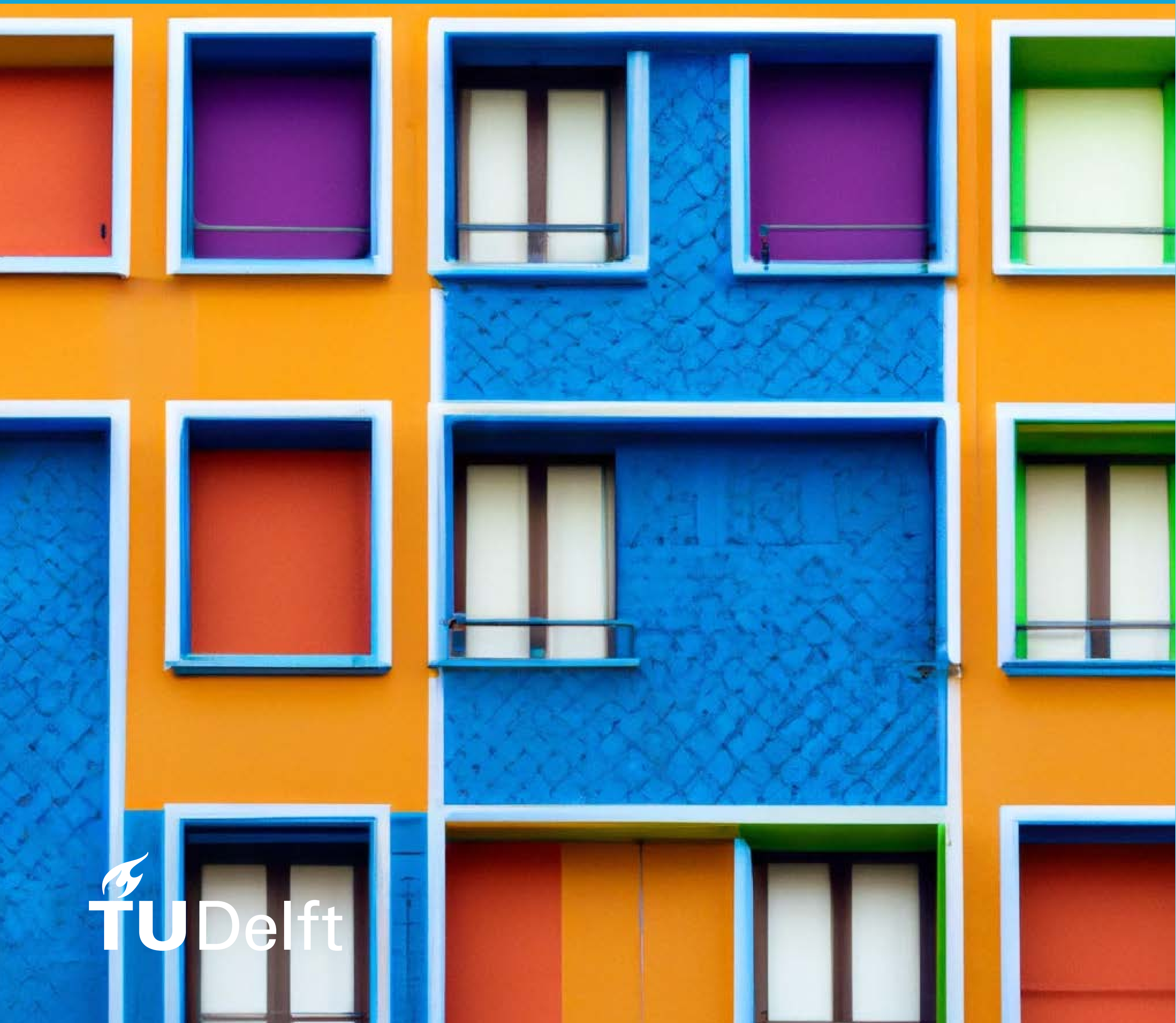
MSc thesis in Geomatics

Floor count from street view imagery using learning-based façade parsing

Daniël James Dobson

2023



TU Delft

MSc thesis in Geomatics

Floor count from street view imagery using learning-based façade parsing

Daniël James Dobson

January 2023

A thesis submitted to the Delft University of Technology in partial fulfillment of the requirements for the degree of Master of Science in Geomatics

Daniël James Dobson: *Floor count from street view imagery using learning-based façade parsing* (2023)

Cover image was created with the assistance of DALL·E 2

The work in this thesis was carried out in the:

3D geoinformation group
Delft University of Technology

Supervisors:	Dr. Ken Arroyo Ohori
	Nail Ibrahimli
Co-reader:	Dr. Hugo Ledoux

Abstract

Street view imagery (SVI) is one of the largest (growing) resources in urban analytics. A global close-up of the urban environment, if you will, which is rich in (untapped) information such as floor count. Floor count is useful in many applications, from improving energy consumption calculations to creation of 3D city models without elevation data. So far, efforts to extract floor count from SVI are mainly approached as a classification problem with the use of convolutional neural networks (CNNs). Limitations of this approach include the need of large (manually annotated) datasets, and uncertainty how these models learn to count storeys. Therefore, we aim to develop a method that can be trained on available datasets and determine floor count in a more explainable manner.

In order to make the floor count determination method more transparent, we mimic the row-wise counting of storeys as humans do: by vertically parsing a column of windows (and occasional door). Façade parsing is a common computer vision task that we can solve with deep learning. In this work, we employ the Mask R-CNN framework, that is trained on publicly available datasets, for the detection and segmentation of windows and doors. Then, the vertical distribution of detected/segmented windows and doors is estimated by computing the kernel density estimation function. The floor count is extracted by finding the number of maxima in the function, as the maxima represent the dense areas of windows and doors on a horizontal axis (i.e. storeys). To improve the results, an automatic image rectification is added as pre-processing step that enforces the regularity and repetitive occurrence of windows and doors. The full pipeline thus consists of three stages: 1) automatic image rectification, 2) window and door detection/segmentation with Mask R-CNN, 3) floor count estimation via maxima finding on the kernel density estimation (KDE) function. In addition, a small "wild" dataset was created that contains a higher variability in floor count, image quality and architectural styles, which better reflect real world SVI than existing façade datasets.

The floor count performance of the full pipeline was evaluated on the Amsterdam Facade (subset), ECP, eTRIMS and "wild SVI" datasets. Since floor count annotations were missing, these are manually added. For detection-based data, the best results are an accuracy of 83% and a mean absolute error (MAE) of 0.17. For normalised segmentation-based data, the best results are an accuracy of 80% and a MAE of 0.20. Considering the method is still at its infancy, the results are promising. With further improvements in the pipeline and addition of automatic façade acquisition, the approach can contribute in large scale extraction of floor count information from SVI. To encourage further development, the pipeline prototype, dataset and floor count annotations are open source and will be released on `https://github.com/Dobberzoon/Facade2Floorcount`.

