 2: All Textural Features	75.4/71.8
Config 3: All Geometric Features	72.6/67.1
Config 4: All Textural Features except: Median HSV, Hue Hist Level [0,1]	76.8/74.3

Table 4.6: Prediction results of Tile B for different compositions of the feature vector. The first column denotes all features that were removed with respect to $X_{i,full}$. OA - Overall Accuracy, WP - Weighted Precision.

different configurations of the feature vector. Testing was performed on Tile B. Although both classification metrics, OA and weighted precision (WP) deliver similar results for all runs, the visualization of the results in figure 4.7 depicts the impact of different compositions of the feature vector. Please note how config 2 (all textural features removed; on the lower left) can no longer distinguish between *impervious surface* and *green space* and consequently only predicts the former. Removing all geometric features (config 3, cf. figure 4.7 in the middle on the bottom) results in failure of identifying buildings (*building mass* and *roof*) correctly.

4.5.2 Influence of Smoothing

Besides the implicit neighborhood smoothing by using multi-scale features, we enforce explicit smoothing by applying a consecutive MRF as described in section 4.4. In general, MRF processing produces visually more appealing results. However, this often comes at the cost of slightly decreasing performance (cf. table 4.4). Mostly regions with low-class-frequency occurrence are affected. For instance, if the classifier predicted a *vehicle* for a small number of faces, this aggregation might get smoothed away due to low support in the considered neighborhood. We claim that implicit smoothing reduces performance gain of the explicit smoothing.

4.5.3 Influence of Synthetic Data

Since data labeling is a very tedious task, we explored an approach to use synthetic data. The synthetic data was generated using ESRI's **CityEngine 2017.1**, which was previously introduced in chapter 3. **CityEngine** is capable of producing large-scale 3D virtual city models by relying on a procedural modeling approach. For each semantic class, category-specific rule sets are defined. We adapted the SynthCity data set provided by [Cabezas et al., 2015], consisting of sample scenes within **CityEngine**. Our goal was to use this data for pre-training in order to boost performance and achieve better generalization. However, performed tests have indicated that this does not lead to any improvement. While training accuracy on the synthetic data quickly converges close to 100 % and performance on a synthetic test set is quite promising, applying transfer learning and using the model parameters as initial weights for the real-world training runs did not provide a speedup or training accuracy boost. This is most probably due to several facts - the Manhattan world assumption, geometric and textural simplifications of the synthetic data, and LOD misalignment of the real-world and synthetic mesh. In addition, it is difficult to model vegetation in a way that it looks as realistic as possible while at the same time retaining a feasible geometric complexity. The used **CityEngine** framework provides three modes for the generation of vegetation: (1) pure texture - that is, no geometry of vegetation at all, only a planar textual representation. (2) billboards - diagonal-crossed planes with projected tree (or other vegetation) texture. (3) complex vegetation model with trunk and foliage. While the first two representations will insufficiently capture the characteristics of real-world mesh vegetation in most cases, the latter drastically increases the number of vertices and triangles. Thus, the computational load to process this synthetic mesh data is heavily increased for the third representation form, without guaranteeing to properly

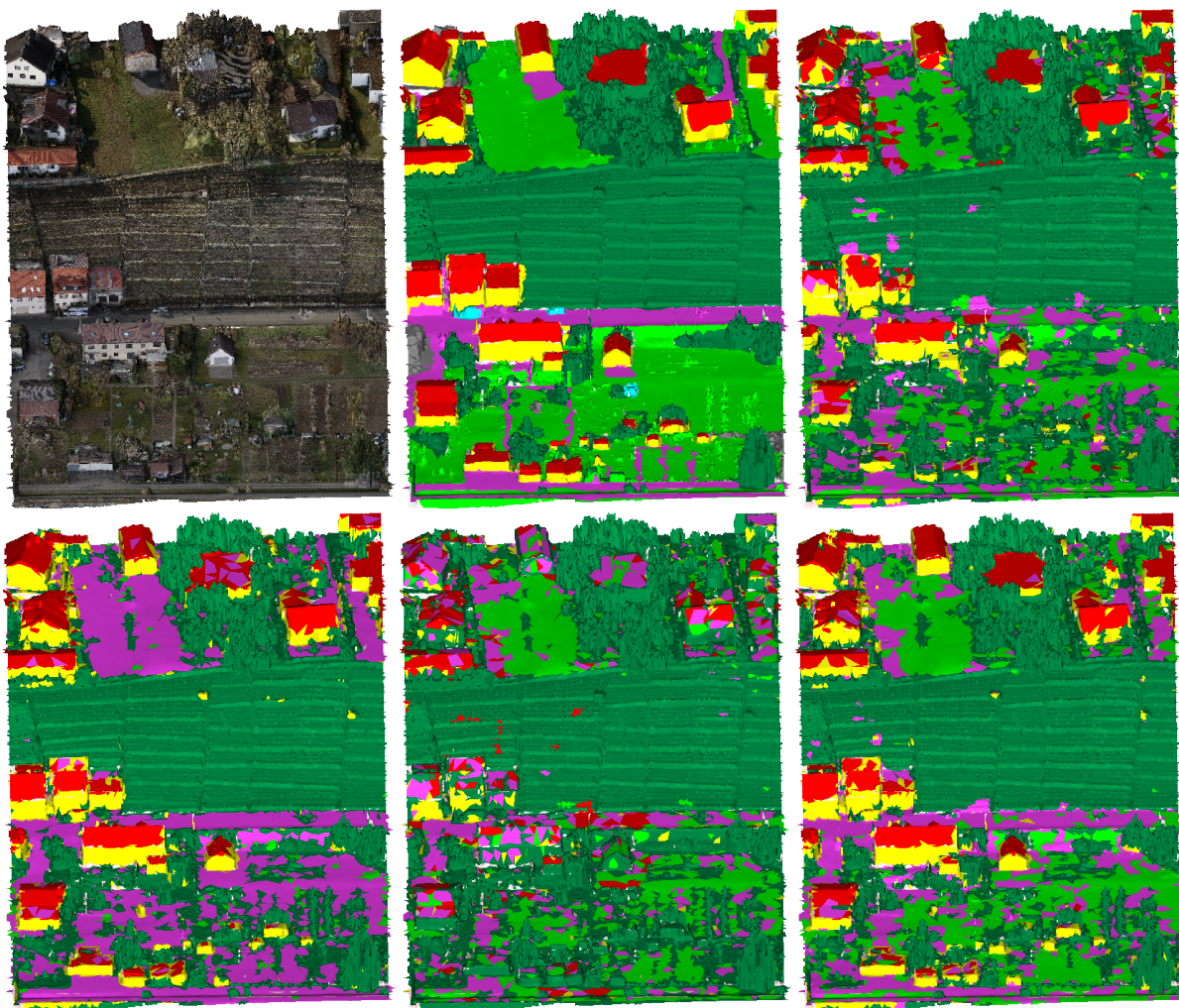


Figure 4.7: Impact of different feature vector compositions with respect to table 4.6. From left to right, top to bottom: Textured Tile B, ground truth labeled mesh, predictions for config 1, 2, 3 and 4, respectively. The figure is best viewed digitally.

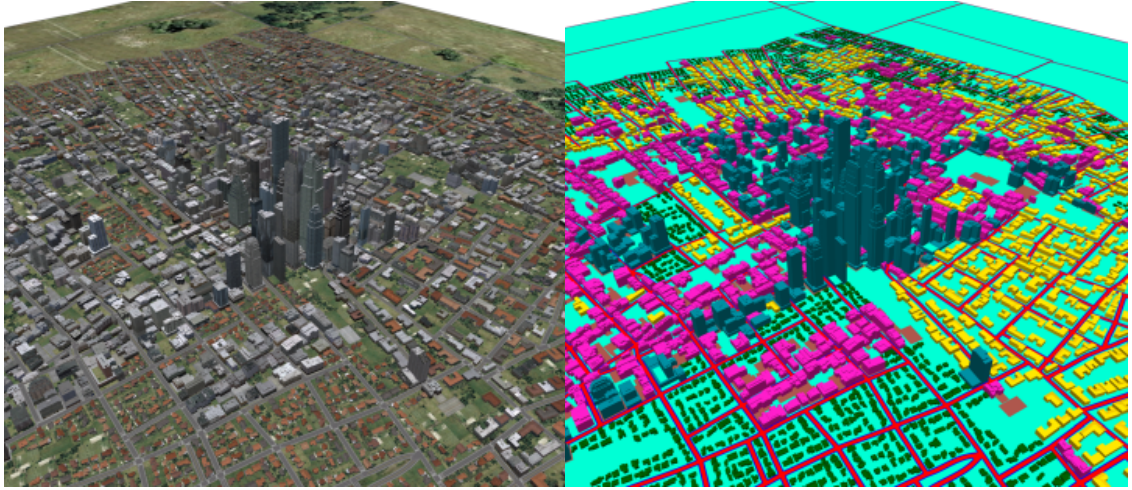


Figure 4.8: Exemplary data set from SynthCity within CityEngine. *Left*: Textured model. *Right*: Semantically labeled model.

adapt to the real-world case. Figure 4.8 depicts an example of a procedurally generated synthetic scene. In general, (pre-)training with synthetic data is a promising way and is commonly used in robotics, e.g. to learn grasping [Bousmalis et al., 2018], which is beyond the scope of this thesis. A vast amount of automotive approaches [Lee et al., 2018, Chen et al., 2018b, Bewley et al., 2018, Tremblay et al., 2018] use synthetic data, with frameworks such as SYNTHIA [Ros et al., 2016], Sim4CV [Müller et al., 2018] or AirSim [Shah et al., 2017]. Hence, we see potential to use such means for urban segmentation as well, once simulated data can be modeled to behave more like actual real-world data obtained from DIM and subsequent meshing. That is, no Manhattan world assumption, irregularities in geometry and texture, suitable representations for vegetation, as well as appropriate LOD scaling.

4.6 Summary for the Semantic Enrichment of Textured Meshes

This chapter outlined an approach that combines feature engineering and feature learning for the semantic segmentation of urban triangle meshes. For each face, a multi-scale feature vector is computed and serves as input to a 1D CNN. For the fine-grained distinction of 8 classes we achieve accuracies close to 80 %. This is slightly better than a RF based classifier. Moreover, training the CNN is faster than training the RF. Inference is faster as well. It has to be noted, that running time of RF and CNN is not fully comparable, since the `scikit-learn` implementation of RFs is running on the CPU and the `PyTorch` implementation of our 1D CNN is running on the GPU. [Liao et al., 2013] report a RF GPU implementation that is ten times faster than `scikit-learn`. However, even with such an improvement, the CNN training currently is still three times faster. DL frameworks are specifically relying on CUDA enabled GPUs with NVIDIA CUDA **deep neural network library** (cuDNN) that use highly optimized implementations for learning routines such as forward/backward convolutions and pooling. Coupled with efficient CPU based batch handling, this is expected to scale better on large data sets and models than RFs, which is important when processing entire cities.

The detection of buildings is an important application-oriented task in geodata processing. Both classifiers deliver encouraging results for *roof* and *building mass* extraction although being not explicitly trained for this task. Moreover, the CNN detects a fraction of the *chimneys/antennas* whereas the RF completely fails. MRF-based smoothing of the segmentation output further enforces smoothness, yet does not consistently improve our results. The class distribution of our data set is quite imbalanced - for example, we currently

have very few samples for the *chimney/antenna* and *vehicle* classes. Normally, class imbalance is cured using data augmentation, which is not trivial for our approach. In general, there is a lack of large-scale labeled real-world data and benchmarks that represent well the complexity and variability of urban scenes. Thus, it is not possible to publicly evaluate mesh-based semantic segmentation approaches. As a first step, we labeled an urban scene manually on our own. With this in mind, one big challenge is to provide annotated data from a variety of scenes. Crowdsourcing seems to be a suitable approach to tackle this problem. We tested the incorporation of procedurally generated data by means of ESRI's **CityEngine**. While exclusively training and testing on synthetic data works quite well, this does not transfer to the real word data. Potential reasons are: a strict Manhattan world assumption, geometric and textural simplifications of the synthetic data and LOD misalignment of the real world and synthetic mesh data.

Currently we do not exploit the full potential of texture information as we rely on a 1D feature representation, that is, histograms of colors. In the future, additional branches could be introduced for texture processing. These branches could either explicitly extract features from the texture patches or even perform a classification only based on the texture. The per-triangle classification could serve as input for a voting scheme for the final semantic class decision for each triangle. Additionally, random search or Bayesian optimization could be used for better performance of the feature vector composition. Using LiDAR as base for the mesh geometry preserves the option of additionally using LiDAR features per point in the future. We anticipate further improvement when ground truth does not suffer from label noise and/or using a 3D mesh instead of a 2.5D mesh. For example, the feature *nDSM* is more expressive in a 3D scenario (becoming *Relative Height Above Ground* consequently). To further increase capacity, a CNN ensemble could be employed where each network within the ensemble processes a different composition of the feature vector. In the longer term it is of course more desirable to avoid feature engineering and to design an end-to-end (Graph-)CNN pipeline instead. However, this requires rethinking CNN architectures for irregular 3D data input with adjacency information and is still in very early stages.

Chapter 5

Building Interpretation

In chapter 3, reconstruction relies heavily on the geometry of the building point cloud. This approach can be a useful tool for automated building model generation, fails however if accuracy is limited or the geometry is too complex. Moreover, as discussed in chapter 4, point clouds are actually an intermediate product in the photogrammetry community. Triangle meshes come with the benefit of textures. Chapter 4 outlined an approach to identify *building masses/facades* from a mesh representation. Once actual buildings are identified in an urban mesh, building interpretation can be based on facade imagery.

Semantic information as required for a multitude of applications like urban planning and infrastructure management includes building use, the number of dwelling units and more [Hecht, 2014]. An essential feature, from which several other metrics can be derived or at least approximated, is the building use. Therefore, we see a need for large-scale automatic building category classification. The following approach proposes an approach to leverage Google’s region-wide available Street View data and link the inherent buildings with data from the digital city base map provided by the City Survey Office Stuttgart. To extract only building-relevant parts from the Street View data, the crawled images are pre-processed. Therefore, we utilize metadata provided by the Street View API¹ and take advantage of a DL framework for semantic image segmentation [Long et al., 2014] to analyze our data for relevant content. Based on the information obtained in the crawling process we try to link image content with building polygons in the ground truth. The outcome is a tuple of building images and its corresponding building category. This data is then used to train a classifier. With the trained classifier it is possible to predict building categories for new input images.

For now, we want to distinguish between five different building use types. The classes *commercial* and *residential* are defined for a singular use of a building, while the class *hybrid* represents a mixture of these two use classes – e.g. shops and apartments in the same building. The class *specialUse* represents the remaining buildings not matching the other three classes, like schools and churches. Finally, the class *underConstruction* contains buildings being under construction, independent of their actual use.

The remainder of this chapter is structured as follows. In section 5.1 the automated generation of training data using Street View imagery is described. Crawling the image data is covered in section 5.1.1. Section 5.1.2 depicts the selection and pre-processing of images to provide suitable image patches for classifier training. In section 5.1.3 we elaborate on linking image patches to existing semantic information using coarse georeferencing information from Street View. Using this data, a CNN for building use classification can be trained. This process is described in detail in section 5.2. CNN training is covered in section 5.2.1 and results thereof are presented in section 5.2.2. Finally, section 5.3 introduces the use of class-activation

¹<https://developers.google.com/maps/documentation/javascript/streetview>. URL date: April 2019

maps, investigates the influence of different representation forms on the classifier’s decision, and presents an application of CAMs for abstract renderings.

5.1 Automated Generation of Training Data Using Street View Imagery

As previously mentioned in chapter 4, a crucial element in performing classification tasks is to obtain an appropriate number of training samples. Frequently, these are available from data sets and benchmarks in computer vision and machine learning. The SUN database [Xiao et al., 2010] consists of almost 4000 object categories but there are only slightly over 1000 images containing buildings. ImageNet [Deng et al., 2009] provides over 20,000 indexed synsets (synonymous word fields) and over 14 million images in total. There are also several benchmarks for urban scenes – [Geiger et al., 2013] developed a mobile mapping platform and host KITTI, a benchmark with data for a variety of vision tasks from stereo matching, over scene flow to semantic segmentation. Likewise, the CITYSCAPES dataset provided by [Cordts et al., 2016] contains scenes from 50 cities with corresponding semantic pixelwise annotations for each frame, obtained by a windshield-mounted stereo camera system. The even more extensive Mapillary Vistas data set for semantic segmentation of street scenes was introduced by [Neuhold et al., 2017]. For some of the data sets GPS information of the car’s trajectory is available. However, for our task, these datasets are not suitable since we aim on assigning specific usage categories to buildings. We pursue another path and make use of municipal surveying data in combination with a publicly available image source. This way we can narrow down and merge the variety of building categories, and enforce correctness of ground truth. There are several reasons why we follow the proposed framework, when there are already CNNs that classify hundreds of categories with a reasonable level of correctness, including classes like *apartment building* or *office building*. First, those very deep CNNs developed by companies are fed with massive amounts of training data – not everyone can provide or produce such huge collections of training samples. Moreover, large CNNs have a broad range of category types they cover while our work aims on a small subset of those classes. We are not interested in classifying a plethora of different categories, but rather very few, with potentially high intra-class variance. The evaluation of state-of-the-art approaches with a multitude of classes is frequently based on the top-5 error, however, since we aim on the determination of a rather limited number of classes at a rather high reliability, the top-1 error is our main interest.

5.1.1 Crawling Street View Imagery

The actual crawling is implemented in Java Script based on [Peter, 2015] modified for our use. As output from the crawling process, we obtain a list of positions P_i (longitude λ_i , latitude ϕ_i) and headings κ_i , where $i = 1, \dots, N$ with N as the total number of crawl positions. By dragging the Google Maps marker one can define the initial crawling position. Using the Street View API, the crawler searches for the next available panorama based on the current position. Figure 5.1 shows the crawling interface with the initial Street View position on the left and all crawled panoramas on the right. We use two different modes of crawling: panorama-link based and random sampling. The first method successively visits the link nodes stored in the current panorama until a predefined total number of panoramas is fetched. However, this method only returns the center heading κ_{c_i} of the street view car for this position. Therefore, when using panorama-link based method we add 90° to κ_{c_i} – thereby we obtain frontal views of the buildings. When using the random sampling technique, we generate random offsets for latitude and longitude, thereby performing a random walk of the geographical position. To prevent from excessive divergence we reset to the initial position in predefined intervals. Based on the randomly sampled positions we then search for the nearest panorama and calculate the heading. Outcome of both crawling processes is a list of 2D geographic coordinates and a corresponding heading κ_i . We use this data together with the parameters pitch Φ and field of view (FOV)

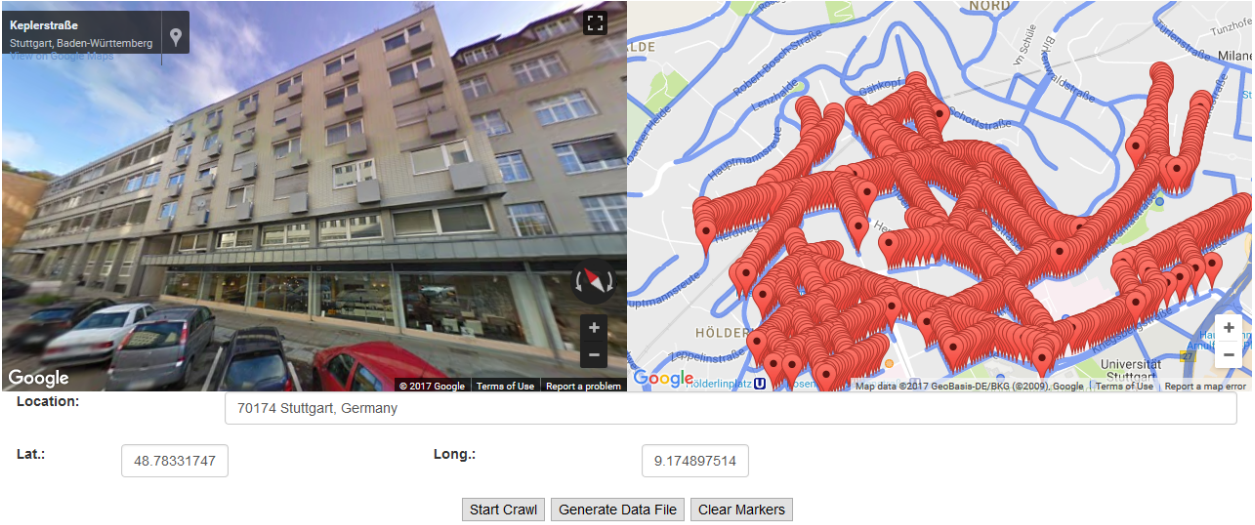


Figure 5.1: GSV crawler interface to select a seed point for the panorama collection. *Left*: Initial crawling position. *Right*: Crawled marker positions after panorama-link based sampling.

to query an image I_i as part of the panorama via the Street View API. Φ is measured positively looking upwards with respect to the camera’s initial horizontal position. We chose $\Phi=15^\circ$ and $FOV_{horizontal} = FOV_{vertical} = 90^\circ$ to ensure that also larger buildings are covered.

5.1.2 Extraction of Building-Relevant Images

We aim on the extraction of good training data, that is, images with clear view onto only one single building in center. However, many of the initial crawled images do not meet those requirements due to occlusions or blurred imagery. Thus, after fetching the Street View data we pre-process all images $I_{1...N}$ to extract only samples with relevant content. One tool we use to analyze the images is a reimplementation of a fully convolutional network [Long et al., 2015] provided by [Caesar et al., 2016]. This end-to-end/pixel-to-pixel trained network uses raw images as input and produces a semantic pixelwise labelling. While section 4.2 outlined the principle of CNNs in general, the following section briefly describes the concept of FCNs.

5.1.2.1 Fully Convolutional Networks

Common CNNs, as discussed earlier in section 4.2 performing classification tasks deliver one value as output. To obtain this value, the back-end of the network usually consists of one or more fully connected layers, with the last one having the number of desired classes as output. However, to solve semantic segmentation tasks for images, it is necessary to extract pixel-wise information. To accomplish this, FCNs leverage an architecture depicted in figure 5.2. Essentially, an FCN uses a CNN originally designed for classification tasks, (VGG-16 to be specific [Simonyan and Zisserman, 2014]), removes the final classification layer and transforms all fully connected layers to convolutional layers. To obtain a pixel-wise class assignment, that is, a semantic segmentation, 1×1 convolutions are used with the channel dimension according to the number of different classes from the training input. However, due to the convolution and pooling layers, the resulting output map is coarse. Hence, *transposed convolution*, or often (misleadingly) referred to as *deconvolution* layers, are used to upsample the coarse maps to dense pixel predictions with dimensions equal to the input image. A transposed convolution does not revert the process of a previously performed convolution. In contrast, it merely generates a feature map with dimensions according to the size of the input before an input was

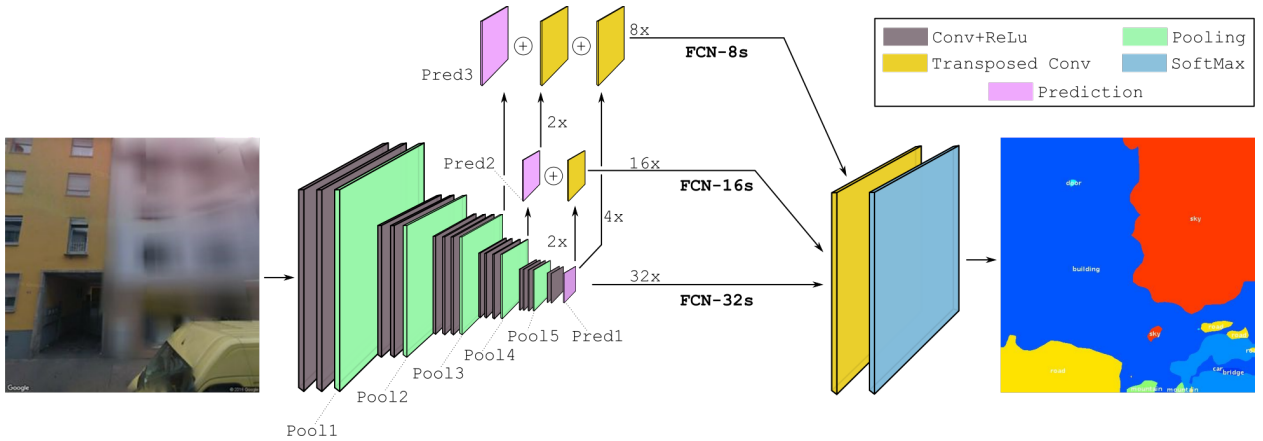


Figure 5.2: FCN architecture. This figure depicts three versions of FCN: 32s, 16s, and 8s. The naming stems from the stride size as will be discussed below. **FCN-32s**, the single stream net, upsamples stride 32 predictions back to individual pixels in one step. **FCN-16s** combines predictions from the final layer **Pred1**, upsampled by the factor 2, with predictions after **Pool4** at stride 16. **FCN-8s** goes one step further and additionally incorporates predictions **Pred3** after **Pool3** with 2x upsampled **Pred2**, and 4x upsampled **Pred1** at stride 8. Please note, that each version would actually produce a different segmentation map (with **FCN-8s** being the most detailed result), for the sake of clarity we only show the result of our actually used **FCN-16s** inference.

passed to a convolution layer. One intuition behind the process is to think of it as direct convolution where respective zero padding is introduced to the input. However, this leads to many multiplications by zero. Therefore, real-world implementations, in contrast, rely on the principle that a kernel defines a convolution but depending on how forward and backward pass are computed, this can either be a direct convolution or a transposed convolution [Dumoulin and Visin, 2016].

In the meantime, there are even more sophisticated approaches for semantic segmentation, such as **UNet** [Ronneberger et al., 2015], **SegNet** [Badrinarayanan et al., 2017], or **DeepLab** [Chen et al., 2017]. All of them rely on an encoder-decoder architecture. The encoder can be considered the left part of the architecture shown in figure 5.2. By consecutively convolving the input, the encoder maps the data into a feature space representation. This feature space is also referred to as latent space. The decoder is essentially a deconvolution network. It takes the latent space representation, performs an upsampling via transposed convolution (or, to introduce another synonym, *fractionally strided convolution*) and finally delivers an activation output map with the same size as the original input. In contrast to the shown FCN, the above-mentioned architectures use a denser representation for the decoder stream and incorporate skip connections between encoder and corresponding decoder layers to transport hierarchical feature information.

We use the **FCN-16s** model, trained on the SIFT Flow data set with roughly 3000 images and their corresponding pixel labels. As we do not need the sharpest segmentation lines possible in order to make assumptions about the semantic main content of an image, we rely on the middle branch of figure 5.2 as a trade-off between accuracy and inference speed. The SIFT Flow pre-trained model was chosen, because this data set contains for us essential building and building part classes. Other data sets such as *Pattern Analysis, Statistical Modelling and Computational Learning* (PASCAL) visual object classes (VOC) do not contain these classes. In total, there are 33 semantic categories ranging from *awning*, *balcony*, *bird*, *over mountain*, *person to tree* and *window*. However, there are not only semantic, but also geometric labels – the FCN can learn a joint representation and predict both. Not all of those classes are relevant for our purpose. Effectively, we only want to detect whether or not a building is the actual main content of the current image. Hence, we merge several classes – for example, we merge *awning*, *balcony* and *window* to the *building* class. Similarly, we merge *grass* and *tree* to the *plant* class. The following section describes how the FCN can be used to filter out unsuitable image candidates.



Figure 5.3: Input image and corresponding output from the FCN evaluation. The semantic class *building* is depicted in blue, *sky* in red, *road* in yellow, *plant* in green and *car* in bright blue, respectively.

5.1.2.2 Handling of Occlusions and Blurred Images

As stated earlier, we have to ensure, that the main image content is the building of interest. Thus, as a first step of processing the crawled urban imagery, we use the described FCN to perform a pixel-wise segmentation. Employing the merged classes introduced in the previous section we obtain results like depicted in figure 5.3 on the right. If the main content of our segmented image consists of *plant* or *car* pixels, we discard this image.

Each building owner has the legal right to demand Google to make his or her private property unrecognizable within the Street View data. Google approaches this the same way they anonymize people – by blurring the affected buildings. Obviously, we want to discard those images since there is no actual content provided. There has been a lot of work on edge-based blur detection [Ong et al., 2003, Narvekar and Karam, 2011]. In fact, edge detection delivers quite consistent results in our case, as shown in figure 5.4. However, as we incorporate the aforementioned FCN, we can make use of a particular property when evaluating images. When using this framework, blurred regions are typically classified as *sky* or *sea* pixels and can thus be detected easily. This way, there is no additional computation overhead and by passing an image through the FCN, we are able to simultaneously filter for occluded buildings by checking for vegetation or vehicles as main image content, and omitting blurred samples if there is an unusual high number of pixels being classified as sky/sea. It is semantically not possible to have water areas in the upper part of our image due to the way the GSV acquisition platform is built and the pitch angle we chose to extract crops from the original panoramas. Similarly, it is not possible to have *sky* pixels in the lower part of the image regions. Hence, these are legitimate heuristics for the filtering process. The following sections presents an alternative to the currently used FCN-based pre-processing. This alternative is a regional CNN and would ultimately make any pre-processing redundant.

5.1.2.3 Regional Convolutional Neural Networks

Faster R-CNN [Ren et al., 2015] is an object detection framework and evolved from two previous iterations: **R-CNN** [Girshick et al., 2014] and **Fast R-CNN** [Girshick, 2015]. Given an input image, R-CNN heavily relies on the selective search algorithm [Uijlings et al., 2013] to generate object proposals. Subsequently, each bounding box from the object proposals is passed through a modified **AlexNet** architecture to compute CNN features. These features are then fed to a support vector machine (SVM) for class predictions. In the last step, the bounding box coordinates for valid predictions are further refined by a linear regressor. Although this approach worked quite well, it is slow due to the following reasons: (a) the CNN forward pass has to be

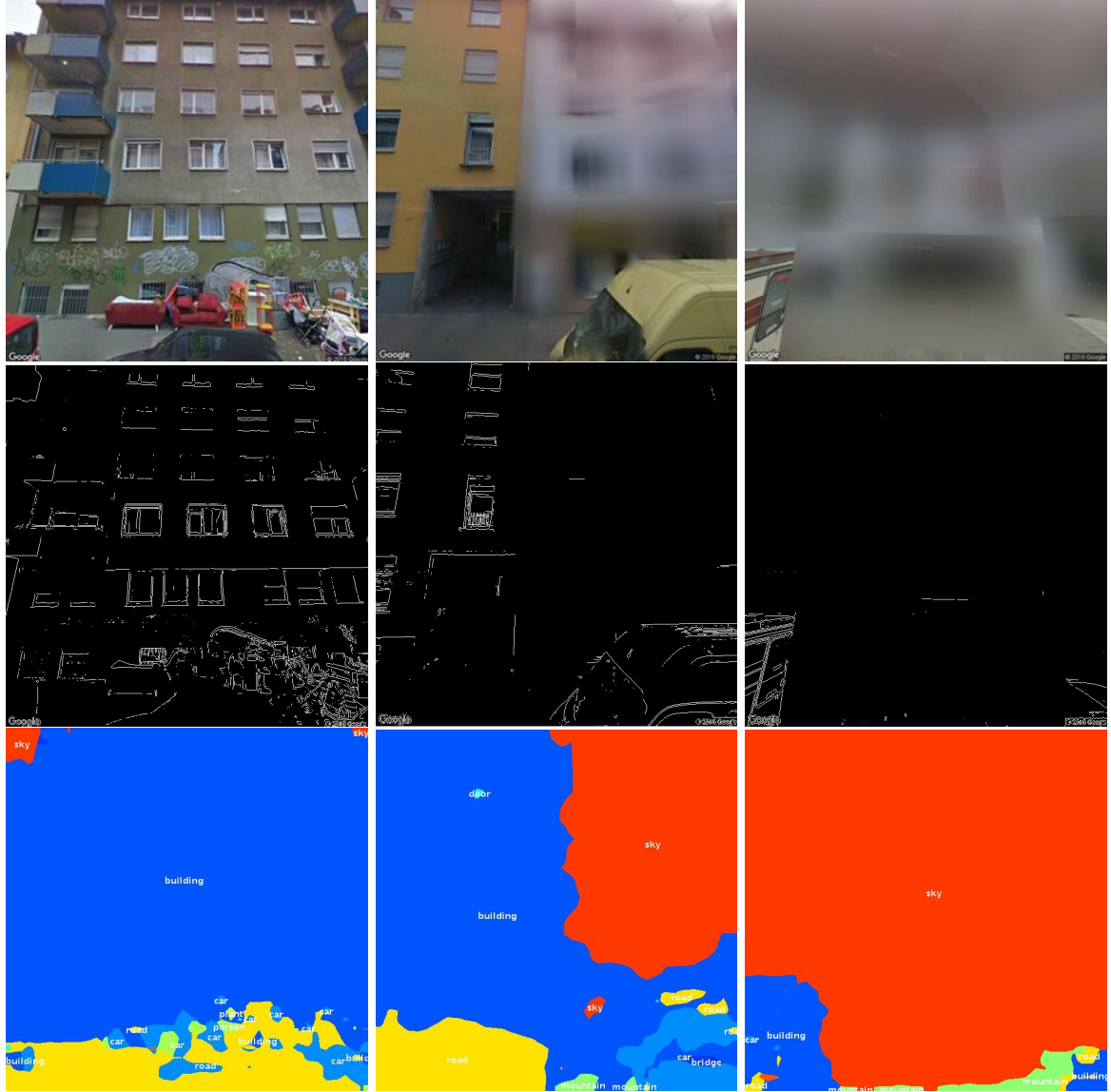


Figure 5.4: Top row: Images from GSV crawling process with ascending level of blurriness. Middle row: Corresponding edge images. Bottom row: Results from FCN based semantic segmentation. The color coding is consistent with figure 5.3.

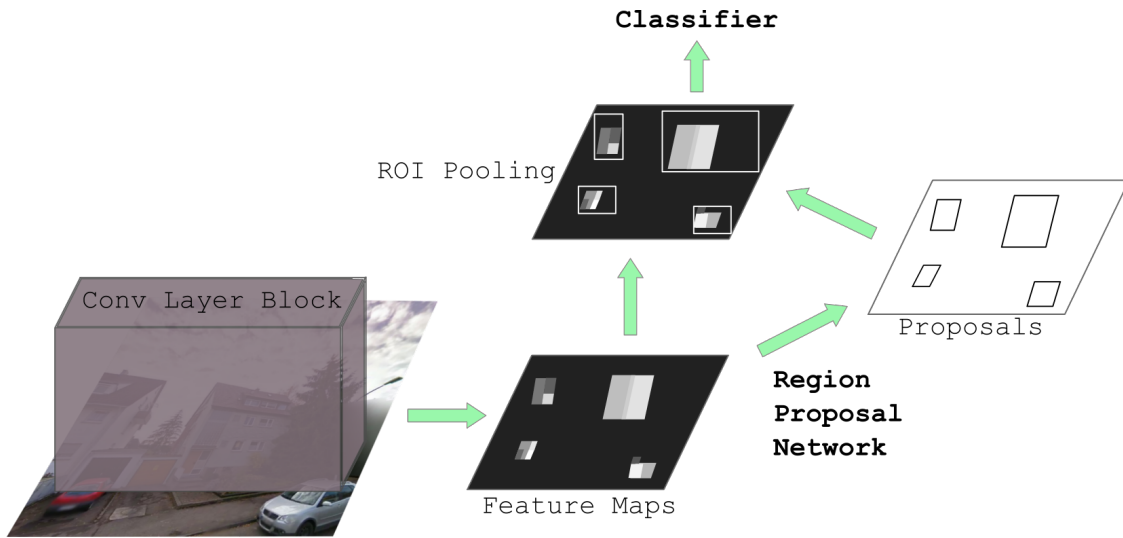


Figure 5.5: Architecture of **Faster R-CNN**. Figure adapted from [Ren et al., 2015].

performed for every region proposal in every image, (b) three different models have to be trained - the CNN for image feature generation, the SVM for class prediction, and the regressor to tighten the final bounding boxes. Additionally, selective search is mandatory for the generation of region proposals. Yet, selective search is not trainable, so R-CNN is only as good as its region proposals. Some of those drawbacks were addressed by the consecutive algorithm, **Fast R-CNN** [Girshick, 2015]. Instead of running CNN inference on every region proposal, **Fast R-CNN** performs one inference for the whole image and generates feature maps. The bounding boxes provided by selective search serve as masks to extract corresponding regions in the feature maps. To generate fixed-size representations for each region proposal, a region of interest (RoI)-pooling layer is introduced. RoI-pooling reshapes the feature region proposals to a compact, fixed-size representation. Hence, each RoI can be passed to a softmax layer (instead of the previously used SVM) to generate the classification result, and to a bounding box regressor to refine box shapes. This way, the previous three separate models could be merged into one trainable network. **Fast R-CNN** still relies on region proposals from selective search, though. The next iteration, **Faster R-CNN**, addressed this drawback by introducing a **region proposal network** (RPN). As depicted in Figure 5.5, the feature map generation is identical to **Fast R-CNN** but now the RPN is used to generate bounding boxes. Essentially, the RPN generates k anchor boxes of different aspect ratios and scales by passing a sliding window over the previously generated features maps. The RPN is comprised of two branches, the regression layer (a) and classification layer (b). For each of the k anchor boxes they respectively generate: (a) the coordinates of the bounding box, that is, the center coordinate plus width and height ($4k$ values), and (b) an objectness score, i.e., the probability whether a proposal is an object or not ($2k$ values). To sum up, the RPN generates bounding boxes containing objects and passes them to detection network previously described for **Fast R-CNN**. Again, the region proposals are passed through a RoI-pooling layer, and finally classification results together with regressed bounding box coordinates are generated. Such a two-stage object detector was employed in the following.