Acknowledgements

My first words go to my mom, as this opportunity would not have existed without her. You have my eternal gratitude, and gave me strength to keep believing in myself. Lena, my love, thank you for giving your relentless support through it all. Floris, who kept climbing with me, even when I was “niet te genieten”.

Dear Ken, I was lucky to have you as my first mentor. You gave me guidance that I needed with the most spot-on feedback I could wish for, but foremost, the trust that I could do it. Thank you, sincerely. Nail, you were always there for me, bouncing ideas back and forth in our meetings. Also ensuring me, that I was not losing my mind, just doing research. Hugo, thank you for co-reading, helping me with errors big and small, and your humor.

Contents

1. Introduction **1**
1.1. Motivation . . . 1
1.2. Objectives . . . 2
1.3. Scope . . . 3
1.4. Thesis Outline . . . 3

2. Theoretical Background **5**
2.1. Regularity of façade elements . . . 5
2.2. Defining floor count . . . 7
2.3. Street View Imagery . . . 9
2.4. Image Rectification . . . 12
2.5. Deep learning . . . 14
2.5.1. Mask R-CNN . . . 17
2.6. Façade parsing . . . 22
2.6.1. Learning-based façade parsing . . . 23
2.7. Vertical clustering . . . 24
2.7.1. HDBSCAN . . . 24
2.7.2. Density estimation functions . . . 24
2.7.3. Floor count evaluation . . . 27

3. Related Work **29**
3.1. Floor count from elevation and attribute data . . . 29
3.2. Floor count from SVI as classification problem . . . 30
3.3. Floor line structures from SVI . . . 33

4. Experimental design and development **37**
4.1. Data preparation . . . 38
4.1.1. Floor count annotation . . . 38
4.1.2. Image rectification . . . 39
4.2. Façade parsing . . . 40
4.2.1. DeepFacade reproduction . . . 40
4.2.2. Mask R-CNN training and optimisation . . . 40
4.3. Floor count estimation . . . 42
4.3.1. FloorLevel-Net . . . 42
4.3.2. Bivariate approach . . . 43
4.3.3. Univariate approach . . . 44
4.3.4. Maxima finding . . . 46

5. Implementation and Experiments **49**
5.1. Software and tools . . . 49
5.2. Datasets . . . 49
5.3. Mask R-CNN training and optimisation experiments . . . 52

6. Results and Analysis **55**
6.1. Image rectification experiments . . . 55
6.1.1. VP estimation . . . 55
6.1.2. RANSAC and homography transform . . . 55
6.2. Façade parsing results . . . 56
6.2.1. Detection and segmentation evaluation . . . 57
6.3. Floor count estimation experiments . . . 59
6.3.1. FloorLevel-Net evaluation . . . 59
6.3.2. Bivariate vertical clustering evaluation . . . 59
6.3.3. Univariate vertical clustering evaluation . . . 60
6.3.4. Floor count performance evaluation . . . 61
6.3.5. Failure cases . . . 64

7. Conclusions **67**
7.1. Research overview . . . 67
7.2. Discussion . . . 69
7.2.1. Contributions . . . 69
7.2.2. Limitations . . . 69
7.2.3. Recommendation for future work . . . 70

A. Appendix **73**
A.1. Experimental design and development . . . 73

List of Figures

1.1. Increasing use of SVI in urban analytics. 1
1.2. Façade parsing example 2

2.1. Discovery of structural regularity in 3D geometry 5
2.2. Image-based procedural modeling of façades 6
2.3. A weak structure model for façades 6
2.4. Symmetry-aware façade parsing with occlusions. 7
2.5. A storey example in IFC model 7
2.6. Floor count definition and street view limitation 8
2.7. Floor count simplification 9
2.8. Distortion in street view image, caused by a raindrop 11
2.9. Stitching error and temporal difference 11
2.10. Automatic rectification of a stereo pair from a façade image 13
2.11. Automatic rectification on singular façade images 13
2.12. Image rectification by vertical and horizontal perspective transformation . . . 14
2.13. Different types of computer vision tasks 15
2.14. Interpolated precision-recall curve 17
2.15. Overview of COCO's evaluation metrics 17
2.16. A schematic overview of a typical CNN architecture. 18
2.17. The receptive field of one neuron 18
2.18. Intuitive explanation of feature learning 18
2.19. R-CNN architecture 19
2.20. Fast R-CNN architecture 20
2.21. Faster R-CNN 20
2.22. Mask R-CNN architecture 21
2.23. Simplified visualisation of Multi Task Network Cascades 22
2.24. DeepFacade 22
2.25. Façade parsing results of DeepWindows 23
2.26. DBSCAN 24
2.27. Bandwidth selection 25
2.28. Comparison of Silverman's rule of thumb and ISJ 26

3.1. CNN-based multiclassifitation approach to floor count determination 30
3.2. Confusion matrix of floor count determination performance ResNet-34 31
3.3. Architecture of TREncNet 32
3.4. Confusion matrix of floor count determination performance TREncNet 32
3.5. FLN architecture 33
3.6. Results of floor-level line detection with FloorLevel-Net 34
3.7. Populating object center horizontally with three equally spaced points 35
3.8. Floor segmentation with façade parsing, multi-RANSAC and object center population 35

4.1. A compact schematic overview of the proposed three-stage method 37

4.2. Flowchart that details the development steps within the data preparation stage. 38
4.3. Rectification enforcing regularity example 39
4.4. Flowchart that details the development steps within the façade parsing stage. 40
4.5. Flowchart that details the development steps within the floor count estimation stage. 42
4.6. HDBSCAN implementation 43
4.7. Point selection explanation 44
4.8. Point selection comparison 45
4.9. Bandwidth and kernel optimisation 45
4.10. Finding maxima in KDE 47

5.1. The car of the City of Amsterdam that is used to capture SVI from Amsterdam 50
5.2. Class imbalances in Amsterdam Facade dataset 50
5.3. Examples wild dataset 52
5.4. The full learning curve is plotted in manual hyperparameter tuning, with a LR of 0.0001 for 50 epochs 53
5.5. Sweep configuration chart that gives an overview of the hyperparameters searched 54

6.1. Image rectification results by VP estimation. 55
6.2. Image rectification results by direct homography transform. 56
6.3. Large distortion due to image rectification of large angle perspective view image 56
6.4. Mask R-CNN façade parsing examples 57
6.5. Results of the FloorLevel-Net reproduction 59
6.6. Results of the bivariate approach 60
6.7. Results of the univariate approach 61
6.8. The effect of automatic hyperparameter tuning, shown in a confusion matrix . 63
6.9. Graph visualisations of best results 64
6.10. Effect of normalisation of segmentation-based data on maxima finding 64
6.11. Undershooting problem of floor count estimation 65