Inference within an FCN-based architecture is time-consuming and, more importantly, labels are only class-aware, not instance-aware. For our purpose it is desirable to distinguish between different object instances. Therefore, we trained an implementation of **Faster R-CNN** [Henon, 2017] on a subset of the Open Images Dataset [Krasin et al., 2017] from scratch. We used three classes: *building*, *tree* (container class for vegetation) and *automobile* (container class for vehicles) with around 23.000, 44.000 and 34.000 images, respectively. These are split into roughly 80 % training and 20 % validation set. Since this merely served as a feasibility study, we do not provide in-depth results on a test set. Instead, the trained **Faster R-CNN** model was applied to the originally crawled GSV imagery, that is, the samples before FCN-based filtering. Figure 5.6

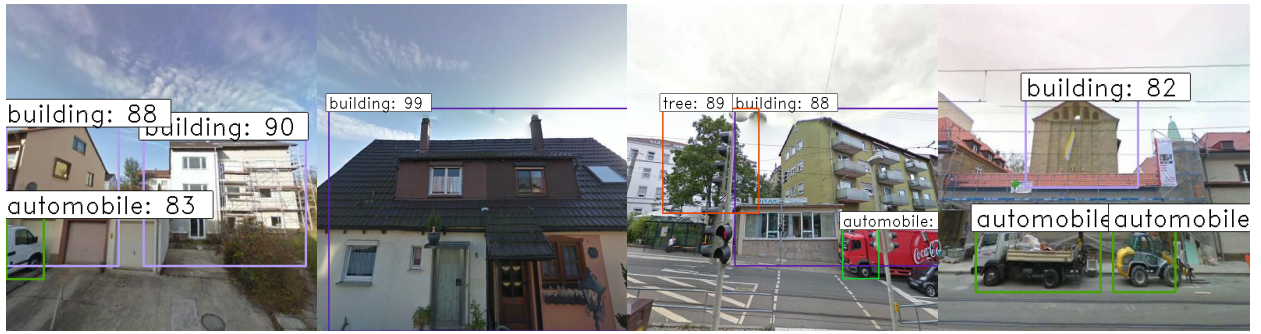


Figure 5.6: **Faster R-CNN** box detection results and associated probability for predicted class.

depicts some inference results and shows that in principle it is possible to detect (a) large bounding boxes of buildings which can be considered good training samples, and (b) objects that are occluding buildings of interest, such as vehicles or trees. Thus, a simple overlapping bounding box check can be performed in order to discard training samples. While the data set presented in this work was generated with the aforementioned FCN processing, future work should rely on a R-CNN-based approach. Due to these reasons, the concept of regional CNNs was presented in this section. They are a powerful tool and a proposal for a regional CNN-based pipeline will be picked up again in section 5.4. It is important to note once again that we perform image classification with the crawled data set. Each image is associated with a one-hot label vector. The human-given label corresponds to the building in the center of the image. We are aware of the fact that this lowers the performance (see last column of figure 5.11), but it is unavoidable in an only class-aware setting as street-level images almost always contain more than one building.

5.1.3 Registration of Imagery with Cadastral Data

Our ground truth data consists of a 2D shape file with ground plan polygons for each building. Every polygon is enriched with several aspects of semantic information like address, communal district, building block number and, especially of our interest, building use. For each building polygon BP_j , we calculate its centroid c_j , where $j = \{1, \dots, M\}$, with M as the total number of buildings in the data set. Once the image content of I_i is analyzed and a valid building is found, this image has to be linked to the correct corresponding ground truth. The actual filtering for valid images is further described in section 5.2. In order to link valid images to the cadastral data, we make use of the previously gathered data from the crawling process. The actual position P_i for each obtained GSV image is known. However, ground truth data is located in the Gauß-Krüger coordinate system and crawled imagery is obtained with geographic coordinates. A datum transformation between geographic coordinates and the reference coordinate systems from the national mapping agency aligns the data. Subsequently, for each P_i , we carry out a k-NN search in the ground truth data set based on the centroids c_j for each building polygon and extract k candidates $BP_{1\dots k}$. Those buildings depict our neighborhood NH_i in which the actual building Γ_i displayed in the image has to be found. To obtain the actual Γ_i , several cases have to be addressed. These are covered in more detail now.

5.1.3.1 Interiors

In the crawling process especially the random sampling approach is not limited to the required street level imagery but potentially also provides images from interior panoramas. To eliminate such data typically covering shops, public institutions and other, we take P_i and perform a point-in-polygon test for each $BP_{1\dots k}$ in NH_i . If the test returns *true* for one of the polygons, I_i contains indoor scenery and is discarded. However,

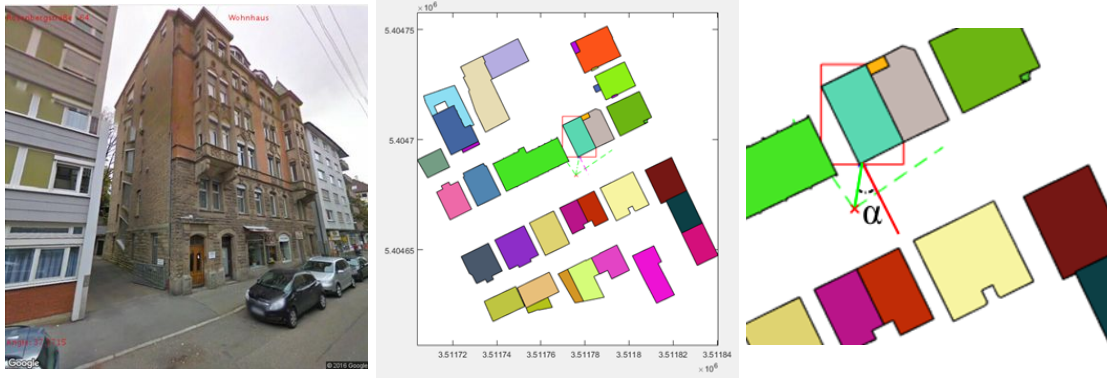


Figure 5.7: *Left*: Street view image of crawling position P_i . *Middle*: NH_i of crawling position P_i . *Right*: Viewing angle dependency. Marked with red cross: GSV acquisition position. Red rectangle: building hit. Striped green lines: field of view (FOV). The red bounding box depicts the detected Γ_i . The straight line emerging from Γ_i is the facade normal, whereas ψ_i is depicted in green. α is the enclosed angle between those lines.

too limited geolocation accuracy of these interior panoramas might lead to an actual position outside the building. In future work we have to counteract this problem since the semantic segmentation FCN is trained for outdoor scenes and hence does not provide useful information in this case. Once interiors are handled we make use of the heading information κ_i to construct a line of sight ψ_i with the corresponding predefined $FOV_{h/v}$. We limit the length of ψ_i to 20 m, to ensure Γ_i is the central content of I_i . In the next step, we determine whether ψ_i hits any of the polygons in NH_i .

5.1.3.2 Multiple Hits and Viewing Angle Dependency

To verify whether or not there exists a suitable Γ_i , we use the line of sight ψ_i and perform a test for intersection with $BP_{1...k}$. If there are intersections, we call this a hit $H_{1...h}$. However, it is possible that we obtain multiple hits. The second hit is likely to be the intersection of the same BP on its rear or side part. For multiple buildings in close proximity, there can be more than two hits. If this occurs, we simply sort $H_{1...h}$ by distance to P_i and take the candidate with the shortest Euclidean distance as our correct hit H_s . Multiple hits are more likely if the viewing angle onto Γ_i is very flat. Not only therefore we want to avoid flat viewing angles but mainly due to the reason, that we do not consider those samples as good training input. Ideally, we aim on quasi-frontal shots of the building facades. Thus, we proceed as follows. First, we determine our hit H_s and detect the edge where Γ_i is intersected. This edge is considered our facade plane. On the location H_s we construct the facade normal N_f and determine the angle α between N_f and ψ_i , representing the viewing angle (figure 5.7 on the right) α . Ideally, α would be close to zero. The viewing angle depicted in figure 5.7 on the right is still in order, however if α exceeds a certain threshold we discard this image candidate. Ideally, not only the central line of sight ψ_i should be considered. But also the bounding rays for our $FOV_{h/v}$, in cases where the hit of ψ_i might not represent the actual central building content but rather a different building polygon within the bounds of the $FOV_{h/v}$. Table 5.1 depicts crawled imagery for all five classes. The first two rows show examples we consider good, whereas the last row demonstrates some negative examples.

5.2 Building Classification

This section covers the actual task of classifying GSV imagery into different usage categories. Our approach aims on the classification of building facades in street-view images into use classes *commercial*, *hybrid*,



Table 5.1: The first two rows depict samples considered to be good, whereas the last row shows bad examples. Respective building categories are depicted on top.

residential, *specialUse*, and *underConstruction*, as introduced in this chapter earlier. For this purpose CNNs are used. As described in the previous section, we leverage Google’s region wide available Street View data. The implemented framework aims on the differentiation of five rather general use classes. These classes are purely defined by the cadastral data and therefore are not constrained to a particular visual appearance of facades. The *underConstruction* is an exception, as it is not contained in the cadastral data, this image-label set has been extracted manually. This following two sections provide these contributions: (1) A CNN based approach to classify street-level facade imagery from (semi-) automatically generated training data (cf. section 5.1). (2) Comparison of several pretrained versus non-pretrained CNN architectures. We show that pretrained models significantly improve accuracy, although the data set for pre-trained models (ImageNet) differs considerably from ours. (3) By using CAMs we get insights into important areas for the classification. Moreover, CAMs provide a visual tool to analyze predictions failure cases, thus slightly softening the “blackbox dogma” of DL. (4) As an application for our approach we show facade stipplings - importance-based image renderings that are steered by CAMs. These stippled images could facilitate the understanding of the semantic content of original images, like different building categories.

The following section 5.2.1 presents the setup of our CNN framework. This includes steps like the selection of a suitable network architecture as well as the training of the network with suitable ground truth data. Section 5.2.2 presents the performance evaluation of our end-to-end approach on different data sets with various CNNs. Section 5.3 then further examines which features are learned during the training process. For this purpose we make use of CAMs, visualizing areas being important for the network’s decision. In addition to visualizing learned features by the classification process, we use CAMs to produce abstract renderings that simplify the visual content of our input images. As described in section 5.3.4, non-photorealistic rendering can guide the viewer’s attention by varying the level of abstraction [Gooch and Gooch, 2001]. In our case

we want to focus the user’s attention to regions which are deemed important by the class-activation maps, thereby helping users to grasp the building’s category more easily.

5.2.1 CNN Training

Our investigations use state-of-the-art architectures like **VGG16**, **VGG19** [Simonyan and Zisserman, 2015], **Resnet50** [He et al., 2016] and **InceptionV3** [Szegedy et al., 2016] which have already been applied successfully for several classification tasks. Furthermore, we design networks with less amount of layers and kernels and thus a smaller number of learnable parameters. In our experiments we focus on reducing the amount of parameters, and therefore the training time, while trying to conserve the overall accuracy compared to state-of-the-art networks. The best-performing self-designed network is a **VGG**-like network that consists of 8 convolutional layers. One convolutional layer performs 1x1 convolutions in order to reduce the amount of trainable parameters. The layers of the network are: two convolutional layers (16 feature maps each), a max pooling layer with stride 2, two convolutional layers (64 feature maps each), a max pooling layer with stride 2, one 1x1 convolutional layer (16 feature maps), two convolutional layers (256 feature maps each), a max pooling layer with stride 2, one convolutional layer (1024 feature maps), a global max pooling layer, two fully connected layers with 1024 neurons each, and an output layer with $n_{classes}$ classes. All layers (except the output layer) use the ReLU activation function. The output layer uses the softmax activation function.

In total, the network consists of 8 convolutional layers, 1456 features, and contains 4.16 million parameters (for 400x400x3 sized images and neglecting the reasoning part consisting of fully connected layers). In comparison to that, the **VGG16** network consists of 13 convolutional layers, 4224 features and makes use of 14.71 million parameters (for 400x400x3 sized images and neglecting the reasoning part consisting of fully connected layers). This means our best-performing self-designed network has only 28.24% of the number of parameters of **VGG16**. **VGG19**, **Resnet50** and **InceptionV3** use even more parameters. We leverage **Keras** 2.04 [Chollet et al., 2015] for the implementation and train on a machine using a **NVIDIA Titan X Pascal GPU** (12GB).

Obviously, in the real world there are more facades of residential buildings than facades belonging to other classes causing an imbalanced data set, initially. With the help of a priori data augmentation (horizontal flipping, warping, cropping, jittering and modification of saturation) we create an equally distributed data set with the same amount of samples for every class. Our final training set is a balanced data set consisting of 75.000 labeled images in total (15.000 images per class). Validation set and test set consist of 350 labeled images each (70 images per class). The comparatively small size of the test and validation set is due to the data set split before data augmentation. Data augmentation is only applied to the training set in order to prevent overfitting. Augmenting validation set and test set would bias the achieved results in a positive way. In addition to this multiclass (5-class) data set, we provide data sets with reduced amount of classes. The binary data sets *residential* vs. *nonResidential* and *underConstruction* vs. *notUnderConstruction* as well as a 3-class data set with classes *commercial*, *hybrid* and *residential* are used to investigate the impact of class definitions to the classification performance. All provided data sets are split into three parts: images for training (training set), images for validating the network’s performance during the training process (validation set) and images for evaluating the network’s performance after the training process (test set). As noted in section 4.2, training a CNN is an optimization task in which the goal is to minimize a cost or loss function. The loss function used is cross-entropy, described in equation (4.6). Parameters are initialized using Glorot uniform initialization [Glorot and Bengio, 2010], while updating is based on backpropagation in combination with mini-batch SGD. Generally, the cross-entropy decreases during training when evaluated on training data. Naturally, this comprises overfitting which means that noise of the training data is learned. To avoid overfitting, we use validation data during training to keep track on the generalization gap - the difference between performance on training data and validation data. Moreover, we use regularization methods like early stopping, weight decay and dropout in order to reduce overfitting. During training, parameters are

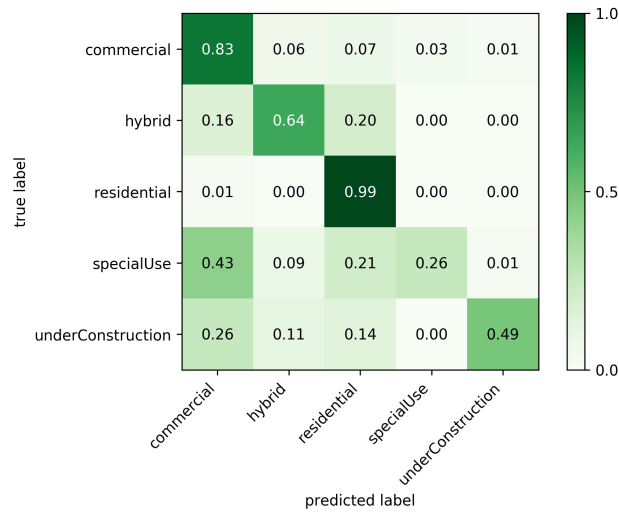


Figure 5.8: Normalized confusion matrix for the best-performing architecture on the 5-class test set, an **InceptionV3** network pre-trained on ImageNet.

adjusted so that the desired task, i.e., the classification of building facades in Google Street View images into the respective use classes is accomplished. After training, a network’s performance is analyzed by testing images with known ground truth data that have not been used during training (see section 5.2.2).

5.2.2 Classification Results

Amongst the investigated networks, an **InceptionV3** network pretrained on ImageNet performs best with an overall accuracy of 64.00 % when being evaluated on our 5-class test set. Its colored normalized confusion matrix (NCM) is shown in figure 5.8. The diagonal elements represent per-class accuracies (in case of NCM, these equal per-class recall values). Per-class precision values can be calculated by dividing the diagonal elements by corresponding column sums. The best performing class (*residential*) achieves an accuracy of 98.57 %, which is encouraging since the majority of real-world buildings are residential buildings. On the other hand, classes *specialUse* and *underConstruction* only achieve accuracies of 25.71 % and 48.57 %, respectively. This is due to the classes’ definitions (see the introduction of chapter 5 and table 5.1). Class *specialUse* has high intra-class variance and therefore bad separability from other classes. The *underConstruction* class essentially is a mixture of all other classes. Frequently, an image of class *underConstruction* differs only from other classes by a scaffold. Normally, scaffolds are of small size in the image and hard to detect by a computer. In contrast, the per-class precision of those two classes is high (90.00 % and 94.44 % respectively). Additionally, table 5.3 compares the classification results of the two best performing pretrained networks. Both network performances exhibit similar characteristics throughout the different classes. Other current

Network	Overall Accuracy [%]
VGG16	63.43
VGG19	63.14
ResNet50	55.43
InceptionV3	64.00

Table 5.2: Highest achieved overall accuracies of different CNNs on the balanced 5-class test set. All networks have been pretrained on ImageNet.

state-of-the-art models (VGG16, VGG19, Res-Net50) perform slightly worse (cf. table 5.2). For pretraining we used ImageNet images. We observed that pretraining improved the performance by 8-10 % although ImageNet images and images of our data sets differ significantly. Our best-performing self-designed network could achieve 52.29 % overall accuracy while reducing the amount of parameters by approximately 72 % (relative to VGG16, see section 5.2.1). The self-designed network was purely trained on our 5-class training set. The achieved overall accuracy is slightly worse than the overall accuracies of not-pretrained VGG-networks (not-pretrained VGG16: 55.43 %, not-pretrained VGG19: 53.43 %). Training time per epoch reduces to $26.3 \frac{\text{min}}{\text{epoch}}$ (VGG16: $54.5 \frac{\text{min}}{\text{epoch}}$, VGG19: $62.5 \frac{\text{min}}{\text{epoch}}$).

Class	VGG16 based		InceptionV3 based	
	Precision	Recall	Precision	Recall
<i>residential</i>	0.626263	0.885714	0.610619	0.985714
<i>hybrid</i>	0.578313	0.685714	0.714286	0.642857
<i>commercial</i>	0.514563	0.757143	0.491525	0.828571
<i>specialUse</i>	0.869565	0.285714	0.900000	0.257143
<i>underConstruction</i>	0.928571	0.557143	0.944444	0.485714
average / total	0.703455	0.634286	0.732175	0.640000

Table 5.3: Precision and recall for the two best multiclass classification networks in terms of overall accuracy. Both networks, VGG16, as well as InceptionV3 based architectures, are pretrained on ImageNet and finetuned on our generated data set.

In addition to our performance analysis for five classes we further used 2-class and 3-class data sets to analyze effects of class separability and intra-class variance on the classification performance. Correspondingly, we train binary and 3-class classifiers. Results of our 2-class and 3-class classifiers prove that better class separability and reduced intra-class variance have a crucial impact on the achieved overall accuracy. The binary classifiers (*residential* vs. *noResidential* and *underConstruction* vs. *notUnderConstruction*) achieve an overall accuracy of approximately 88 %. Our 3-class classifier (*residential* vs. *hybrid* vs. *commercial*) achieves an overall accuracy of approximately 73 %. Table 5.4 lists the classification report accordingly. We

Class	Precision	Recall
<i>residential</i>	0.681319	0.885714
<i>hybrid</i>	0.703704	0.542857
<i>commercial</i>	0.830769	0.771429
average / total	0.738597	0.733333

Table 5.4: Precision and recall for the VGG16-based 3-class classifier.

will refer to this 3-class classifier again in section 5.3.2.

An in-depth discussion of different architectures, parameter, and data set configurations can be found in [Laupheimer, 2016]. This involves investigations of different data set compositions, different ways of using pretrained models, and own architecture implementations as mentioned above. The results presented here are the core insights. In summary, using pretrained networks generates superior results than training own network architectures from scratch. Additionally, finetuning all layers from pretrained networks generally produces better results than using those architectures as fixed features extractors, where only the last layer is replaced to produce an output that is equal to the number of classes defined for the task at hand (five classes for the multiclass case). These observations seem to be justifiable since our self-generated data set differs significantly from ImageNet.

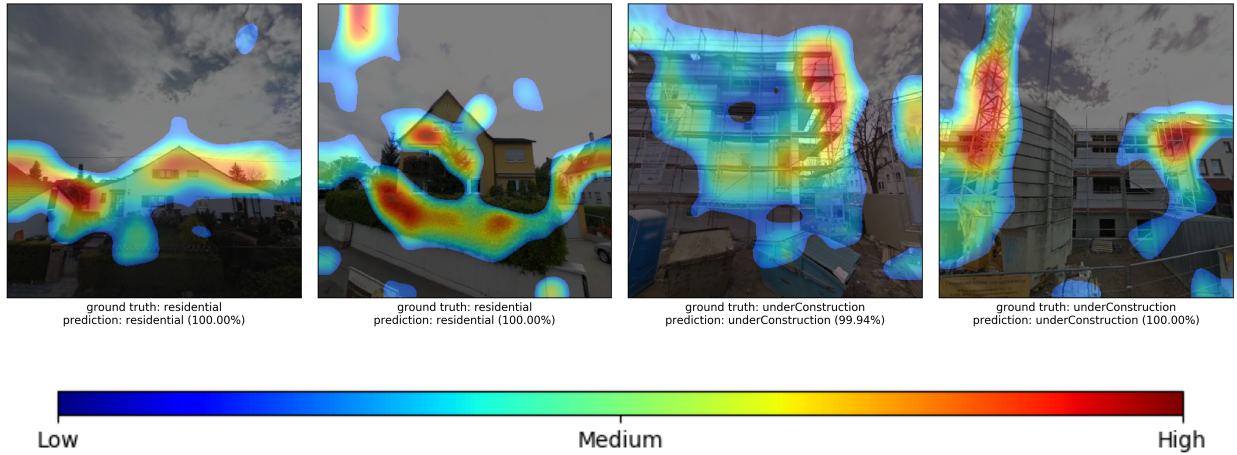


Figure 5.9: Correctly predicted examples for binary classification, overlaid with CAMs. The colorbar in the second row depicts the importance for a certain region that triggered the network's decision.

5.3 CNN-Learned Features

In addition to the numerical classification results presented in the previous section, we want to make further evaluations. CNNs can provide class-activation maps as additional output, which are a powerful tool to interpret and localize learned features within input images. Hence, we elaborate on CAMs in the following section 5.3.1. Subsequently, we use the trained 3-class network in order to investigate its transferability to different representation forms (section 5.3.2). In order to do so, we pick images from our user study in section 3.2.1 and feed them to the network. Section 5.3.3 deals with the influence of manipulated imagery on the network's decision. Additionally, we briefly discuss the results on sketch-like images of different buildings. In section 5.3.4, we present an application for CAMs - they can be used to steer stippings. Stippings were briefly introduced in section 5.2 and will be further discussed in section 5.3.4. CAMs can be used as importance weights for the stippling process. Thus, non-photorealistic renderings that guide the user's focus to important areas in the image can be generated.

5.3.1 Class-Activation Maps for Prediction Interpretability

CAMs are heatmaps that highlight image sections, which are discriminant for the respective classification process. Thereof, a human operator can derive learned features which are useful for understanding the network's decisions. These insights could be used to improve the classification in future work. So-called Grad-CAMs represent the gradient of one specific class with respect to the activation maps of the last convolutional layer of a trained CNN [Selvaraju et al., 2017]. Each activation map shows the presence of a specific feature. Hence, gradients reflect the importance of a specific feature regarding the network's decision for that class. While fully connected layers will discard the spatial information on the presence of a high-level feature, this information is still available in each activation map of the last convolutional layer. In this way, CAMs conserve the spatial information, too. We use the implementation of the Keras Visualization Toolkit [Kotikalapudi, 2017]. For better visualization and interpretability, CAMs are upsampled by means of bicubic spline interpolation and overlaid to their corresponding RGB input images. Upsampling is required since the CNN pipeline convolves and pools the original image several times, which reduces the initial spatial dimension. Figure 5.9 shows correctly classified images for the binary classification overlaid by the associated upsampled CAMs which highlight decisive image parts. Please note the importance of the vegetation in the

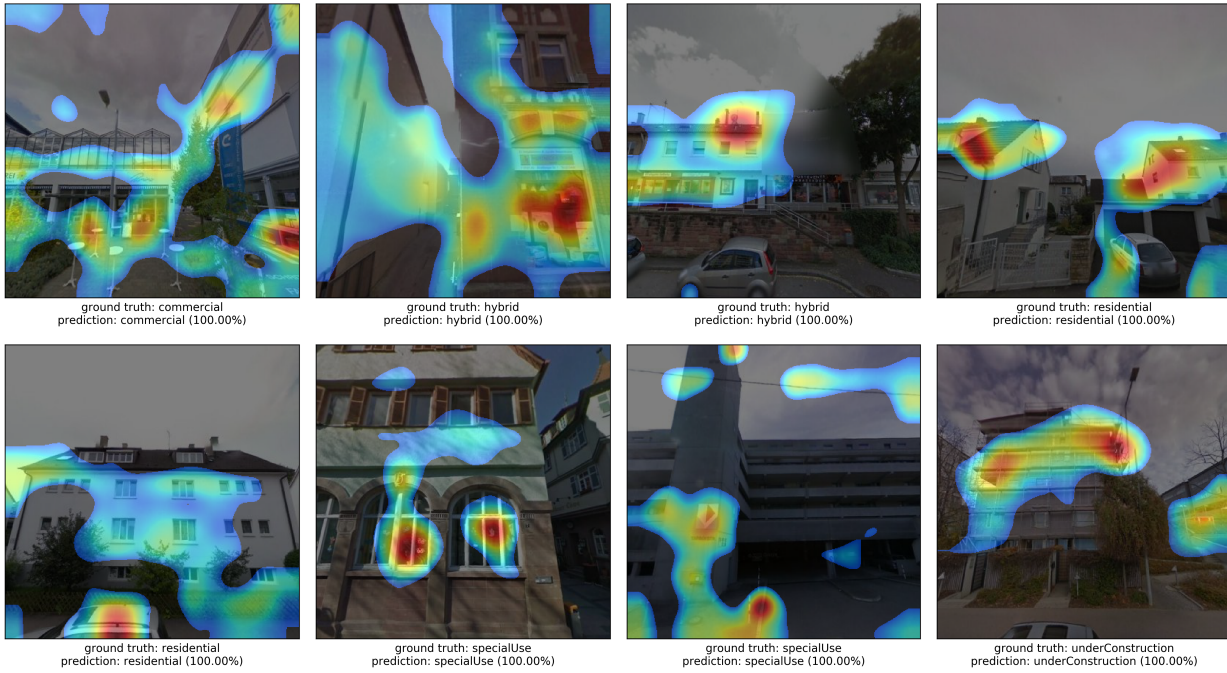


Figure 5.10: Correctly predicted examples for multiclass classification, overlaid with CAMs. The colorbar is according to figure 5.9.

second from left image for a correct prediction. Figure 5.10 shows correctly classified images for the 5-class classification overlaid by the associated upsampled CAMs. The second image from left in the second row shows a kindergarten. Please note that the big windows with the window decorations attract the network’s attention. Similarly, there is a high activation visible for cars in front of residential buildings, which semantically makes sense (last image from left in the first row and first image in the second row). Images overlaid with CAMs give insight about the features learned for decision making. Similar to features a human operator would use instinctively, a CNN uses ‘facade-related’ information like rooftops, windows and chimneys as well as contextual information like trees, cars and sky. CAMs give a hint on image areas that are crucial for image labeling. This can be used to analyze potential misclassifications. As it is visible from the depicted examples in figure 5.11, misclassifications frequently result from features, which are not detected by the CNN or from features, which are misinterpreted. The restaurant with accommodation on the left side in the first row in figure 5.11 is classified as *specialUse* due to its church-alike entrance. Besides, dormers are not detected. The church on the third from left in the second row is classified as *residential* as the bell tower might be mistaken as a chimney. Another church in the second from left image in the first row is misclassified as *residential*, since the network picked up on the person on the sidewalk and the surrounding bushes. The lower left image shows a *residential* building with many windows and satellite dishes. As the satellite dishes are not detected, the building resembles an office building (*commercial*). The second from left image in the second row shows a *residential* image with traffic signs in the front. Certainly, they are mistaken as advertisement panels leading to the prediction *hybrid*.

As it is also visible in the last column of figure 5.11, the choice of ground truth labels has an influence on the classification performance. From a human point of view the building in the image center is decisive for labeling. In contrast, CNNs simply decide by using those features with the most impact on predictions. The location and size within the image do not matter. Hence, predicted and ground truth labels can differ, although basically, both are correct. Such predictions still are lowering the overall accuracy if they do not match the human-given label. We refer to this type as “false” predictions as they are not necessarily false. The

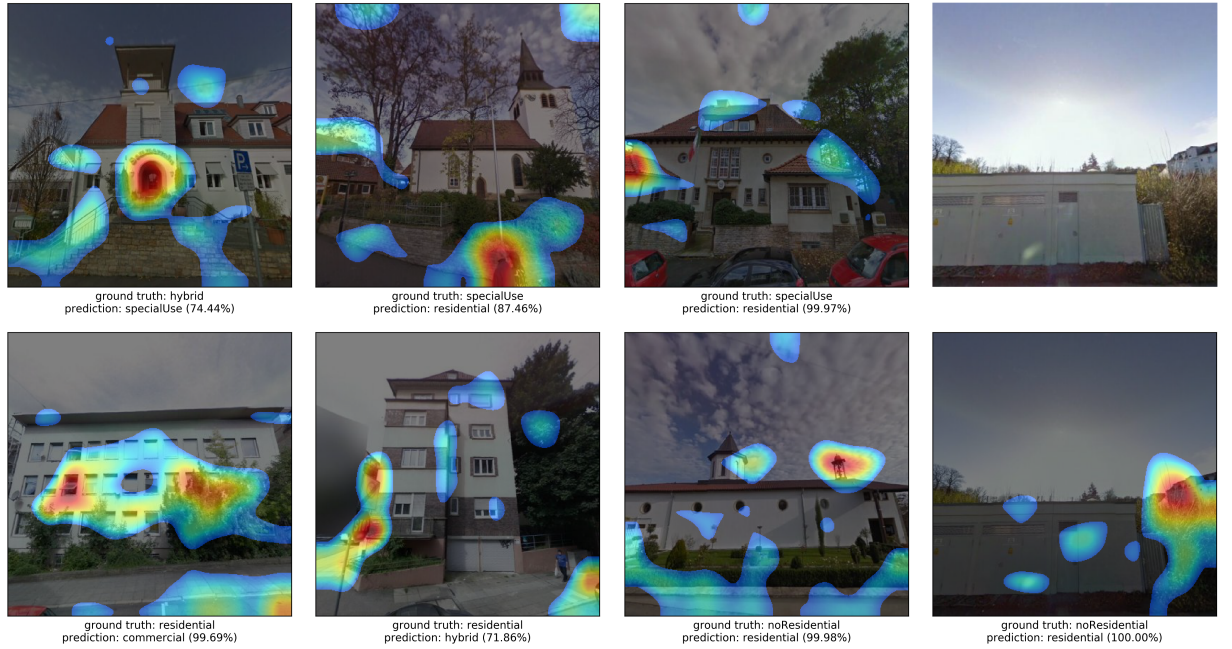


Figure 5.11: Misclassifications. The last column is an example of a “false” prediction. Original image (*top*) and original image overlaid by its upscaled CAM (*bottom*).

last column of figure 5.11 shows such a “false” prediction where the residential building and the vegetation on the right hand side in the image cause the prediction *residential*. On the contrary, the transformer house in the image center is decisive for the human-given label *noResidential* (binary classification). Such a problem can be salvaged by using regional CNNs with instance-aware classifications.

5.3.2 Evaluation on Different Representation Types

In section 3.2 humans had to classify building categories based on different representations of virtual building models. As discussed in the previous sections, the CNNs were trained on GSV imagery. In order to examine the transferability and generalization capability, this section investigates the influence of different representation types on prediction results. Since building categories were defined slightly different for the user study as compared to the DL based classification, a class remapping has to be defined. However, this is straight-forward and depicted in table 5.5. Since these initial classes get remapped to only three target

<i>residential</i>	<i>hybrid</i>	<i>commercial</i>
<i>one-family building, multi-family building, residential tower</i>	<i>building with shops</i>	<i>office building, industrial facility</i>

Table 5.5: Remapping of classes used in user study (*bottom row*, cf. section 3.2) to fit definitions used in this chapter (*top row*).

classes, the 3-class classifier described in section 5.2.2 is used for predictions. Table 5.6 gives classification results for all representations and building categories. However, it has to be noted, that due to the low number of samples, no statistically sound statements can be made. The data can nevertheless be used to get an impression of the influence of different representations on the classifier performance. Masked GSV imagery representations exhibit the lowest error rates which is what would be expected as it resembles the original training data the closest. The untextured LOD3 models have an error rate of almost 50 %. This representation type is furthest away from the training data. Basically only building edges are depicted.

	OFB	MFB	RT	BWS	OFF	IF	Total
Untextured LOD3 BRep	0.87 (13/15)	0.59 (10/17)	0.60 (9/15)	0.33 (5/15)	0.13 (2/15)	0.13 (5/15)	0.48 (44/92)
Textured Mesh from GE	0.00 (0/0)	0.50 (2/4)	0.80 (8/10)	0.00 (0/6)	0.00 (0/6)	0.33 (2/6)	0.38 (12/32)
Masked GSV imagery	0.00 (0/5)	0.21 (3/14)	0.83 (5/6)	0.00 (0/10)	0.16 (1/6)	0.00 (0/3)	0.20 (9/44)
Total Error Rate	0.65 (13/20)	0.43 (15/35)	0.71 (22/31)	0.16 (5/31)	0.11 (3/27)	0.29 (7/24)	0.39 (65/168)

Table 5.6: Relative error rates for different representations of the six building categories introduced in the user study. Numbers in brackets denote the fraction of erroneously classified images.

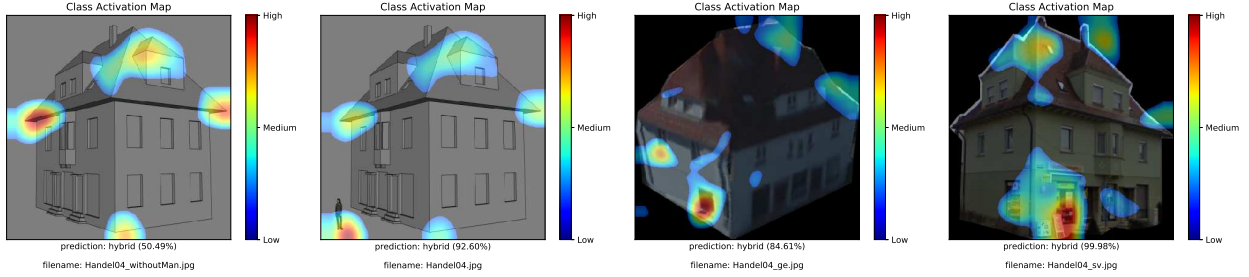


Figure 5.12: Comparison of predictions with overlaid CAM for three different representations of the same building. The ground truth label is *hybrid*. From left to right: Untextured LOD3 model without and with stylized human, respectively, GE textured mesh, and GSV image. Please note, how the addition of the stylized human in the LOD3 presentation drastically boosts the prediction confidence.

Hence, the result is justifiable as well. Interestingly, the *residential* samples error rates are higher than those for *commercial* and *hybrid*, mostly throughout all three representations. This contradicts the results of the user study in section 3.2. Worst results are for masked Street View images of *residential towers*. It can be concluded that environmental objects surrounding the building of interest take an integral part for the classifier decision. This holds especially true for residential buildings. Some examples of correct predictions for the different representation forms with their CAMs overlaid are depicted in table 5.7. Noteworthy is the very high activation for the chimneys in the textured mesh representation of a fabric, hence being correctly classified as *commercial* building. Both, the LOD3 model, and the textured mesh of the *residential* category exhibit high activations on the windows. Figure 5.12 depicts three different representation types of the same *hybrid* building with two variants of the untextured LOD3 model. The textured mesh and masked GSV representation are both correctly classified with high confidence. However, the untextured LOD3 model highly relies on the stylized human in the left front of the building as decisive feature. Without the human, the prediction confidence is only slightly over 50 % with the highest activation on the roof ridge, whereas adding the human boosts the prediction confidence to over 92 % and shifts the highest activation accordingly to the human.

In addition to the above-shown representations, we briefly examined the VGG16 based (multiclass) network when being presented with pictograms. No classifications metrics are presented for this task, the focus is on visual impressions. Some predictions results with CAM overlays are shown in figure 5.13. Please note, the highest activation for the (a) *residential* example (first image) in the tree, (b) *commercial* example of a sketched factory (third image) on the chimneys, (c) *specialUse* example of a church (fourth image) on the clock in the tower, the entrance and the arched windows, and for the (d) *underConstruction* example on the scaffold and crane. The listed activating areas nicely align with those found in the original GSV imagery. However, there is also a large amount of prediction runs and pictogram examples where the CNN classification failed. DL architectures simply are not yet capable of semantic reasoning the same way humans do. Hence, they have to rely on low to high level features. For the pictograms shown in figure 5.13 this worked surprisingly well.