A.1. The complete and detailed overview of the proposed three-staged method and developments steps 74

List of Tables

2.1. Mask R-CNN performance with different backbones 21

3.1. Performance floor count determination comparison of machine learning (ML) approach and geometric approach 29
3.2. A comparison of the image-based floor count classification works 33

5.1. Performance comparison of façade parsing model, manual versus automatic hyperparameter tuning 54
5.2. Comparison of the ResNet-50 FPN and ResNet-101 FPN backbone performance 54

6.1. Detection and segmentation performance on classes in terms of AP 57
6.2. Detection and segmentation performance across different detection sizes and thresholds 58
6.3. Floor count determination performance from the manually optimized façade parsing model 62
6.4. Floor count determination performance from the automatically optimized façade parsing model 62

Acronyms

GIS	geographical information system	1
SVI	street view imagery	1
GSV	Google Street View	1
OSM	Open Street Map	8
OGC	Open Geospatial Consortium	8
VSVI	volunteered street view imagery	9
VGI	volunteered geographic information	9
API	application programming interface	10
VP	vanishing point	13
mAP	mean average precision	15
AP	average precision	16
IoU	intersection over union	16
CNN	convolutional neural network	17
COCO	Common Objects in Context	17
ANN	artificial neural network	18
R-CNN	region-based CNN	19
RoIs	regions of interest	19
RoI	region of interest	19
FC	fully connected	19
FCN	fully convolutional network	20
MTL	multi task learning	21
KDE	kernel density estimation	24
FFTKDE	fast fourier transform kernel density estimation	25
DBSCAN	density-based spatial clustering of applications with noise	24
HDBSCAN	hierarchical density-based spatial clustering of applications with noise	24
FFT	fast fourier transform	25
ISJ	improved Sheather Jones	26
AMISE	asymptotic mean integrated square error	26
DEM	digital elevation model	29
MAE	mean absolute error	27
ME	mean error	27
FLN	FloorLevel-Net	33
LR	learning rate	41

1. Introduction

1.1. Motivation

Floor count information is important to geographical information systems (GISs) and urban analytics in many applications, as it improves how we model the built environment. For instance, having the information on the number of storeys (floors, levels) allows for the generation of a 3D city model from building footprints through extrusion [Biljecki, 2017]. Thus, extrusion is possible even when lacking elevation data, as is still the case for many areas in both developing and developed countries [Hartmann et al., 2016]. It can also help us concentrate our sustainability efforts to where it is needed most, as floor count improves energy modeling by enabling the computation of volume per building layer [private communication, Swinkels, 2023].

Even though floor count has potential to sharpen our insights on the entire built environment, cadastral datasets are often missing this crucial information. For example, we found that the cadastre of the municipality of Amsterdam is missing floor count information for 87.3% of its 196,881 buildings. Other problems raised by the Datateam of the municipality of Amsterdam include missing records on older buildings and unreliable acquisition on recent buildings [interview, Goes, 2022]. Recent work by Roy et al. [2022] shows it is possible to infer missing floor count information with a machine learning model that is trained on cadastral data. It achieved an accuracy of 94.50 % on buildings ≤ 5 storeys, and 52.3 % for buildings with > 5 storeys. It was trained on attribute data from 294,746 buildings, of which only 22,328 are open-source. This limits the reproducibility, thus we investigate the feasibility to infer the data from a different data source that has open-source or low cost alternatives: street view imagery (SVI).

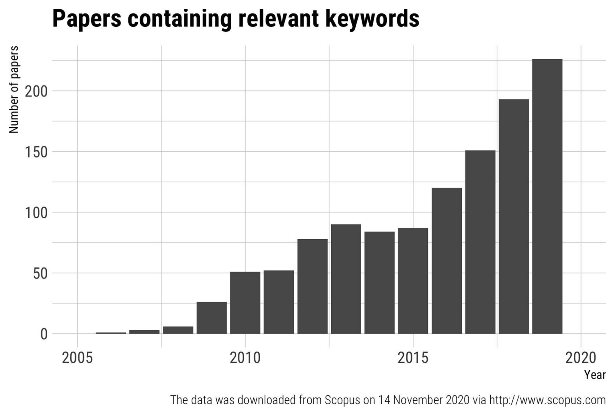


Figure 1.1.: The use of Street View Imagery in urban analytics and GIS research has steadily increased since the introduction of Google Street View in 2007. Adopted from [Biljecki and Ito, 2021].

Since the introduction of Google Street View (GSV) in 2007 ([Campkin and Ross, 2016; Gilge, 2016]), SVI grew to cover half of the world's population by 2017 ([Goel et al., 2018]).

At this pace, we will have global full global coverage sooner, rather than later. In parallel, the use of SVI in studies related to the built environment increased significantly (see Figure 1.1). Besides the growth of SVI coverage, increasing computing power, automation and advances in computer vision with deep learning have contributed to the rise of SVI in urban studies ([Biljecki and Ito, 2021]). What makes SVI interesting for the extraction of floor count (urban analyses), is that SVI can be captured regularly, cost effectively, and with unmatched perspective and level of detail on the urban environment. It can be regarded as a large scale close-up of our civilisation.

A common problem in computer vision that is approached with deep learning is façade parsing; the semantic identification of façade elements, e.g. windows, doors, balconies. In this work, we aim to extract floor count from SVI with façade parsing, as we recognise that the regularity in façade design implicitly contains the floor count information. In other words, we aim to count the number of storeys by using the repetitive occurrence of windows and doors.

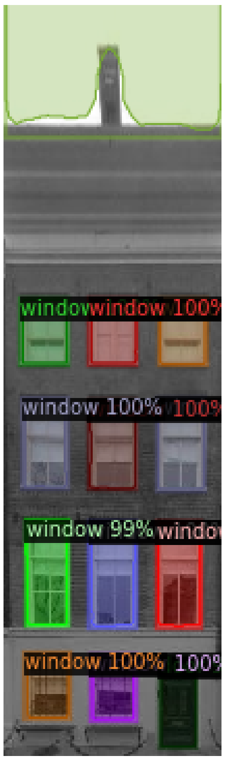


Figure 1.2.: Façade parsing example: identification and segmentation of windows and doors. The aim in this thesis is to count the number of storeys from the repetitive occurrence of windows and doors.

1.2. Objectives

Using SVI in urban analytics and GIS research becomes increasingly more interesting as its coverage grows. Since we can benefit from floor count information, even where no elevation data is available, and given that advances in CV with deep learning (e.g. learning-based façade parsing) could potentially uncover this information the main objective of this thesis is:

How to determine floor count in an image with the use of learning-based façade parsing?

A method has been developed that uses learning-based façade parsing to automatically infer floor count from SVI. The development process revealed several obstacles in the use of SVI and application of learning-based façade parsing, which raised the following subquestions:

- How can we use deep learning for façade parsing?
- How should façade parsing outputs be processed for floor count determination?
- How can we vertically group façade parsing outputs, such that the groups represent countable storeys?
- What pre- and/or post-processing steps do we need to improve floor count determination performance?

1.3. Scope

This thesis focuses on the method to automatically determine the floor count from an image, by the application of learning-based façade parsing and using its detection and segmentation outputs. This includes data-engineering of the input and output data for the deep learning model, employing, training and testing of different deep learning models, and various clustering related techniques.

Although some data from current SVI platforms will be acquired, this will be done manually for testing purposes. This thesis will not explore the automated method to acquire and extract individual building façades from a SVI database or platform, although observations from other work regarding the use of APIs will be noted. Also, building façades from SVI are originally panoramic omni-directional images. In order to extract single building façades from panoramic omni-directional images, equirectangular projection has to be performed, but is also not covered in this work. Lastly, the outputs of the floor count determination are not (automatically) stored in a subsequent or corresponding GIS dataset (e.g. footprint data).

1.4. Thesis Outline

In Chapter 2, the theoretical background describes the underlying principles for how regularity in façades can help determine floor count, define what floor count is and how image rectification enforces regularity. Then, deep learning concepts and learning-based façade parsing are introduced. Finally, the vertical clustering methods used for extracting floor count from façade parsing results are described. This will build the theoretical framework that bounds the methods used in this thesis and provides the concepts needed to understand the following chapters. Chapter 3 discusses related work in existing floor count determination research, and delineates what the limitations are that the proposed method overcomes. Then, the proposed method is described in detail in Chapter 4. Starting with an overview, the sections follow the order of the method's pipeline. The chapter is followed by the implementation details of the methodology in Chapter 5, describing datasets, programming details and how the experiments are conducted. The results of experiments are assessed and analysed in Chapter 6, providing a closer look into the learning-based façade parsing method, but mainly how the floor count determination performs. In addition, results and discussion are provided on methods that failed as well. Finally, the findings are discussed, main contributions are described, along with the limitations and future work in Chapter 7.

2. Theoretical Background

2.1. Regularity of façade elements

The phenomenon of symmetry in digital data is researched and used extensively in the fields of computer vision and computer graphics for over half a century [Liu et al., 2010]. A good overview of symmetry and (structural) façade analysis is given by Zhang et al. [2013]. Symmetry in façade designs is used extensively [Jiang et al., 2016], and the vertical symmetry is found to be more dominant compared to horizontal symmetry [Aydin and Mirzaei, 2020]. Since storeys are repeated vertical structures in buildings and their façades, a few examples are highlighted to show how symmetry and regularity are useful in the detection of storeys.

Symmetry detection is used by Pauly et al. [2008] to discover regular structures in point-based 3D models *without* prior knowledge of the size, shape and location of the underlying elements of the model. In the structure detection step of their pipeline the vertical regularity, represented by the horizontal lines in the lattice structures, corresponds to the occurrence of storeys (illustrated in Figure 2.1).

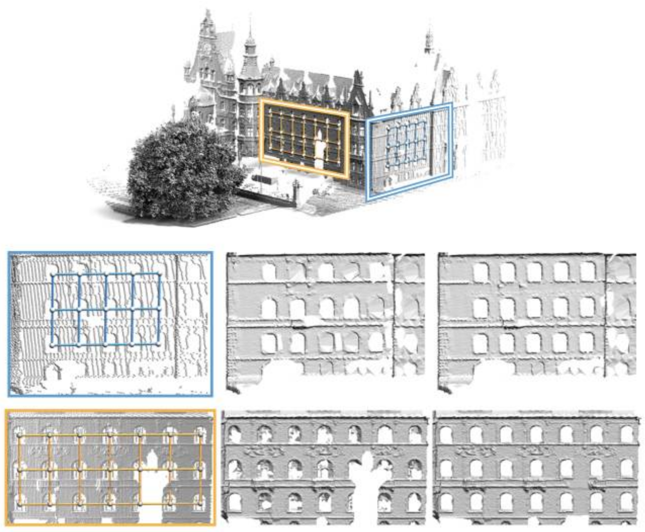


Figure 2.1.: The regularity of symmetry in architecture is used in the work of Pauly et al. [2008] to detect structure in this point-based model of a building. Albeit incomplete in this example, the vertical structure elements (horizontal lines in lattice structures) correspond to storeys, indicating regularity in façades can be used to extract floor count.

The image-based work of Müller et al. [2007] utilised the regularity of façade elements in façade structure detection, where translational symmetries that govern floors (storeys) and tiles are expected. The input is a single rectified image, and the output is a shape grammar rule set that can be used for the procedural modeling of 3D models. Uncovering the structural information could also be used to count the storeys, as one can observe in

Figure 2.2. However, the structure detection takes several minutes per image, which is not suitable for the large scale extraction of floor count information.

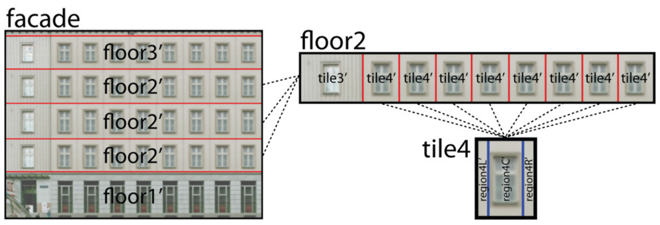


Figure 2.2.: Müller et al. [2007] used the symmetries that govern floors (storeys) and tiles to detect the façade structure in a single rectified image. Here, the floor numbers do not indicate the floor count, but the floor type that is detected. Still, the floor count information is implicitly present.

Tyleček and Šára [2010] assumed structural regularity of (façade) elements in terms of uniform distribution, alignment, spacing and configuration for the application of window detection. With a probabilistic approach of the embedded structure in the rectified image, the classifier knows "where to look". Especially the uniform distribution and alignment result in finding a (skeleton) structure that captures the underlying *storey* structures well, represented by the horizontal lines as illustrated in Figure 2.3. Given that the *weak* structure model still results in a number of horizontal lines that correspond with the number of storeys, indicates an approximation of the vertical distribution (of façade elements) can achieve floor count estimation.