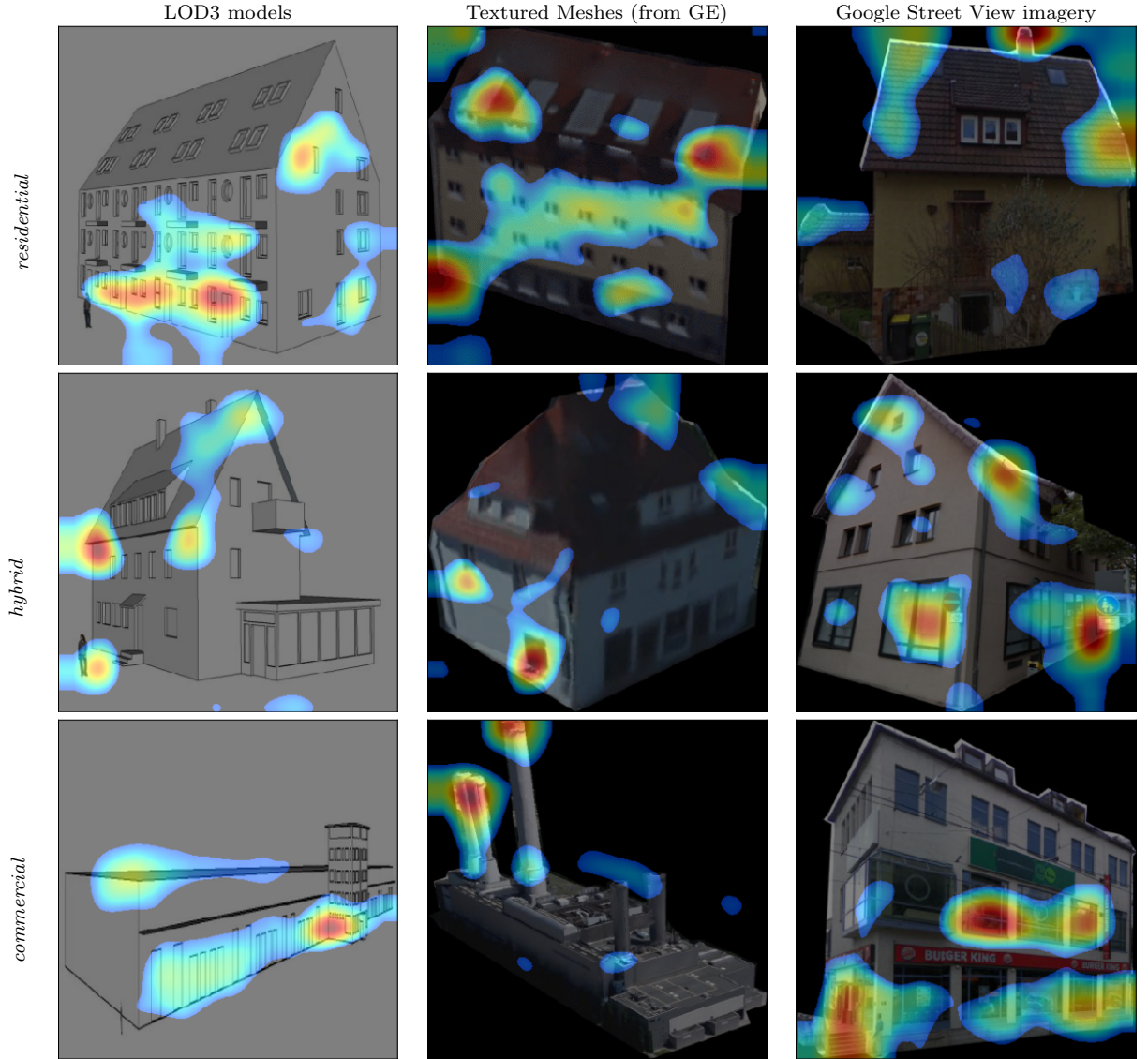


Table 5.7: Correctly predicted snapshots of different representation forms with their CAMs overlaid. The underlying CNN has a 3-class output.

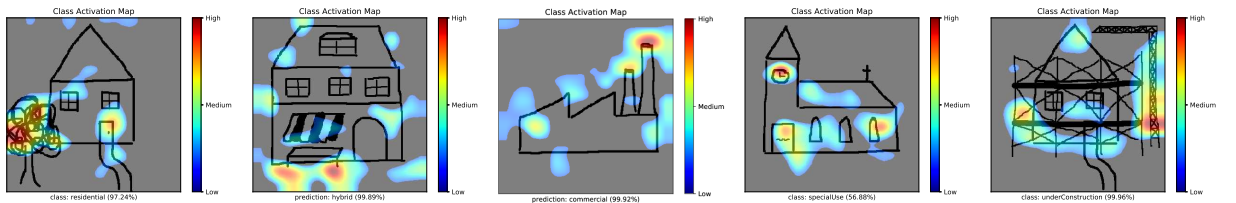


Figure 5.13: Predictions with overlaid CAMs for pictograms of ground truth classes *residential*, *hybrid*, *commercial*, *specialUse*, *underConstruction*, respectively. Some of the activating features nicely align with those found in the actual GSV imagery predictions.

5.3.3 Influence of Manipulated Images

Adversarial attacks and how to make networks robust against them is currently a hot topic in DL [Szegedy et al., 2013, Goodfellow et al., 2014b, Kurakin et al., 2016, Madry et al., 2017]. For image classifications, such attacks work in a way, that the input to a CNN is manipulated and thus inducing wrong classification results. Moreover, the adversarial attack can happen by adding noise to the image, which accounts for such small perturbations that the change of the input is not perceivable for humans. To give a concrete example, a CNN is presented with an image of a cat. Initially, the CNN correctly predicts the label ‘cat’ with a confidence of 99 %. After adding crafted noise to the cat image, the image is fed through the network again. This time, however, the CNN predicts ‘dog’ with a confidence score of, e.g., 79 %. For humans, the manipulated image is not distinguishable from the original one, we still correctly recognize the cat, but the CNN gets fooled into predicting a dog. [Goodfellow et al., 2014b] proposed the *fast gradient sign method* to generate such adversarial examples. Prerequisite is a trained network with fixed weights. The gist of this approach (as well as multiple other adversarial attack approaches) is to calculate the gradient of the loss with respect to the input image, rather than the model weights (as would be the case for traditional CNN training). The magnitude of the gradient is not important, only its direction/sign. In order to generate an adversarial example, each pixel of the input image gets altered by a small amount of noise ϵ . The alteration, that is, either subtracting or adding ϵ to the input image pixels, depends on whether the gradient sign is positive or negative. Pixels are altered in the direction of the gradient. Hence the name *fast gradient sign method*. Instead of minimizing the loss by optimizing model parameters as usual in CNN training, an adversarial attack decreases the loss by adding crafted noise to the input. Similarly, some approaches introduced small patches to insert into images and fooling the classifier that way [Brown et al., 2017].

Inspired by these works, we investigated the impact of manipulated images to our trained VGG16 network (cf. table 5.2). However, we do not specifically generate adversarial examples in the way outlined above but rather check the robustness of the trained network against manual modifications of input images. These manipulations are not aimed at high visual fidelity. They merely introduce visual features that were previously identified as being important for certain class predictions.

Table 5.8 depicts an example of the best performing class *residential* and the second worst performing class *underConstruction*, respectively. Rows 1 and 3 show the original image along with several manipulations, whereas row 2 and 4 depict respective prediction results with CAMs overlaid. The network correctly predicted the classes for the original images with a confidence of 100 %. Adding an advertisement panel does not have much influence on the prediction results since the surrounding vegetation has a substantial impact on the decision. However, stacking several floors and adding large advertisement panels (row 1, column 3) leads to a prediction of the class *commercial*. Highest activations can be found on the advertisements in the center of the facade. Introducing a ‘night mode’ does not affect the correct prediction. The area of activation, however, changes from the facade itself (row 2, column 1) to the facade edges (row 2, column 4), probably due to the facade being hardly visible anymore. Replacing the top scaffold in table 5.8 (row 3, column 2) with the advertisement panel of a pharmacy, fools the classifier into predicting a *commercial* building. Previous high activations of the crane in the top center of the images are suppressed, while the pharmacy sign on the very right top of the image at the same time causes high activations. The *commercial* prediction result is amplified by removing the crane and adding vegetation in the foreground (row 4, column 3). Additionally, random RGB noise manipulations were investigated (row 3, column 4). Though this does not have any impact on correct predictions, and the respective CAM barely changed (row 4, column 4). In conclusion, the experiments in this section have shown, that visual features previously identified as decisive for specific building categories can be used to manipulate input images in order to fool a network. On the other hand, the used CNN proved to be robust against random RGB noise and night mode alterations.


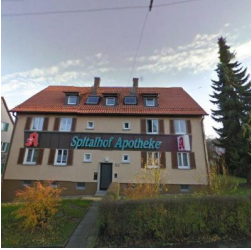
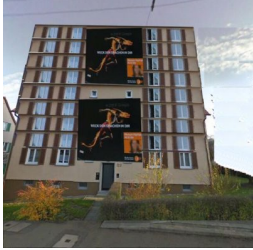
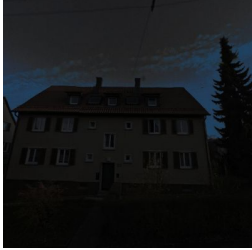
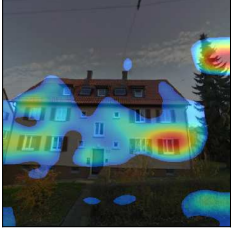

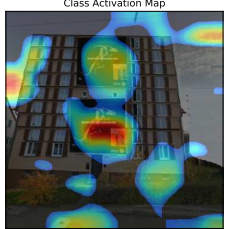
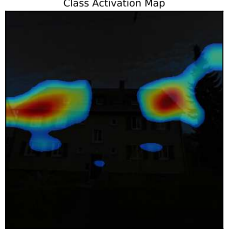




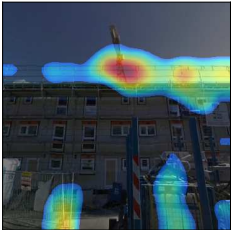
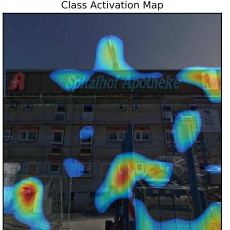
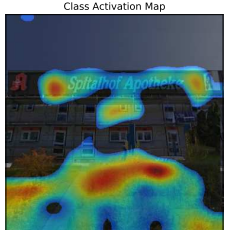
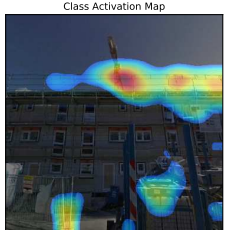
Original		Manipulations	
			
Class Activation Map  prediction: residential (100.00%) filename: 37617_3511640.0864_5410335.6828.jpg	Class Activation Map  prediction: residential (99.94%) filename: 37617_3511640.0864_5410335.6828_modified.jpg	Class Activation Map  prediction: commercial (65.27%) filename: 37617_3511640.0864_5410335.6828_modified5.jpg	Class Activation Map  prediction: residential (99.36%) filename: 37617_3511640.0864_5410335.6828_modified9.jpg
			
Class Activation Map  prediction: underConstruction (100.00%) filename: 14941_3515892.3037_5408692.5921.jpg	Class Activation Map  prediction: commercial (96.87%) filename: 14941_3515892.3037_5408692.5921_modified.jpg	Class Activation Map  prediction: commercial (99.99%) filename: 14941_3515892.3037_5408692.5921_modified5.jpg	Class Activation Map  prediction: underConstruction (100.00%) filename: 14941_3515892.3037_5408692.5921_modified7.jpg

Table 5.8: Influence of manipulated images. Row 1 and 3 depict original images along with manipulated versions thereof. Row 2 and 4 show respective CAMs. The very last column in rows 3 and 4 uses RGB noise manipulation, hence no modification is visually perceivable.

5.3.4 Importance-Based Abstract Renderings

Important features for humans in order to visually comprehend different building categories were derived in section 3.2.1. These features were used to generate more abstract visual representations of the initial 3D models in section 3.2.2.7 (and section 3.3.2). By emphasizing essential structures and simultaneously eliminating unimportant parts, the building models become more easily understandable for a human user. Similarly, CAMs provide importance maps for building categorization by machine vision in image space. They can be used to steer the generation of non-photorealistic renderings. This process should again maintain a high level of detail for essential regions, while less important ones are abstracted. Thus, a CAM-based abstraction using the provided importance maps can help to focus a human viewer’s attention to key regions for the visual discrimination of building categories. We are aware of the fact that CAMs sometimes strongly focus on specific small scale details. Hence, human attention might be distracted in those cases.

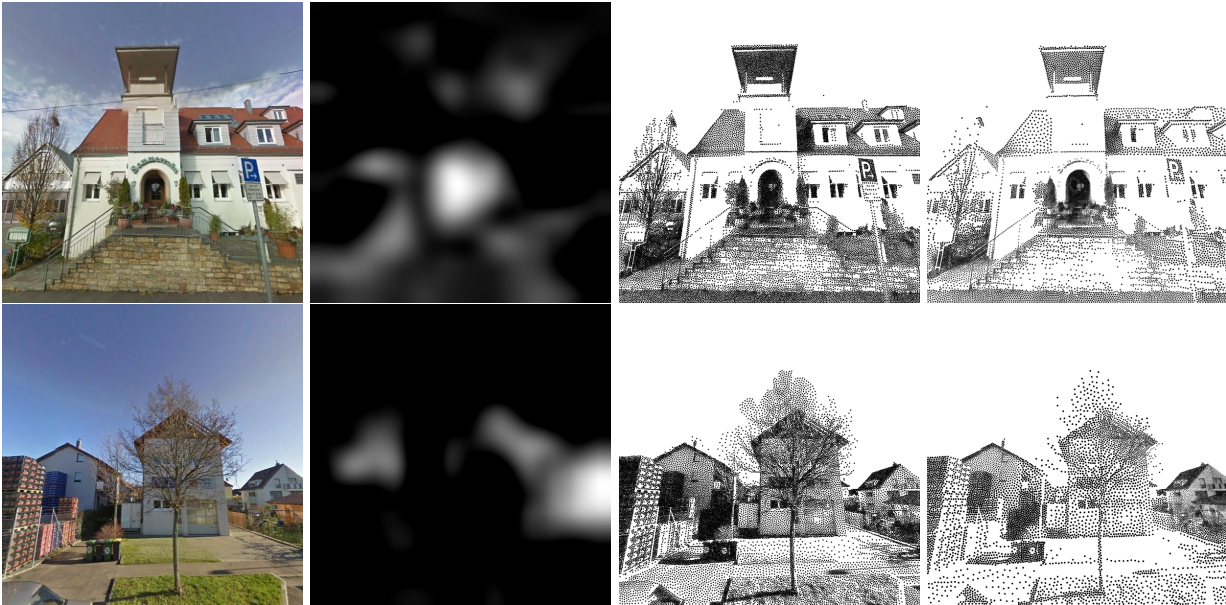


Figure 5.14: Comparison of stipplings without (third from left image) and with (right-most image) CAM-steering to focus the user’s attention. *Top row*: the focus is set on the door area. *Bottom row*: focus on the building in the back right.

Our abstract rendering bases on stippling, which is frequently used in architecture for sketching and illustration purposes. For matching the tone and texture of an image, visually similar point sets are created: in dark areas, many points are used and, conversely, only few points are stippled in bright regions. An increasing degree of abstraction is achieved by reducing the amount of points in the less important regions (provided by CAMs) while increasing the point size. In this way, smaller details are removed progressively. To summarize, the amount of distributed points and their respective size depend on the tone, texture, and the local importance. We use the stippling algorithm of [Deussen et al., 2017] due to its ability to locally vary the degree of abstraction while avoiding visual artifacts such as distracting regularities within the point sets [Deussen et al., 2000, Secord and Adrian, 2002].

The algorithm uses the original image and its corresponding gray-scale CAM as input. First, the contrast and brightness of the input image is increased to make different point densities visually more dissimilar and to reduce the overall number of points in the final representation. Our stippled results aim for detailed visualization in important areas and a high degree of abstraction in regions with low importance. This is achieved by not only varying the amount of used points but also their size. In order to preserve details,

many small points are used in important regions. In contrast, few large points visualize less important regions. This removes extraneous details and thus provides a more abstract representation. The respective point size is derived from the CAMs. Hence, it is directly correlated to the local importance. The algorithm of [Deussen et al., 2017] dynamically distributes points with given sizes to match the local tonal value of the input image. Using gray-scale CAMs allows a direct linear mapping from importance $[0;1]$ to point size $[\min;\max]$. In the original implementation, dark but unimportant image regions would provide large points to represent the local image tone. Since this would distract the viewer’s attention from regions to be emphasized, the density of points in these regions is further reduced. To realize this thinning, our modified algorithm brightens less important regions. Except from that, our experiments use parameters suggested by the original authors for hysteresis, supersampling, and the number of iterations. In figure 5.14 we compare stippled results of constant point size and without brightening to CAM-steered results created with our pipeline. The area in focus (bright regions in the gray-scale CAMs) is clearly distinguishable from the rest in our result. In figure 5.15 we show input images, their gray-scale CAMs, and the final stippled results side-

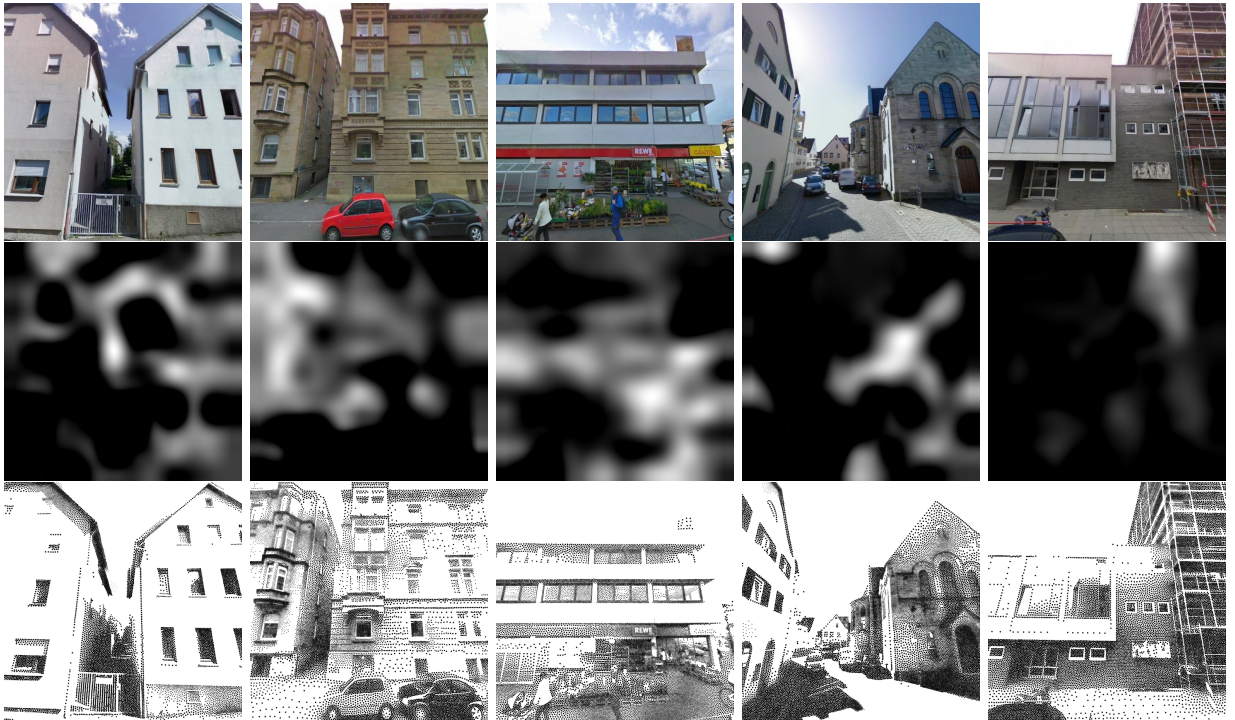


Figure 5.15: Rows showing the comparison of input images, their gray-scale CAMs and resulting stipplings, respectively. From left to right: *residential*, *hybrid*, *commercial*, *specialUse*, *underConstruction*

by-side for comparison. Areas with high importance are represented with many small points and therefore have a high amount of detail, while less important regions are represented by only few points and therefore few details. At the same time, the relative tonal values from the input image are represented faithfully: dark areas remain dark in the stippled result (e.g., shadows of houses) and vice versa.