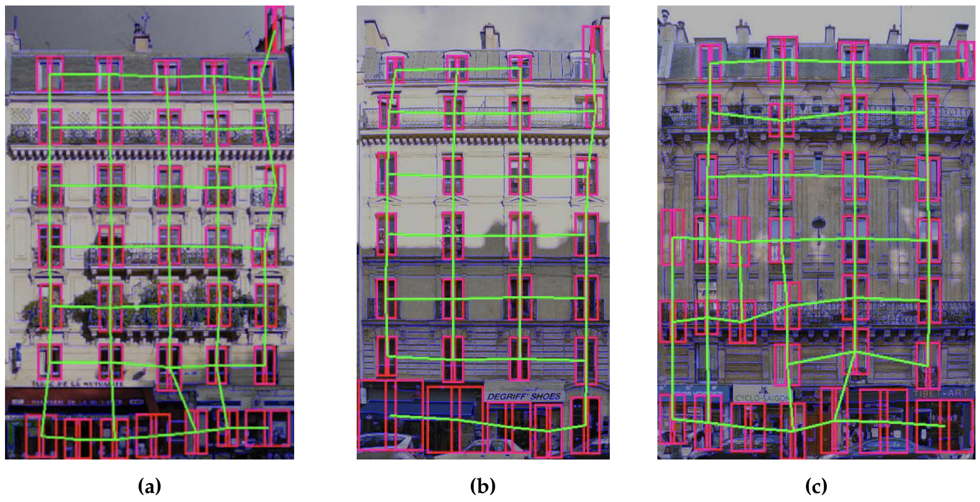


Figure 2.3.: A weak structure model for regular pattern recognition applied to (rectified) façade images by Tyleček and Šára [2010], for the application of window detection. A byproduct of the method is capturing of the underlying storey structures, represented by the green horizontal lines. Counting the horizontal lines corresponds to the counting of the number of storeys.

Cohen et al. [2017] leverages regularity information in a sequential optimization approach, for the segmentation of rectified façade images. The detection of symmetry improves the façade parsing results especially in occluded areas, recovering the underlying lattice structure. As illustrated in Figure 2.4, the vertical elements of the lattice structure correspond to the storeys, even when occluded. This indicates the regularity assumption can deliver robust global structural information, even when data or detections are incomplete.

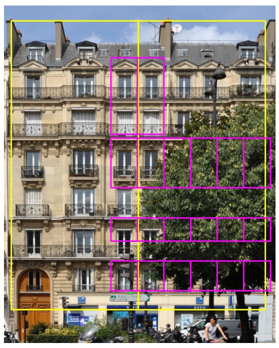


Figure 2.4.: Symmetry-aware façade parsing with occlusions [Cohen et al., 2017]. The use of regularity results in a robust understanding of the global structure, even when the façade image is (severely) occluded.

To summarise, previous work shows recognising symmetry in man-made objects and assuming regularity helps to recover the repeating structures (e.g. storeys) in façades more robustly. Although proven effective for façade parsing, this approach can be slow and does not fully make use of learning from data. Thus, a learning-based approach for façade parsing is applied in this work instead (described later in Section 2.6.1). The regularity assumption is still a strong precursor for the detection of the global structure of the façade, so the inverse of the regularity assumption will be used: we assume that façade parsing will result in elements (e.g. windows and doors) that occur in a regular manner.

2.2. Defining floor count

Basic definitions A storey is a level of a building, as illustrated in Figure 2.5. The number of storeys in a building is called the floor count, not to be confused with floor numbering. Storeys are counted as non-negative integers, and fractions are not allowed. In the Netherlands, a building level is included as (full) storey if the space is included as utility space (e.g. (ground) floor, basement or attic), is accessible to humans and has a net-height of 1.5 m or higher [Gemeente Amsterdam, 2007].

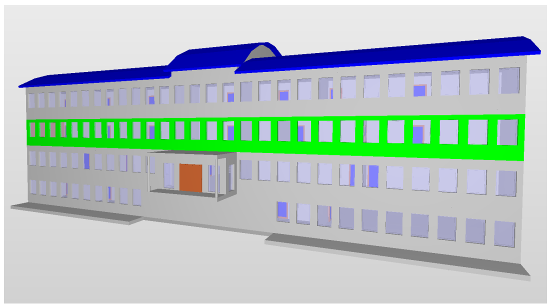


Figure 2.5.: A storey is a building level, example highlighted in green. IFC model AC11-Institute-Var-2-IFC from Laakso et al. [2012].

Attribute candidates In terms of data attribute for floor count, two candidates are considered from Open Street Map (OSM) and Open Geospatial Consortium (OGC)'s CityGML version 3.0. OSM defines `building:levels` (i.e. floor count) as the number of storeys above the ground level as non-negative fractions, excluding `roof:levels` (i.e. attics) from the count [OpenStreetMap Wiki, 2022]. The ground level is defined as the lowest entrance to a building. Since version 3.0 of the OGC CityGML standard, the notion of a storey is included [Kutzner et al., 2020]. Conveniently, the number of storeys is divided into the attributes `storeysAboveGround` and `storeysBelowGround`, stored as a non-negative integer.

Practical limitations In practice, we obtain the floor count from a (street view) image by counting the windows and/or doors vertically or row-wise. There are a number of limitations to this approach. First, negative storeys (e.g. basements) are in most cases invisible from the street view. Second, it is very challenging to assess whether an attic is accessible to humans and has a minimum net-height of 1.5 m from a (street view) image. Third, buildings on sloped terrain can result in an ambiguous number of storeys above ground level (see Figure 2.6) [Biljecki, 2017]. Fourth, there are exception cases that can look like separate storeys from the outside, such as semi-basements and mezzanines.

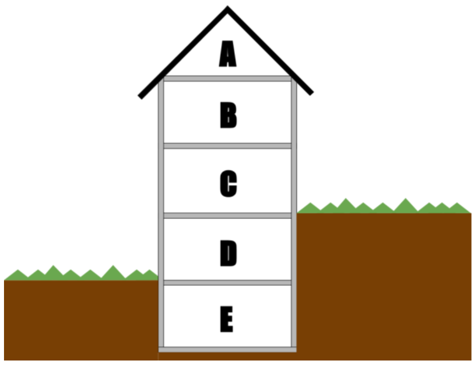


Figure 2.6.: In this work, floor count is defined as the number of full storeys above the ground level. The resulting floor count is stored as an attribute named `storeysAboveGround` [0..1]. A limitation of a street view image approach is exposed in the figure, as the floor count depends from which side the image is taken. When the street view is from the left, the floor count will include storeys A + B + C + D = 4. When the street view is from the right, the floor count will only include storeys A + B + C = 3 [OpenStreetMap Wiki, 2022].

The first limitation can be met with a compromise by only counting the storeys that are above the ground level, thus visible from the street view, and to store the value with an appropriate key. OGC's `storeysAboveGround` meets this requirement and will be used as key, the value will be stored as a non-negative integer as defined by OGC.

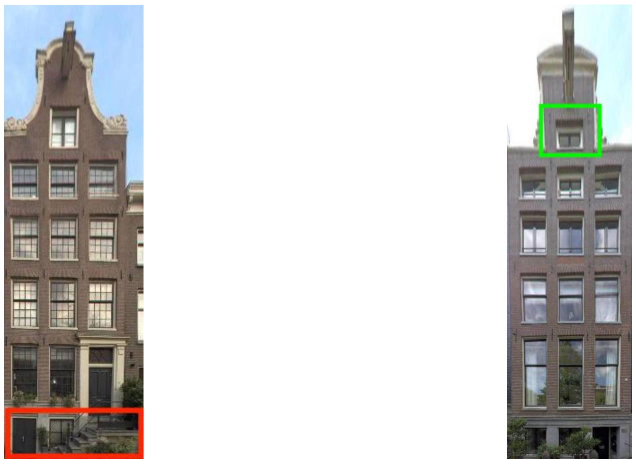
The second limitation makes the application of OSM's `building:levels` hard to implement, whereas OGC's `storeysAboveGround` does not specify whether or not to include attics. However, OGC's CityGML version 3.0 standard is extensive and allows for configuration of custom logic. The current approach is not capable of detecting roofs or different storey (types). This could be achieved in the future, and additional CityGML attributes can used for correction later on.

The third limitation could be solved by applying a rule that defines the ground level. OSM defines the ground level as the lowest entrance to a building. This means the ground level is D in Figure 2.6 and highest floor is B, resulting in a count of B + C + D = 3. However, since

SVI limits how many views we can obtain from a building, it is hard to tell whether or not the street view image contains the lowest entrance.

The fourth limitation is problematic with the approach of counting storeys with rows of windows and doors, since semi-basements have windows visible from the street view. A mezzanine is considered as a storey when it is a closed space, but the differentiation between a closed and open mezzanine is not visible from the street view.

Simplified problem In order to account for the second, third and fourth limitations, another compromise is made. We simplify the problem by regarding the first or lowest visible row of windows and/or doors as the ground floor, and ignore the concept roof or attic and count the last or highest visible row of windows and/or doors as a full storey. We use the `storeysAboveGround` attribute key, and store the value as a non-negative integer.



(a) Semi-basement. **(b)** Attic.

Figure 2.7.: The problem of which storey (types) to include in floor count is simplified to counting the total number of visible rows of windows and/or doors. We disregard the concepts of semi-basements or attics, as either are counted as a (full) storey.

2.3. Street View Imagery

Data type and sources As motivated in the introduction (Section 1.1), street view imagery (SVI) is the data type for façade images in current study. In our case, we are working with panoramic SVI data, from both commercial (Google Street View (GSV)) and open-source (Mapillary). Mapillary is a platform for volunteered street view imagery (VSVI), a form of volunteered geographic information (VGI), that is collected through crowdsourcing [Mahabir et al., 2020]. The challenge working with VSVI is that its data quality is heterogeneous [Hou and Biljecki, 2022]. We use a subset from Mapillary that is professionally created by the municipality of Amsterdam (see Section 5.2), hence the quality is homogeneous. A comprehensive review of SVI in urban analytics and GIS is given by Biljecki and Ito [2021].

Definition SVI In order to maintain a common understanding of SVI in research, we adopt the following definition of SVI:

> "Street view imagery (SVI) is typically a sequence of geotagged, ground level photographs taken along a trajectory, providing spatially continuous observation of its vicinity." [Hou and Biljecki, 2022, p. 6]

SVI quality In the context of façade parsing and large scale extraction of building assets or attributes, it is important to consider different quality aspects that affect the processing of SVI. Although SVI acquisition and equirectangular projection are outside of the scope of this research, these (pre)processes affect the quality of SVI data, thus the performance of subsequent methods that are applied and developed in current study. Therefore, it is important to mention how SVI data quality can be described and assessed. Hou and Biljecki [2022] developed an extensive framework to evaluate the quality of SVI data, that groups the quality aspects into the categories mentioned below:

1. Image quality
2. Metadata availability and accuracy
3. Spatial quality
4. Temporal quality
5. Logical consistency
6. Redundancy
7. Privacy

We focus on (the elements within) image quality, as image processing is of primary concern in this study. Different elements of image quality are encountered in the development of methods in this study, in manual creation of a custom SVI dataset, and in selection and processing of other datasets. In addition to quality aspects mentioned, practical issues found in image processing of SVI are observed in this work as well, of which a good overview is given by Gaw et al. [2022]. Coverage related issues are discussed, since these were found to be most limiting.

Image quality Image size, distortion, obstructions and stitching errors were the most encountered image quality issues. The issues are described in the relevant context of deep learning, façade parsing and other related works with panoramic SVI.

Image size and field of view (horizontal angle) have to be set upon query, when a SVI dataset is created with an application programming interface (API). The image size and field of view dictate the image shape and resolution, which can affect training a deep learning network. A common practice is to make the image size as small as possible, where the vision task at hand can still be done by a human. Larger image sizes can allow for more detail in vision tasks, but also increase training time (in deep learning). The image shape can introduce bias into the deep learning network as well, even if data augmentations (i.e. resizing images during training) are applied. Another aspect related to image size and resolution, is the scale of the image. Chen et al. [2022] generated a large SVI dataset of buildings, containing well over 780,000 buildings under GSV coverage. One of the reasons only 75% of that dataset resulted in suitable building (façade) images, is that buildings appeared too small in the image. We found a similar observation when creating a custom dataset (see Section 5.2), where the road (point of capture) is too far away from the building of interest. Using manual software (spherical image viewer), zooming in was possible to get a usable scale of the façade image.

Distortion can be caused by the wide angle in panoramic images or equirectangular projected images, or raindrops [Hou and Biljecki, 2022]. Distortions may also be introduced with further processing of images, such as image rectification. Especially when the angle in a perspective view image is large, the distortion after image rectification can be severe as illustrated in Figure 6.3.