5.4 Summary for the Image-Based Interpretation of Building Facades

In section 5.1 we successfully linked Google Street View imagery to a database that contains semantic information for every contained building polygon. Such databases are available for a number of cities.

Hence, it is possible to generate large amounts of training data, which is a prerequisite for the successful application of Deep Learning frameworks for classification. Some additional work has to be done in the pre-processing step. Indoor scenes with limited geo-location accuracy have to be detected and eliminated. In our investigations, we found that semantic data provided by the city administration can be ambiguous or even erroneous. This label noise is an issue, which at the same time shows the necessity of the proposed approach of automatic building use classification. For now, clearly wrong or ambiguous samples were discarded in an interactive post-processing step to provide a reasonable training input.

Classifying street-level images of building facades into different utility classes is a highly complex real-world task struggling with various problems depending on image properties, object properties and environmental factors (variation of object scale and orientation, changing amount of buildings per image, cropped facades in images, occlusions, or changing illumination). The proposed end-to-end approach for classifying street-level images of building facades on a high abstraction level achieves promising results for the automatic semantic enrichment of 3D urban models. Future work could be based on training a variety of architectural styles, as well as performing the training phase in one city and testing in a different one to investigate transferability. One application for our approach is the area-wide enrichment of crowd-sourced data like OSM building polygons. Since a further diversification from the current classes is desirable, it is conceivable to leverage the original building-related segmentation classes from the FCN (*awning, balcony, door, window*) as a meta-classifier. Moreover, the FCN used for image analyses could be replaced by an object detector framework like **Faster R-CNN** or **Mask R-CNN** [He et al., 2017] since we are ultimately only interested in the bounding boxes of buildings. However, pre-trained models do not contain a building class yet. Therefore, such a network has to be trained from scratch. We depicted a first feasibility test for this task in section 5.1.2.3, where we trained **Faster R-CNN** on three classes that are essential for our task: *vegetation, vehicles, and buildings*. With the rapid development of DL-based object detectors, we see this as the future of semantic enrichment of urban scenes. The evolution from **Faster R-CNN** over **Mask R-CNN** to panoptic feature pyramid networks [Kirillov et al., 2019] is a perfect example and has shown tremendous advances in object detection, instance segmentation, as well as semantic segmentation. Training an instance-level segmentation approach with fine-grained building use classes could be a promising way to enhance the understanding of urban building sceneries. Dense pixel-level segmentation maps could then be projected to point clouds and meshes to further semantically enhance these 3D visualizations. In essence, our approach lined out in chapter 4 might become a feedback loop with the building classification task depicted in this chapter.

In order to form a reasonable judgment of the CNNs' performance, a user study on our data set could be conducted to derive human performance for comparison. The need for such a study is evident since the appearance of a facade is not necessarily in accordance with the actual usage of the building. For example, there can be a medical practice in a former purely residential building without a constructional change of the facade. While there is no change in its appearance, the house type will change from class *residential* to *hybrid*. We assume this discrepancy cannot be solved within the visual space. Hence, it is a natural limit for the achievable overall accuracy (both for human and machine performance). Nevertheless, the error rate of 36 % misclassified images (5-class case) reflects the need for further improvement.

With the help of CAMs, the reasons for false predictions could be determined. Main error contributors are "false" predictions, that is, buildings that were actually correctly predicted but are located in the background or on the side of an image and hence do not represent the current building of interest. The second error source is misclassifications due to misinterpreted features or not detected features. Additionally, the influence of different representation types, as well as manipulated images have been investigated. CAMs show decisive features for the classification results on different representations. These regions are often in accordance with the examples presented in section 5.3.1. An interesting takeaway from the CAM inspections is the importance of surrounding environmental objects (mostly vegetation), especially to identify *residential* buildings. Results of the image manipulations have shown that CNNs are robust against random noise and night mode manipulations. More intrusive manipulations such as adding important features of certain categories (like advertisement panels) might influence the network predictions. Recently, [Mitsuhara et al.,

2019] introduced an attention branch network for human-in-the-loop interaction on CAMs. Users are able to modify a CNN-generated heatmap and steer the attention to areas considered more important by the operator. This mechanism is especially useful when multiple objects are present within an image and might provide a way to improve building use classification as well.

The results of our binary and 3-class classifiers show that the type and number of classes are critical to attainable performance. One option to find a reasonable amount of classes and sensible class definitions would be unsupervised learning methods. This will improve class separability and reduce intra-class variance. A subtle side effect of better class separability and reduced intra-class variance is the increasing quality of CAMs as more features will be class-specific and therefore more discriminative. Accordingly, better importance maps can be provided for the presented stippling algorithm (cf. section 5.3.4) resulting in even more appealing stippling-based abstract renderings which facilitate the comprehension of semantic image content for human users.

Chapter 6

Conclusion and Outlook

In this thesis, we presented several integral parts contributing to a holistic pipeline for semantically interpretable virtual city models. We briefly conclude the core subjects of this work and then give an outlook on where those domains might be headed in the near future.

A semi-automatic building reconstruction algorithm is used to obtain building models from point clouds. This algorithm is based on previous work and extends it in several ways, making it more generic, flexible, and robust. It is more generic by means of using standardized data input and output formats, allowing to directly export georeferenced, enhanced building models. Due to a GUI, users can interactively select facades of interest and additionally adapt hyperparameters, thus making the pipeline more flexible. Robustness is increased by proposing the radiometric segmentation pipeline as support for distinction between *facade points* and *non-facade points* when operating on point clouds with limited geometric accuracy. The subsequent grammar-based modeling generates an enriched version of the input building model. We implemented a CityGML-compliant pipeline. Hence, the final model can be exported to the CityGML data format and thus explicitly capsules semantic information. This combination of a bottom-up, followed by a top-down approach is capable of extracting facade semantics and consecutively using this knowledge to generate new geometry on remaining facades.

In order to get a deeper understanding of human perception of buildings, we conducted user studies. The knowledge derived thereof can be used for a perception-based abstraction algorithm. This approach simplifies building models but retains features that are important for people to be able to assign the building to a certain category. Additionally, we proposed a pipeline for the enhancement of virtual building models. Above-mentioned CityGML models are parsed to extract key properties of a building. Through a feature space embedding, unknown buildings can be mapped to a certain building category. Once a use type is assigned, building category-specific rule sets can be used to enhance the appearance of a building. Similarly, coarse models can be enriched by the application of category-specific rules. One way to investigate the impact of different building representation types is the use of a VR environment. Therefore we developed a framework to showcase different representation forms of virtual city models in VR. This application has been in use for PR purposes for several years now.

For the nowadays more commonly used triangle mesh representation of urban scenes we introduced a semantic segmentation pipeline. A lot of effort is put in the semantic interpretation of point clouds, meshes have been somewhat neglected so far, so we see this as an important topic. Our approach is a hybrid model, explicit feature calculation is combined with convolutional feature learning. For each face, a multi-scale feature vector is generated. Due to the lack of benchmarks for urban meshes that provide a semantic labeling for each triangle, we generated our own data set. As a baseline, we trained a RF. Our 1D CNN approach

achieves slightly better results in the range of up to 80% overall accuracy while being significantly faster, both in training and inference time.

Once a mesh is semantically interpretable, it is possible to locate buildings. Thus it is possible to get a more fine-grained classification of different building use types. To that end, we presented a framework for the automatic generation of building use classification training data. By linking GSV imagery with cadastral data, it is possible to generate meaningful training samples. To distinguish between different use types of buildings, we used this crawled data to train a CNN classifier. Such a framework can be used for large scale building use classification. A typical use case for such a scenario is the enrichment of OSM data. If the building use type can be classified from imagery, this information can in the future in turn be used to refine geometry of building presentations as discussed earlier in our grammar-based enhancement.

In addition to the numeric results of the CNN pipeline, we thoroughly inspected CAMs in order to get an impression of machine-learned features, how the network behaves when presented with unknown building representation types or manipulated imagery. Results showed that although trained purely on GSV imagery, the CNN is nevertheless able to correctly predict samples from different representation forms such as textured GE meshes or even pictograms. We introduced an application of CAMs to steer importance-based abstract renderings. Such stipplings can be useful to guide the users attention to specific features in order to quickly recognize a certain building category.

The landscape of virtual building models is vast. No standardized data exchange format has been established yet. Therefore, this work investigated several data representations of virtual city models, how to extract semantic information and, in turn, how to apply this information in a meaningful way. It remains unclear what the final photogrammetric product will be in the future. In our perspective, textured meshes have a high potential, due to the compact representation, adjacency information and explicit texture maps. Semantic information could potentially be encoded by adding per face (semantic) material, as we did for the ground truth labeling process described in section 4.1.2. Nevertheless, for urban planning and simulation purposes BReps are indispensable.

The enhancement of virtual building representations will become even more important in the future. The presented user studies are a first endeavor in order to extract knowledge of human understanding of building models. In the future, crowd-sourcing is a promising way to extract such knowledge much faster, with more user diversity on a large scale. Integrating essential feature knowledge into grammar-based approaches enables the generation of semantically enriched building models. Extending the focus from a single building of interest to urban parcels opens up additional research topics. Homogeneity descriptors could be defined for such urban parcels. If local homogeneity is high, building blocks can be abstracted by a grammar-based approach, effectively merging them to larger superstructures. This way, semantic information is retained while complexity is decreased.

With the rapid developments in DL, semantic segmentation of urban triangle meshes will undoubtedly (a) get more attention from research and (b) consequently make rapid advances in generalization capacity and scalability. A crucial prerequisite, though, is the availability of ground truth data. Hence, semi-supervised and unsupervised learning schemes might become more prevalent in order to efficiently (pre-)segment mesh data and assist supervised algorithms. GCNNs have shown potential for point cloud classification tasks. Up now, however, they operate on vertex-based features and exploit edge connections. Therefore, it is not possible yet to directly incorporate textural information, which extends over complete faces. To bypass this, first approaches have been proposed to project local geodesic neighborhoods onto planes and perform specifically defined convolution operations on this domain [Huang et al., 2018]. Further developments must therefore show what is best suited for processing triangle meshes.

Image-based building use classification can greatly profit from crowd-sourcing. Mechanisms like reCAPTCHA [Von Ahn et al., 2008] serve as identification for human user input on websites. One exemplary task is to select images containing storefronts from GSV imagery. Similarly, [Hillen and Höfle, 2015] use this concept to digitize building footprints. Those are methods to cheaply (and for the user sometimes even unknowingly)

generate training data, and classification algorithms undoubtedly profit thereof. Fine-grained urban object interpretation, not only for building use classification, but also street furniture and vegetation, will only be successful on an instance-level. Latest approaches like **Mask R-CNN** have proven to be very successful in this context and can thus be leveraged in the domain of urban environment understanding. Eventually, semantic interpretation will presumably find its way into all parts of the image-based reconstruction pipeline. That is, not only the segmentation of point clouds and meshes but within the image matching process itself. Semantic interpretation can serve as prior and constraint, for instance, to steer smoothness constraints to preserve sharp edges on buildings.

This work has shown a direction to tackle individual parts of the comprehensive problem of representation, understanding, and interpretability of virtual city models by introducing semantics on various levels. Besides the point cloud based reconstruction of buildings, we proposed an approach for the enrichment of building models based on human perception. Furthermore, an algorithm for the area-wide analysis of textured meshes was introduced. For a more detailed interpretation of individual buildings, a pipeline for the automated generation of training data and a corresponding DL classification approach were presented. While the semantic understanding and enhancement of virtual city models is far from being solved, this thesis introduced several concepts that contribute to the overarching goal of generating understandable representations of urban scenes.

Bibliography

- [Adabala, 2009] Adabala, N. (2009). A Technique for Building Representation in Oblique View Maps of Modern Urban Areas. *The Cartographic Journal*, 46(2):104–114.
- [ADV, 2018] ADV, Arbeitsgemeinschaft der Vermessungsverwaltungen der Länder der Bundesrepublik Deutschland. (2018). Dokumentation zur Modellierung der Geoinformationen des amtlichen Vermessungswesens. <http://www.adv-online.de/icc/extdeu/nav/a63/binarywriterservlet?imgUid=b001016e-7efa-8461-e336-b6951fa2e0c9&uBasVariant=11111111-1111-1111-1111-111111111111>.
- [Armeni et al., 2017] Armeni, I., Sax, A., Zamir, A. R., and Savarese, S. (2017). Joint 2D-3D-Semantic Data for Indoor Scene Understanding. *ArXiv e-prints*.
- [Badrinarayanan et al., 2017] Badrinarayanan, V., Kendall, A., and Cipolla, R. (2017). Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE transactions on pattern analysis and machine intelligence*, 39(12):2481–2495.
- [Ball et al., 2017] Ball, J. E., Anderson, D. T., and Chan, C. S. (2017). Comprehensive survey of deep learning in remote sensing: theories, tools, and challenges for the community. *Journal of Applied Remote Sensing*, 11(4):042609.
- [Becker, 2009] Becker, S. (2009). Generation and application of rules for quality dependent façade reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 64(6):640–653.
- [Becker, 2011] Becker, S. (2011). *Automatische Ableitung und Anwendung von Regeln fuer die Rekonstruktion von Fassaden aus heterogenen Sensordaten*. PhD thesis.
- [Becker and Haala, 2009] Becker, S. and Haala, N. (2009). Grammar supported facade reconstruction from mobile lidar mapping. In *ISPRS Workshop, CMRT09-City Models, Roads and Traffic*, volume 38, page 13.
- [Becker et al., 2008] Becker, S., Haala, N., and Fritsch, D. (2008). Combined knowledge propagation for facade reconstruction. In *ISPRS Congress Beijing*, pages 423–430.
- [Bengio, 2012] Bengio, Y. (2012). Practical recommendations for gradient-based training of deep architectures. In *Neural networks: Tricks of the trade*, pages 437–478. Springer.
- [Bewley et al., 2018] Bewley, A., Rigley, J., Liu, Y., Hawke, J., Shen, R., Lam, V.-D., and Kendall, A. (2018). Learning to Drive from Simulation without Real World Labels.
- [Biljecki et al., 2016a] Biljecki, F., Ledoux, H., and Stoter, J. (2016a). Generation of multi-lod 3d city models in citygml with the procedural modelling engine random3dcity. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 3(4).
- [Biljecki et al., 2016b] Biljecki, F., Ledoux, H., and Stoter, J. (2016b). An improved lod specification for 3d building models. *Computers, Environment and Urban Systems*, 59:25–37.
- [Biljecki et al., 2015] Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889.
- [Bittner et al., 2018] Bittner, K., D’Angelo, P., Körner, M., and Reinartz, P. (2018). DSM-to-LoD2: Spaceborne Stereo Digital Surface Model Refinement. *Remote Sensing*, 10(12):1926.
- [Blaha et al., 2017] Blaha, M., Rothermel, M., Oswald, M. R., Sattler, T., Richard, A., Wegner, J. D., Pollefeys, M., and Schindler, K. (2017). Semantically informed multiview surface refinement. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 3819–3827.