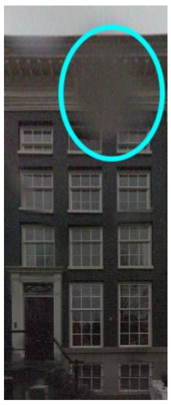
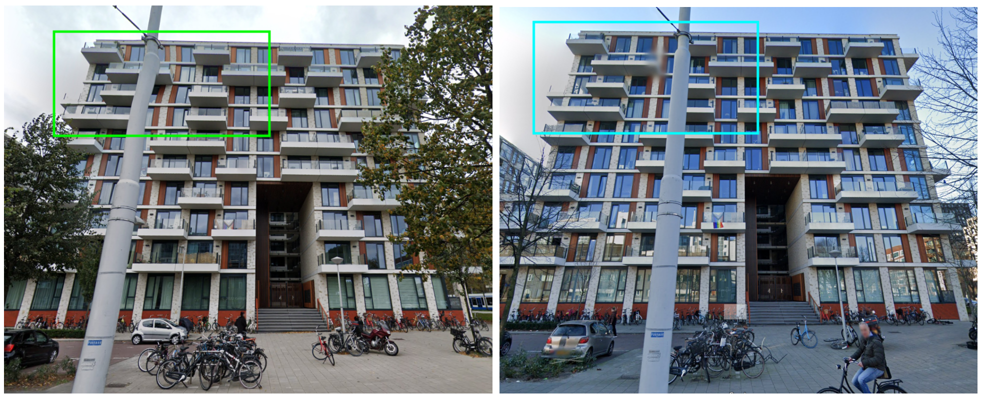


Figure 2.8.: An example of a significant distortion caused by a raindrop near the top of the image, highlighted in cyan.

Obstructions or occlusion of the image can be caused by (parking) cars, vegetation (e.g. trees), street signs etc. For the task of façade parsing, vegetation was the most limiting occlusion, then cars. Trees with leaves can occlude large parts of the façade and are permanent, cars have windows that may be wrongly detected as façade windows. Related to temporal quality, in GSV the capture time of the same location can be chosen e.g. to a different season where trees lack foliage or a moment where there is no car(s) blocking the façade of interest.



(a) Summer 2022. **(b)** Winter 2022.

Figure 2.9.: Difference of distortions present in the same façade from a GSV image, over two different seasons. Figure 2.9a was captured in June 2022 (summer) with the trees having foliage causing to partially occlude the façade, and has a stitching error highlighted in green. As Figure 2.9b was captured in March 2022 (winter), the trees lack foliage thus occluding the façade significantly less. The same area highlighted in cyan has no stitching error, although a small raindrop is introduced.

Stitching errors are rare in modern SVI datasets where a professional image capturing equipment is used, such as the one mentioned in Figure 5.1 or by GSV [Anguelov et al., 2010]. Still, stitching errors may occur and can result in an unnatural shape of the façade as illustrated in Figure 2.9.

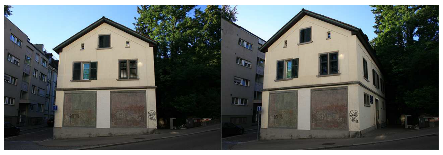
Coverage One of the most pressing issues in (large scale) SVI processing, is related to spatial and temporal SVI coverage [Gaw et al., 2022]. Other work concerning the large scale processing of SVI have found that poor coverage limits the ability to extract the number of storeys [Gaw et al., 2022], or results in unacceptable views [Chen et al., 2022]. Possible explanations are that some buildings might be contained in a building block, are on private properties where a street image mapping car is not allowed to access [Ning et al., 2022], or are simply not covered (yet).

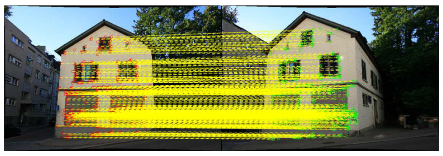
In conclusion, it is important to consider different quality aspects of SVI, when using or developing SVI datasets. For large scale façade parsing and image processing, image quality and coverage are the most important quality aspects to consider.

2.4. Image Rectification

Real world façade datasets acquired from an SVI source, potentially generated with an API, will mostly contain perspective view images from various angles. Regularity in the form of symmetry cannot be assumed for façade elements (i.e. windows and doors) along both horizontal and vertical axes in perspective view images, meaning we need to apply image rectification for the assumption to hold. We describe a couple of methods of image rectification relevant in the context of façade imagery and large scale image processing.

Tsironis et al. [2017] suggested the automatic rectification of façades from suitably configured calibrated stereo images, by the use of point matching of the planar object, camera calibration and either epipolar geometry or inter-image homography (illustrated in Figure 2.10). The approach is dependent on a stereo pair and the configuration thereof, and knowledge of interior camera orientation parameters. The necessity of a stereo pair doubles the dataset size, which is especially problematic with an eventual large scale application in mind. The stereo pair configuration refers to position, rotation and orientation of each stereo image, which need to be within certain boundaries to achieve good results. As the sequences of SVI are captured automatically, at different driver speeds, different road-façade distances, and different obstructions at different angles, obtaining consistent and matching stereo pairs is challenging. In the case of crowdsourced VSVI, camera calibration or knowledge of camera parameters is often not available. With the intention to minimize dataset size and having VSVI as a future option, automatic rectification of singular images without the need for camera calibration or knowledge of camera parameters is a more suitable approach.


(a)


(b)

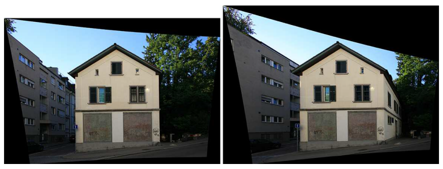

(c)

Figure 2.10.: Automatic rectification of a stereo pair from a façade image, by the use of point matching. In 2.10a are the suitably configured input images, in 2.10b the result of point matching, and in 2.10c the end result of the automatic rectification [Tsironis et al., 2017].

Liu [2011] performed automatic rectification on singular façade images by using RANSAC on line segment features followed by vanishing point (VP) estimation, VPs are then used to force dominant lines to become parallel by horizontally and vertically warping the image. However, this transform only resolves affine transformations and even when VP estimations are only slightly off, final rectifications are significantly affected [Affara et al., 2016]. Furthermore, Liu [2011]'s method is sensitive to scale which can be problematic with the use of different SVI/VGI datasets as scaling could vary among them. Wu et al. [2010] added a VP-refinement step to recover orthogonality and improve VP estimations, though is computationally expensive [Affara et al., 2016].

The automatic and efficient image rectification method by Affara et al. [2016] works on singular façade images, by finding the most prevalent horizontal and vertical line segments with RANSAC, followed by homography transformation. Horizontal and vertical line segments naturally reside in objects that are rectangular in the real world, such as window and door frames (see Figure 2.11).

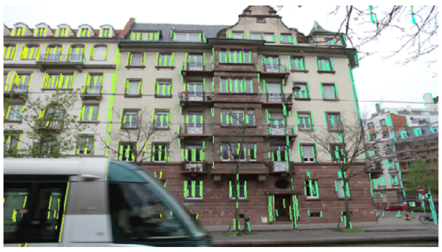


Figure 2.11.: Automatic rectification on singular façade images, that works by finding most prevalent horizontal and vertical line segments, which naturally reside in objects that are rectangular in the real world (e.g. windows and doors). As a result of the rectification, the lines in windows and doors are forced to be parallel, ensuring the applicability of the regularity assumption [Affara et al., 2016].