- [Bottou, 2013] Bottou, L. (2013). Large-Scale Learning with Stochastic Gradient Descent. In *Neural Networks: Tricks of the Trade, Reloaded*.
- [Boulch et al., 2017] Boulch, A., Saux, B. L., and Audebert, N. (2017). Unstructured Point Cloud Semantic Labeling Using Deep Segmentation Networks. In Pratikakis, I., Dupont, F., and Ovsjanikov, M., editors, *Eurographics Workshop on 3D Object Retrieval*. The Eurographics Association.
- [Bousmalis et al., 2018] Bousmalis, K., Irpan, A., Wohlhart, P., Bai, Y., Kelcey, M., Kalakrishnan, M., Downs, L., Ibarz, J., Pastor, P., Konolige, K., et al. (2018). Using simulation and domain adaptation to improve efficiency of deep robotic grasping. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4243–4250. IEEE.
- [Boussaha et al., 2018] Boussaha, M., Vallet, B., and Rives, P. (2018). Large scale textured mesh reconstruction from mobile mapping images and LiDAR scans. In *ISPRS 2018 - International Society for Photogrammetry and Remote Sensing*, pages 49–56.
- [Branson et al., 2018] Branson, S., Wegner, J. D., Hall, D., Lang, N., Schindler, K., and Perona, P. (2018). From google maps to a fine-grained catalog of street trees. *ISPRS Journal of Photogrammetry and Remote Sensing*, 135:13–30.
- [Bronstein et al., 2016] Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. (2016). Geometric deep learning: going beyond euclidean data. *CoRR*, abs/1611.08097.
- [Brown et al., 2017] Brown, T. B., Mané, D., Roy, A., Abadi, M., and Gilmer, J. (2017). Adversarial patch. *arXiv preprint arXiv:1712.09665*.
- [Cabezas et al., 2015] Cabezas, R., Straub, J., and Fisher III, J. W. (2015). Semantically-Aware Aerial Reconstruction from Multi-Modal Data. In *International Conference on Computer Vision (ICCV)*.
- [Caesar et al., 2016] Caesar, H., Uijlings, J., and Ferrari, V. (2016). Region-based semantic segmentation with end-to-end training. *arXiv:1607.07671 [cs]*.
- [Cao et al., 2007] Cao, F., Delon, J., Desolneux, A., Musé, P., and Sur, F. (2007). A unified framework for detecting groups and application to shape recognition. *Journal of Mathematical Imaging and Vision*, 27(2):91–119.
- [Cauchy, 1847] Cauchy, A. (1847). Méthode générale pour la résolution de systèmes d’équations simultanées. In *European Conference on Computer Vision*, pages 536–538.
- [Cavegn et al., 2014] Cavegn, S., Haala, N., Nebiker, S., Rothermel, M., and Tutzauer, P. (2014). Benchmarking high density image matching for oblique airborne imagery. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(3):45.
- [Chang et al., 2018] Chang, J., Gu, J., Wang, L., Meng, G., Xiang, S., and Pan, C. (2018). Structure-aware convolutional neural networks. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R., editors, *Advances in Neural Information Processing Systems 31*, pages 11–20. Curran Associates, Inc.
- [Chen et al., 2019] Chen, J., Zhou, Y., Zipf, A., and Fan, H. (2019). Deep Learning From Multiple Crowds: A Case Study of Humanitarian Mapping. *IEEE Transactions on Geoscience and Remote Sensing*, 57(3):1713–1722.
- [Chen et al., 2017] Chen, L.-C., Papandreou, G., Schroff, F., and Adam, H. (2017). Rethinking Atrous Convolution for Semantic Image Segmentation.
- [Chen et al., 2018a] Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., and Adam, H. (2018a). Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 801–818.
- [Chen et al., 2018b] Chen, Y., Li, W., Chen, X., and Van Gool, L. (2018b). Learning Semantic Segmentation from Synthetic Data: A Geometrically Guided Input-Output Adaptation Approach.
- [Chollet et al., 2015] Chollet, F. et al. (2015). Keras. <https://keras.io>.
- [Chu et al., 2016] Chu, H., Wang, S., Urtasun, R., and Fidler, S. (2016). HouseCraft: Building Houses from Rental Ads and Street Views. In *European Conference on Computer Vision*, Lecture Notes in Computer Science, pages 500–516. Springer, Cham.
- [Cisco, 2017] Cisco (2017). The Zettabyte Era: Trends and Analysis.
- [Cordts et al., 2016] Cordts, M., Omran, M., Ramos, S., Rehfeld, T., Enzweiler, M., Benenson, R., Franke, U., Roth, S., and Schiele, B. (2016). The Cityscapes Dataset for Semantic Urban Scene Understanding.

- [Cramer et al., 2018] Cramer, M., Haala, N., Laupheimer, D., Mandlbürger, G., and Havel, P. (2018). Ultra-high precision uav-based LiDAR and dense image matching. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-1:115–120.
- [Dai et al., 2017] Dai, A., Chang, A. X., Savva, M., Halber, M., Funkhouser, T., and Nießner, M. (2017). ScanNet: Richly-annotated 3d reconstructions of indoor scenes. In *Proc. Computer Vision and Pattern Recognition (CVPR)*, IEEE.
- [Deng et al., 2009] Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., and Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255.
- [Deussen et al., 2000] Deussen, O., Hiller, S., Van Overveld, C., and Strothotte, T. (2000). Floating points: A method for computing stipple drawings. In *Computer Graphics Forum*, volume 19, pages 41–50. Wiley Online Library.
- [Deussen et al., 2017] Deussen, O., Spicker, M., and Zheng, Q. (2017). Weighted linde-buzo-gray stippling. *ACM Transactions on Graphics*, 36(6):1–12.
- [Dumoulin and Visin, 2016] Dumoulin, V. and Visin, F. (2016). A guide to convolution arithmetic for deep learning.
- [El-Mekawy et al., 2011] El-Mekawy, M., Östman, A., and Shahzad, K. (2011). Towards interoperating citygml and ifc building models: a unified model based approach. In *Advances in 3D geo-information sciences*, pages 73–93. Springer.
- [Fukushima, 1980] Fukushima, K. (1980). Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological cybernetics*, 36(4):193–202.
- [Gadde et al., 2018] Gadde, R., Jampani, V., Marlet, R., and Gehler, P. V. (2018). Efficient 2d and 3d facade segmentation using auto-context. *IEEE transactions on pattern analysis and machine intelligence*, 40(5):1273–1280.
- [Gebru et al., 2017] Gebru, T., Krause, J., Wang, Y., Chen, D., Deng, J., Aiden, E. L., and Fei-Fei, L. (2017). Using Deep Learning and Google Street View to Estimate the Demographic Makeup of the US. *arXiv:1702.06683 [cs]*.
- [Geiger et al., 2013] Geiger, A., Lenz, P., Stiller, C., and Urtasun, R. (2013). Vision Meets Robotics: The KITTI Dataset. *Int. J. Rob. Res.*, 32(11):1231–1237.
- [Geofly, 2018] Geofly (2018). Building extraction from 3d meshes. <https://web.geofly.eu/de/news/building-extraction-from-3d-mesh.html>. [Online; accessed 07-May-2019].
- [George et al., 2017] George, D., Xie, X., and Tam, G. K. (2017). 3d mesh segmentation via multi-branch 1d convolutional neural networks. *arXiv preprint arXiv:1705.11050*.
- [Gevaert et al., 2018] Gevaert, C., Persello, C., Nex, F., and Vosselman, G. (2018). A deep learning approach to DTM extraction from imagery using rule-based training labels. *ISPRS Journal of Photogrammetry and Remote Sensing*, 142:106–123.
- [Girshick, 2015] Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.
- [Girshick et al., 2014] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.
- [Glander and Döllner, 2009] Glander, T. and Döllner, J. (2009). Abstract representations for interactive visualization of virtual 3D city models. *Computers, Environment and Urban Systems*, 33(5):375–387.
- [Glorot and Bengio, 2010] Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the thirteenth international conference on artificial intelligence and statistics*, pages 249–256.
- [Glorot et al., 2011] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep Sparse Rectifier Neural Networks. *JMLR W&CP*, (15):315–323.
- [Gooch and Gooch, 2001] Gooch, B. and Gooch, A. (2001). *Non-photorealistic rendering*. AK Peters/CRC Press.
- [Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press. <http://www.deeplearningbook.org>.

- [Goodfellow et al., 2014a] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., and Bengio, Y. (2014a). Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.
- [Goodfellow et al., 2014b] Goodfellow, I. J., Shlens, J., and Szegedy, C. (2014b). Explaining and harnessing adversarial examples. *arXiv preprint arXiv:1412.6572*.
- [Graham et al., 2018] Graham, B., Engelcke, M., and van der Maaten, L. (2018). 3d semantic segmentation with submanifold sparse convolutional networks. pages 9224–9232.
- [Gröger et al., 2012] Gröger, G., Kolbe, T., Nagel, C., and Häfele, K. (2012). Ogc city geography markup language (citygml) encoding standard, version 2.0, ogc doc no. 12-019. *Open Geospatial Consortium*.
- [Haala, 2013] Haala, N. (2013). The landscape of dense image matching algorithms.
- [Haala and Kada, 2010] Haala, N. and Kada, M. (2010). ISPRS Journal of Photogrammetry and Remote Sensing An update on automatic 3D building reconstruction. *ISPRS Journal of Photogrammetry and Remote Sensing*, 65:570–580.
- [Hackel et al., 2017] Hackel, T., Savinov, N., Ladicky, L., Wegner, J. D., Schindler, K., and Pollefeys, M. (2017). SEMANTIC3D.NET: A new large-scale point cloud classification benchmark. In *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, volume IV-1-W1, pages 91–98.
- [Hackel et al., 2016] Hackel, T., Wegner, J. D., and Schindler, K. (2016). Fast Semantic Segmentation of 3D Point Clouds With Strongly Varying Density. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 3(3).
- [Han, 2016] Han, M. (2016). Photogrammetry in Virtual Reality Environments. Master’s thesis, University of Stuttgart, Institute for Photogrammetry.
- [He and Upcroft, 2013] He, H. and Upcroft, B. (2013). Nonparametric semantic segmentation for 3d street scenes. In Amato, N., editor, *IROS2013: IEEE/RSJ International Conference on Intelligent Robots and Systems : New Horizon*, Tokyo, Japan.
- [He et al., 2017] He, K., Gkioxari, G., Dollar, P., and Girshick, R. (2017). Mask R-CNN. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 2980–2988.
- [He et al., 2016] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.
- [Hecht, 2014] Hecht, R. (2014). Automatische Klassifizierung von Gebäudegrundrissen.
- [Henon, 2017] Henon, Y. (2017). keras-frcnn. <https://github.com/yhenon/keras-frcnn>.
- [Hillen and Höfle, 2015] Hillen, F. and Höfle, B. (2015). Geo-recaptcha: Crowdsourcing large amounts of geographic information from earth observation data. *International journal of applied earth observation and geoinformation*, 40:29–38.
- [Hirschmüller, 2008] Hirschmüller, H. (2008). Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341.
- [Hornik et al., 1990] Hornik, K., Stinchcombe, M., and White, H. (1990). Universal approximation of an unknown mapping and its derivatives using multilayer feedforward networks. *Neural networks*, 3(5):551–560.
- [Hua et al., 2016] Hua, B., Pham, Q., Nguyen, D. T., Tran, M., Yu, L., and Yeung, S. (2016). Scenenn: A scene meshes dataset with annotations. In *2016 Fourth International Conference on 3D Vision (3DV)*, pages 92–101.
- [Huang and Mayer, 2015] Huang, H. and Mayer, H. (2015). Robust and efficient urban scene classification using relative features. In *Proceedings of the 23rd SIGSPATIAL International Conference on Advances in Geographic Information Systems - GIS ’15*, pages 1–4, New York, New York, USA. ACM Press.
- [Huang and You, 2016] Huang, J. and You, S. (2016). Point cloud labeling using 3d convolutional neural network. In *2016 23rd International Conference on Pattern Recognition (ICPR)*, pages 2670–2675.
- [Huang et al., 2018] Huang, J., Zhang, H., Yi, L., Funkhouser, T., Nießner, M., and Guibas, L. (2018). Texturenet: Consistent local parametrizations for learning from high-resolution signals on meshes. *arXiv preprint arXiv:1812.00020*.
- [Isola et al., 2016] Isola, P., Zhu, J.-Y., Zhou, T., and Efros, A. A. (2016). Image-to-Image Translation with Conditional Adversarial Networks.

- [Kalogerakis et al., 2017] Kalogerakis, E., Averkiou, M., Maji, S., and Chaudhuri, S. (2017). 3D Shape segmentation with projective convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017-January:6630–6639.
- [Kang et al., 2018] Kang, J., Körner, M., Wang, Y., Taubenböck, H., and Zhu, X. X. (2018). Building instance classification using street view images. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145:44–59.
- [Kelly et al., 2018] Kelly, T., Guerrero, P., Steed, A., Wonka, P., and Mitra, N. J. (2018). Frankengan: Guided detail synthesis for building mass models using style-synchronized gans. *ACM Trans. Graph.*, 37(6):216:1–216:14.
- [Keskar et al., 2016] Keskar, N. S., Mudigere, D., Nocedal, J., Smelyanskiy, M., and Tang, P. T. P. (2016). On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima.
- [Kim et al., 2019] Kim, B., Reif, E., Wattenberg, M., and Bengio, S. (2019). Do neural networks show gestalt phenomena? an exploration of the law of closure. *arXiv preprint arXiv:1903.01069*.
- [Kingma and Ba, 2014] Kingma, D. P. and Ba, J. (2014). Adam: A Method for Stochastic Optimization.
- [Kipf and Welling, 2016] Kipf, T. N. and Welling, M. (2016). Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*.
- [Kirillov et al., 2019] Kirillov, A., Girshick, R., He, K., and Dollár, P. (2019). Panoptic Feature Pyramid Networks.
- [Kolbe et al., 2005] Kolbe, T. H., Gröger, G., and Plümer, L. (2005). CityGML: Interoperable access to 3D city models. In *Geo-information for disaster management*, pages 883–899. Springer.
- [Kölle et al., 2019] Kölle, M., Laupheimer, D., and Haala, N. (2019). Klassifikation hochaufgelöster LiDAR- und MVS-punktwolken zu monitoringzwecken. In *39. Wissenschaftlich-Technische Jahrestagung der OVG, DGPF und SGPF in Wien*, volume 28, pages 692–701. Deutsche Gesellschaft für Photogrammetrie, Fernerkundung und Geoinformation (DGPF) e.V.
- [Kotikalapudi, 2017] Kotikalapudi, R. (2017). Keras visualization toolkit. <https://github.com/raghakot/keras-vis>.
- [Krasin et al., 2017] Krasin, I., Duerig, T., Alldrin, N., Ferrari, V., Abu-El-Haija, S., Kuznetsova, A., Rom, H., Uijlings, J., Popov, S., Veit, A., et al. (2017). Openimages: A public dataset for large-scale multi-label and multi-class image classification. *Dataset available from <https://github.com/openimages>*, 2:3.
- [Krizhevsky et al., 2012] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.
- [Kubovy and Van Den Berg, 2008] Kubovy, M. and Van Den Berg, M. (2008). The whole is equal to the sum of its parts: A probabilistic model of grouping by proximity and similarity in regular patterns. *Psychological review*, 115(1):131.
- [Kurakin et al., 2016] Kurakin, A., Goodfellow, I., and Bengio, S. (2016). Adversarial examples in the physical world. *arXiv preprint arXiv:1607.02533*.
- [Landrieu and Simonovsky, 2017] Landrieu, L. and Simonovsky, M. (2017). Large-scale point cloud semantic segmentation with superpoint graphs. *CoRR*, abs/1711.09869.
- [Laupheimer, 2016] Laupheimer, D. (2016). Deep Learning for the Classification of Building Facades. Master’s thesis, University of Stuttgart, Institute for Photogrammetry.
- [Laupheimer et al., 2018] Laupheimer, D., Tutzauer, P., Haala, N., and Spicker, M. (2018). Neural networks for the classification of building use from street-view imagery. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4(2).
- [Lawin et al., 2017] Lawin, F. J., Danelljan, M., Tosteberg, P., Bhat, G., Khan, F. S., and Felsberg, M. (2017). Deep projective 3d semantic segmentation. *CoRR*, abs/1705.03428.
- [LeCun et al., 2015] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *Nature*, 521(7553):436–444.
- [LeCun et al., 1998] LeCun, Y., Bottou, L., Bengio, Y., Haffner, P., et al. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.
- [LeCun and Cortes, 2010] LeCun, Y. and Cortes, C. (2010). MNIST handwritten digit database.
- [Lee et al., 2018] Lee, K.-H., Ros, G., Li, J., and Gaidon, A. (2018). Spigan: Privileged adversarial learning from simulation. *arXiv preprint arXiv:1810.03756*.

- [Li et al., 2004] Li, Z., Yan, H., Ai, T., and Chen, J. (2004). Automated building generalization based on urban morphology and Gestalt theory. *International Journal of Geographical Information Science*, 18(5):513–534.
- [Liao et al., 2013] Liao, Y., Rubinsteyn, A., Power, R., and Li, J. (2013). Learning Random Forests on the GPU. In *Big Learning 2013: Advances in Algorithms and Data Management*.
- [Lin et al., 2017] Lin, T. Y., Goyal, P., Girshick, R., He, K., and Dollar, P. (2017). Focal Loss for Dense Object Detection. *Proceedings of the IEEE International Conference on Computer Vision*, 2017-October:2999–3007.
- [Liu et al., 2015] Liu, W., Rabinovich, A., and Berg, A. C. (2015). Parsenet: Looking wider to see better. *CoRR*, abs/1506.04579.
- [Long et al., 2014] Long, J., Shelhamer, E., and Darrell, T. (2014). Fully Convolutional Networks for Semantic Segmentation. *arXiv:1411.4038 [cs]*.
- [Long et al., 2015] Long, J., Shelhamer, E., and Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3431–3440.
- [Luo et al., 2019] Luo, L., Xiong, Y., Liu, Y., and Sun, X. (2019). Adaptive gradient methods with dynamic bound of learning rate. *arXiv preprint arXiv:1902.09843*.
- [Maaten and Hinton, 2008] Maaten, L. v. d. and Hinton, G. (2008). Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(Nov):2579–2605.
- [Madry et al., 2017] Madry, A., Makelov, A., Schmidt, L., Tsipras, D., and Vladu, A. (2017). Towards deep learning models resistant to adversarial attacks. *arXiv preprint arXiv:1706.06083*.
- [Marcos et al., 2018] Marcos, D., Volpi, M., Kellenberger, B., and Tuia, D. (2018). Land cover mapping at very high resolution with rotation equivariant cnns: Towards small yet accurate models. *ISPRS Journal of Photogrammetry and Remote Sensing*, 145:96–107.
- [Martinović et al., 2015] Martinović, A., Knopp, J., Riemenschneider, H., and Van Gool, L. (2015). 3D all the way: Semantic segmentation of urban scenes from start to end in 3D. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 07-12-June, pages 4456–4465. IEEE.
- [Mathias et al., 2015] Mathias, M., An, ., Delo Martinovi, ., Luc, ., Gool, V., Delo, ., and Luc, M. . (2015). ATLAS: A Three-Layered Approach to Facade Parsing. *International Journal of Computer Vision*.
- [McCulloch and Pitts, 1943] McCulloch, W. S. and Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *The bulletin of mathematical biophysics*, 5(4):115–133.
- [Michaelsen et al., 2012] Michaelsen, E., Iwaszczuk, D., Sirmacek, B., Hoegner, L., and Stilla, U. (2012). Gestalt grouping on faade textures from ir image sequences: Comparing different production systemse. *International Archives of Photogrammetry, Remote Sensing and Spatial Information Science*, 39(B3):303–308.
- [Minsky and Papert, 1969] Minsky, M. and Papert, S. (1969). *Perceptrons: An Introduction to Computational Geometry*. MIT Press, Cambridge, MA, USA.
- [Mitsuhara et al., 2019] Mitsuhara, M., Fukui, H., Sakashita, Y., Ogata, T., Hirakawa, T., Yamashita, T., and Fujiyoshi, H. (2019). Embedding Human Knowledge in Deep Neural Network via Attention Map.
- [Movshovitz-Attias et al., 2015] Movshovitz-Attias, Y., Yu, Q., Stumpe, M. C., Shet, V., Arnoud, S., and Yatziv, L. (2015). Ontological supervision for fine grained classification of Street View storefronts. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1693–1702.
- [Müller et al., 2018] Müller, M., Casser, V., Lahoud, J., Smith, N., and Ghanem, B. (2018). Sim4CV: A Photo-Realistic Simulator for Computer Vision Applications. *International Journal of Computer Vision*, 126(9):902–919.
- [Müller et al., 2006] Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L. (2006). Procedural modeling of buildings. In *Acm Transactions On Graphics (Tog)*, volume 25, pages 614–623. ACM.
- [Müller et al., 2007] Müller, P., Zeng, G., Wonka, P., and Van Gool, L. (2007). Image-based procedural modeling of facades. In *ACM Transactions on Graphics (TOG)*, volume 26, page 85. ACM.
- [Musialski et al., 2012] Musialski, P., Wimmer, M., and Wonka, P. (2012). Interactive coherence-based façade modeling. In *Computer Graphics Forum*, volume 31, pages 661–670. Wiley Online Library.
- [Musialski et al., 2013] Musialski, P., Wonka, P., Aliaga, D. G., Wimmer, M., van Gool, L., and Purgathofer, W. (2013). A Survey of Urban Reconstruction. *Computer Graphics Forum*, 32(6):146–177.

- [Nair and Hinton, 2010] Nair, V. and Hinton, G. (2010). Rectified Linear Units Improve Restricted Boltzmann Machines. In *Proceedings of the 27th International Conference on Machine Learning*, pages 807–814.
- [Nan et al., 2011] Nan, L., Sharf, A., Xie, K., Wong, T.-T., Deussen, O., Cohen-Or, D., and Chen, B. (2011). Conjoining Gestalt rules for abstraction of architectural drawings. In *Proceedings of the 2011 SIGGRAPH Asia Conference on - SA '11*, page 1.
- [Nan et al., 2010] Nan, L., Sharf, A., Zhang, H., Cohen-Or, D., and Chen, B. (2010). Smartboxes for interactive urban reconstruction. *ACM Transactions on Graphics (TOG)*, 29(4):93.
- [Narvekar and Karam, 2011] Narvekar, N. D. and Karam, L. J. (2011). A No-Reference Image Blur Metric Based on the Cumulative Probability of Blur Detection (CPBD). *IEEE Transactions on Image Processing*, 20(9):2678–2683.
- [Neuhold et al., 2017] Neuhold, G., Ollmann, T., Rota Bulò, S., and Kotschieder, P. (2017). The mapillary vistas dataset for semantic understanding of street scenes. In *International Conference on Computer Vision (ICCV)*.
- [Nguattem and Mayer, 2017] Nguattem, W. and Mayer, H. (2017). Modeling Urban Scenes from Pointclouds. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 3857–3866.
- [Nielsen, 2015] Nielsen, M. A. (2015). *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, USA.
- [Niemeyer et al., 2014] Niemeyer, J., Rottensteiner, F., and Sörgel, U. (2014). Contextual classification of LiDAR data and building object detection in urban areas. *ISPRS Journal of Photogrammetry and Remote Sensing*, 87:152 – 165.
- [Nishida et al., 2018] Nishida, G., Bousseau, A., and Aliaga, D. G. (2018). Procedural modeling of a building from a single image. In *Computer Graphics Forum*, volume 37, pages 415–429. Wiley Online Library.
- [Nishida et al., 2016] Nishida, G., Garcia-Dorado, I., Aliaga, D. G., Benes, B., and Bousseau, A. (2016). Interactive sketching of urban procedural models. *ACM Transactions on Graphics*, 35(4):1–11.
- [Noh et al., 2015] Noh, H., Hong, S., and Han, B. (2015). Learning deconvolution network for semantic segmentation. *CoRR*, abs/1505.04366.
- [Ong et al., 2003] Ong, E., Lin, W., Lu, Z., Yang, X., Yao, S., Pan, F., Jiang, L., and Moschetti, F. (2003). A no-reference quality metric for measuring image blur. In *Seventh International Symposium on Signal Processing and Its Applications, 2003. Proceedings.*, volume 1, pages 469–472.
- [Pasewaldt et al., 2014] Pasewaldt, S., Semmo, A., Trapp, M., and Döllner, J. (2014). Multi-perspective 3d panoramas. *International Journal of Geographical Information Science*, 28(10):2030–2051.
- [Peter, 2015] Peter, A. (2015). Street view crawler. <https://github.com/peterashwell/streetview-crawler>.
- [Qi et al., 2017] Qi, C. R., Yi, L., Su, H., and Guibas, L. J. (2017). Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Advances in Neural Information Processing Systems*, pages 5105–5114.
- [Reinhardt, 2019] Reinhardt, T. (2019). Using global localization to improve navigation. [Online; posted 11-February-2019].
- [Remondino et al., 2013] Remondino, F., Spera, M. G., Nocerino, E., Menna, F., Nex, F., and Gonizzi-Barsanti, S. (2013). Dense image matching: Comparisons and analyses. In *2013 Digital Heritage International Congress (DigitalHeritage)*, pages 47–54. IEEE.
- [Ren et al., 2015] Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems*, pages 91–99.
- [Ripperda, 2010] Ripperda, N. (2010). *Rekonstruktion von Fassadenstrukturen mittels formaler Grammatiken und Reversible Jump Markov Chain Monte Carlo Sampling*. PhD thesis, University of Hanover.
- [Ritter et al., 2017] Ritter, S., Barrett, D. G. T., Santoro, A., and Botvinick, M. M. (2017). Cognitive psychology for deep neural networks: A shape bias case study. In Precup, D. and Teh, Y. W., editors, *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 2940–2949, International Convention Centre, Sydney, Australia. PMLR.
- [Ronneberger et al., 2015] Ronneberger, O., Fischer, P., and Brox, T. (2015). U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pages 234–241. Springer.

- [Ros et al., 2016] Ros, G., Sellart, L., Materzynska, J., Vazquez, D., and Lopez, A. M. (2016). The SYNTHIA Dataset: A Large Collection of Synthetic Images for Semantic Segmentation of Urban Scenes. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 3234–3243.
- [Rosenblatt, 1958] Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386.
- [Rothermel et al., 2012] Rothermel, M., Wenzel, K., Fritsch, D., and Haala, N. (2012). Sure: Photogrammetric surface reconstruction from imagery. In *Proceedings LC3D Workshop, Berlin*, volume 8, page 2.
- [Rouhani et al., 2017] Rouhani, M., Lafarge, F., and Alliez, P. (2017). Semantic Segmentation of 3D Textured Meshes for Urban Scene Analysis. *ISPRS Journal of Photogrammetry and Remote Sensing*, 123:124–139.
- [Roynard et al., 2018] Roynard, X., Deschaut, J., and Goulette, F. (2018). Classification of point cloud scenes with multiscale voxel deep network. *CoRR*, abs/1804.03583.
- [Rumelhart et al., 1988] Rumelhart, D. E., Hinton, G. E., Williams, R. J., et al. (1988). Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.
- [Schmidhuber, 2015] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117.
- [Schmittwilken, 2011] Schmittwilken, J. (2011). *Attributierte Grammatiken zur Rekonstruktion und Interpretation von Fassaden*. PhD thesis.
- [Schmittwilken et al., 2006] Schmittwilken, J., Kolbe, T. H., and Plümer, L. (2006). Der Gebäudekragen – Eine detaillierte Betrachtung des Übergangs von Gebäude und Gelände. *Geoinformatik und Erdbeobachtung*, 26:127–135.
- [Schmitz and Mayer, 2016] Schmitz, M. and Mayer, H. (2016). A convolutional network for semantic facade segmentation and interpretation. In *International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences - ISPRS Archives*, volume 41, pages 709–715.
- [Schmohl and Sörgel, 2019] Schmohl, S. and Sörgel, U. (2019). Submanifold sparse convolutional networks for semantic segmentation of large-scale point clouds. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W5:77–84.
- [Schwarz and Müller, 2015] Schwarz, M. and Müller, P. (2015). Advanced procedural modeling of architecture. *ACM Transactions on Graphics (TOG)*, 34(4):107.
- [Secord and Adrian, 2002] Secord, A. and Adrian (2002). Weighted Voronoi stippling. In *Proceedings of the second international symposium on Non-photorealistic animation and rendering - NPAR '02*, page 37, New York, New York, USA. ACM Press.
- [Selvaraju et al., 2017] Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., and Batra, D. (2017). Grad-CAM: Visual Explanations from Deep Networks via Gradient-Based Localization. In *Proceedings of the IEEE International Conference on Computer Vision*, volume 2017-Octob, pages 618–626. IEEE.
- [Shah et al., 2017] Shah, S., Dey, D., Lovett, C., and Kapoor, A. (2017). Airsim: High-fidelity visual and physical simulation for autonomous vehicles. In *Field and Service Robotics*.
- [Shilane et al., 2004] Shilane, P., Min, P., Kazhdan, M., and Funkhouser, T. (2004). The princeton shape benchmark. In *Shape modeling applications, 2004. Proceedings*, pages 167–178. IEEE.
- [Simonyan and Zisserman, 2014] Simonyan, K. and Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.
- [Simonyan and Zisserman, 2015] Simonyan, K. and Zisserman, A. (2015). Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*.
- [Smelik et al., 2014] Smelik, R. M., Tutenel, T., Bidarra, R., and Benes, B. (2014). A Survey on Procedural Modelling for Virtual Worlds. *Computer Graphics Forum*, 33(6):31–50.
- [Snavely et al., 2006] Snavely, N., Seitz, S. M., and Szeliski, R. (2006). Photo tourism: exploring photo collections in 3d. In *ACM transactions on graphics (TOG)*, volume 25, pages 835–846. ACM.
- [Srivastava et al., 2014] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

- [Su et al., 2018] Su, H., Jampani, V., Sun, D., Maji, S., Kalogerakis, E., Yang, M.-H., and Kautz, J. (2018). SPLAT-Net: Sparse lattice networks for point cloud processing. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2530–2539.
- [Szegedy et al., 2016] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., and Wojna, Z. (2016). Rethinking the Inception Architecture for Computer Vision. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, volume 2016-Decem, pages 2818–2826.
- [Szegedy et al., 2013] Szegedy, C., Zaremba, W., Sutskever, I., Bruna, J., Erhan, D., Goodfellow, I., and Fergus, R. (2013). Intriguing properties of neural networks. *arXiv preprint arXiv:1312.6199*.
- [Taime et al., 2018] Taime, A., Saaïdi, A., and Satori, K. (2018). A new semantic segmentation approach of 3d mesh using the stereoscopic image colors. *Multimedia Tools and Applications*, 77(20):27143–27162.
- [Te et al., 2018] Te, G., Hu, W., Guo, Z., and Zheng, A. (2018). RGCNN: Regularized graph cnn for point cloud segmentation. *arXiv preprint arXiv:1806.02952*.
- [Teboul et al., 2010] Teboul, O., Simon, L., Koutsourakis, P., and Paragios, N. (2010). Segmentation of building facades using procedural shape priors. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3105–3112. IEEE.
- [Theodoridis and Koutroumbas, 2008] Theodoridis, S. and Koutroumbas, K. (2008). *Pattern Recognition, Fourth Edition*. Academic Press, Inc., Orlando, FL, USA, 4th edition.
- [Theologou et al., 2015] Theologou, P., Pratikakis, I., and Theoharis, T. (2015). A comprehensive overview of methodologies and performance evaluation frameworks in 3d mesh segmentation. *Comput. Vis. Image Underst.*, 135(C):49–82.
- [Tremblay et al., 2018] Tremblay, J., Prakash, A., Acuna, D., Brophy, M., Jampani, V., Anil, C., To, T., Cameracci, E., Boochoon, S., and Birchfield, S. (2018). Training deep networks with synthetic data: Bridging the reality gap by domain randomization. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops*, pages 969–977.
- [Tutzauer et al., 2016a] Tutzauer, P., Becker, S., Fritsch, D., Niese, T., and Deussen, O. (2016a). A study of the human comprehension of building categories based on different 3d building representations. *Photogrammetrie-Fernerkundung-Geoinformation*, 2016(5-6):319–333.
- [Tutzauer et al., 2017] Tutzauer, P., Becker, S., and Haala, N. (2017). Perceptual rules for building enhancements in 3d virtual worlds. *i-com*, 16(3):205–213.
- [Tutzauer et al., 2016b] Tutzauer, P., Becker, S., Niese, T., Deussen, O., and Fritsch, D. (2016b). Understanding human perception of building categories in virtual 3d cities: a user study. In *XXIII ISPRS Congress, Commission II*, pages 683–687.
- [Tutzauer and Haala, 2015] Tutzauer, P. and Haala, N. (2015). Facade reconstruction using geometric and radiometric point cloud information. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 40(3):247.
- [Tutzauer and Haala, 2017] Tutzauer, P. and Haala, N. (2017). Processing of crawled urban imagery for building use classification. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 42.
- [Tutzauer et al., 2019] Tutzauer, P., Laupheimer, D., and Haala, N. (2019). Semantic urban mesh enhancement utilizing a hybrid model. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-2/W7:175–182.
- [Tyleček and Šára, 2013] Tyleček, R. and Šára, R. (2013). Spatial pattern templates for recognition of objects with regular structure. In *German Conference on Pattern Recognition*, pages 364–374. Springer.
- [Uijlings et al., 2013] Uijlings, J. R., Van De Sande, K. E., Gevers, T., and Smeulders, A. W. (2013). Selective search for object recognition. *International journal of computer vision*, 104(2):154–171.
- [Valentin et al., 2013] Valentin, J. P. C., Sengupta, S., Warrell, J., Shahrokni, A., and Torr, P. H. S. (2013). Mesh based semantic modelling for indoor and outdoor scenes. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2067–2074.
- [Von Ahn et al., 2008] Von Ahn, L., Maurer, B., McMillen, C., Abraham, D., and Blum, M. (2008). recaptcha: Human-based character recognition via web security measures. *Science*, 321(5895):1465–1468.

- [Weinmann et al., 2015] Weinmann, M., Jutzi, B., Hinz, S., and Mallet, C. (2015). Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. *ISPRS Journal of Photogrammetry and Remote Sensing*, 105:286 – 304.
- [Wenzel, 2016] Wenzel, S. (2016). *High-level facade image interpretation using marked point processes*. PhD thesis, Universitäts- und Landesbibliothek Bonn.
- [Wenzel et al., 2008] Wenzel, S., Drauschke, M., and Förstner, W. (2008). Detection of repeated structures in facade images. *Pattern Recognition and Image Analysis*, 18(3):406–411.
- [Wenzel and Förstner, 2016] Wenzel, S. and Förstner, W. (2016). Facade interpretation using a marked point process. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, 3:363.
- [Wertheimer, 1923] Wertheimer, M. (1923). Untersuchungen zur lehre von der gestalt. ii. *Psychological Research*, 4(1):301–350.
- [Wichmann et al., 2018] Wichmann, A., Agoub, A., and Kada, M. (2018). Roofn3d: Deep learning training data for 3d building reconstruction. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2:1191–1198.
- [Wonka et al., 2003] Wonka, P., Wimmer, M., Sillion, F., and Ribarsky, W. (2003). *Instant architecture*, volume 22. ACM.
- [Workman et al., 2017] Workman, S., Zhai, M., Crandall, D. J., and Jacobs, N. (2017). A Unified Model for Near and Remote Sensing.
- [Wu et al., 2015] Wu, Z., Song, S., Khosla, A., Yu, F., Zhang, L., Tang, X., and Xiao, J. (2015). 3d shapenets: A deep representation for volumetric shapes. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1912–1920.
- [Xiao et al., 2010] Xiao, J., Hays, J., Ehinger, K. A., Oliva, A., and Torralba, A. (2010). SUN database: Large-scale scene recognition from abbey to zoo. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 3485–3492.
- [Xu et al., 2017] Xu, Y., Hoegner, L., Tuttas, S., and Stilla, U. (2017). Voxel- and graph-based point cloud segmentation of 3d scenes using perceptual grouping laws. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 4.
- [Yi et al., 2017] Yi, L., Su, H., Guo, X., and Guibas, L. (2017). SyncSpecCNN: Synchronized spectral CNN for 3D shape segmentation. In *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, volume 2017-Janua, pages 6584–6592.
- [Yu and Koltun, 2015] Yu, F. and Koltun, V. (2015). Multi-scale context aggregation by dilated convolutions. *CoRR*, abs/1511.07122.
- [Yu et al., 2015] Yu, Q., Szegedy, C., Stumpe, M. C., Yatziv, L., Shet, V., Ibarz, J., and Arnoud, S. (2015). Large Scale Business Discovery from Street Level Imagery. *arXiv:1512.05430 [cs]*.
- [Yuan, 2018] Yuan, L. (2018). How cheap labor drives china’s a.i. ambitions. [Online; posted 25-November-2018].
- [Zhang et al., 2016] Zhang, L., Zhang, L., and Du, B. (2016). Deep Learning for Remote Sensing Data: A Technical Tutorial on the State of the Art. *IEEE Geoscience and Remote Sensing Magazine*, 4(2):22–40.
- [Zhang et al., 2017] Zhang, W., Huang, H., Schmitz, M., Sun, X., Wang, H., and Mayer, H. (2017). Effective Fusion of Multi-Modal Remote Sensing Data in a Fully Convolutional Network for Semantic Labeling. *Remote Sensing*, 10(2):52.
- [Zhao et al., 2017] Zhao, H., Shi, J., Qi, X., Wang, X., and Jia, J. (2017). Pyramid scene parsing network. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2881–2890.
- [Zhou, 2018] Zhou, Q. (2018). Pymesh. <https://github.com/PyMesh/PyMesh>.

Acknowledgements

I would like to thank the German Research Foundation (DFG) for financial support within the project D01 of SFB/Transregio 161.

My gratitude goes to all the people who have contributed to the creation of this work. First of all, I would like to thank Prof. Norbert Haala for giving me the opportunity to work at the Institute for Photogrammetry (ifp), his supervision, and for supporting me in every way in conducting this thesis.

I want to thank Prof. Helmut Mayer for agreeing to be co-referee, his time, and giving me valuable input beforehand. Moreover, I would like to thank Prof. Uwe Sörgel for acting as a second co-referee, especially on such short notice.

Thanks go to the supervisors within the SFB/Transregio 161, Prof. Dieter Fritsch and in particular Susanne Becker, with whom several publications have been written. Additionally, I am grateful to Till Niese and Dr. Marc Spicker for their contributions in the abstraction and stippling context.

My appreciation goes to all my colleagues at the ifp over the years. I enjoyed fruitful discussions, had successful collaborations and experienced an overall nice working atmosphere where all necessary tools were made available at any time. Special mention goes to Dominik Laupheimer, who greatly contributed to this work with his Master's thesis and was a co-author on several publications.

Last but not least I want to thank Diana Moll as well as my family for always being there. I appreciate your constant support and encouragement to finalize this thesis.

Curriculum Vitae

Personal

Name	Patrick Tutzauer
Date of birth	05.03.1988
Place of birth	Herrenberg, Germany

Education

2007-2013	Diploma, Geodesy and Geoinformatics, University of Stuttgart
1998-2007	Abitur, Andreae-Gymnasium Herrenberg

Experience

May 2012 - Jun 2013	Research Intern and Diploma Thesis at Robert Bosch GmbH, Leonberg
Oct 2017 - Dec 2017	Research Stay at the Department of Computer Science, University of Kentucky
Aug 2013 - Jun 2019	Research Associate, Institute for Photogrammetry, University of Stuttgart
Jul 2019 - Present	Software Engineer, nFrames GmbH, Stuttgart