As a result of the rectification, the most prevalent rectangular elements (i.e. windows and doors) are structured repetitively along horizontal and *vertical* axes [Affara et al., 2016]. As such, the applicability of the regularity assumption is enforced.

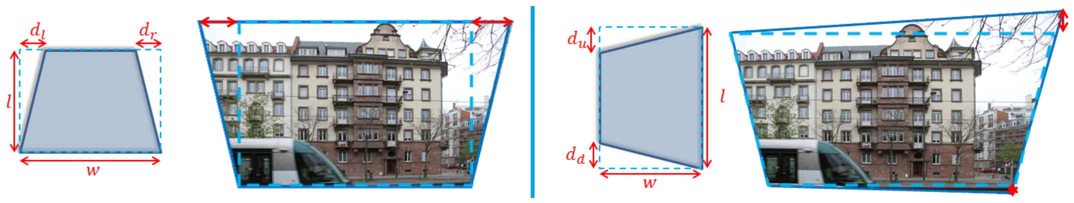


Figure 2.12.: Image rectification by applying horizontal (left) and vertical (right) perspective transformation on a façade image [Affara et al., 2016].

The biggest difference is that the method of Affara et al. [2016] does not require VP estimation. Instead, the best homography is found that aligns vanishing line segments horizontally and vertically, by decomposition of full transformation $\boldsymbol{H}$ into the combination of two simpler transformations of vertical perspective $\boldsymbol{H_v}$ and horizontal perspective $\boldsymbol{H_h}$ as formulated in Equation 2.1.

$$H = H_v H_h \tag{2.1}$$

As illustrated in Figure 2.12, the parameters for horizontal (d_l, d_r) and vertical (d_u, d_d) shifts in image corners, along with corrections in the width w and length l of the image are used to define the matrices $\boldsymbol{H_v}$ and $\boldsymbol{H_h}$, formulated in Equation 2.2 and 2.3, respectively.

$$H_v = \begin{bmatrix} 1 + \frac{d_r - d_l}{w} & \frac{-d_l}{l} & d_l \\ 0 & 1 + \frac{d_r - d_l}{w} & 0 \\ 0 & \frac{d_l - d_r}{wl} & 1 \end{bmatrix} \tag{2.2}$$

$$H_h = \begin{bmatrix} 1 + \frac{d_d - d_u}{w} & 0 & 0 \\ \frac{-d_u}{w} & 1 + \frac{d_d - d_u}{l} & d_u \\ \frac{d_d - d_u}{wl} & 0 & 1 \end{bmatrix} \tag{2.3}$$

The shift parameters are found by filtering line segments' orientations on a vertical and horizontal linearity constraint using RANSAC, and transform the filtered line segments into a vertical and horizontal by solving a minimization problem for (d_l, d_r) and (d_u, d_d), respectively.

To summarise, it is important to harmonise the input data characteristics and requirements of the intended final application. As neither of the mentioned methods are explicitly evaluated on SVI, it is of interest to study the performance on real world SVI with varying image quality and characteristics. For the large scale image rectification of SVI/VSVI, an efficient, automatic image rectification method for singular images by Affara et al. [2016] promises a suitable method.

2.5. Deep learning

Since we are interested in the application of learning-based façade parsing, it is important to understand what deep learning and its related concepts are. Façade parsing, an important task in the field of computer vision, is approached as different vision tasks: as object detection and instance segmentation.

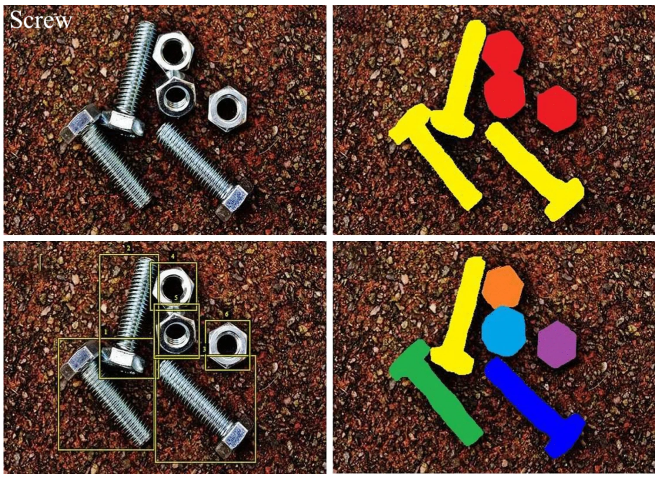


Figure 2.13.: Different types of computer vision tasks, from coarse to finer-grained. Top row: classification and semantic segmentation. Bottom row: object detection and instance segmentation. Adapted from Sharma et al. [2022].

Object detection is the classification and localisation of individual objects using bounding boxes, illustrated in the bottom left of Figure 2.13. Object detection outputs are interesting since it allows for intraclass differentiation. Intraclass differentiation is the ability to discern different instances of the same class, which is useful in counting the number of individual instances within a certain class.

Semantic segmentation is the classification of each pixel. The output is a single segmentation mask without intraclass differentiation, illustrated in top right of Figure 2.13. This makes the output unsuitable for counting object within a certain class, without further image processing.

Object instance segmentation or simply instance segmentation combines both approaches, i.e. the classification of each pixel with intraclass differentiation. This is illustrated in bottom right of Figure 2.13, where each screw is identified with a different colour. An important difference between object detection and instance segmentation regarding instances, is that the former allows overlap and the latter does not. This could influence the number of countable (clusters of) instances.

Evaluation metrics The most common metric to measure an object detector's performance is the mean average precision (mAP) metric. The numerical metric helps understanding the detector's performance in terms of both precision and recall over different confidence thresholds. Precision and recall can be useful to measure prediction performance, especially when classes in the dataset are (very) imbalanced [Kramer, 2016]. The metrics are calculated based on assessing each detection as true positive (TP), false positive (FP), true negative (TN) and false negative (FN).

Precision measures the proportion of total number positive predictions that are correctly identified as positive detections, and is defined as:

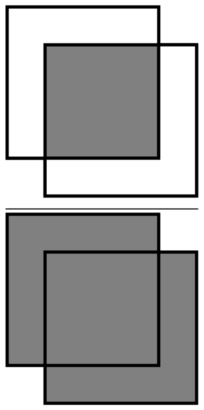
$$precision = \frac{TP}{TP + FP} \tag{2.4}$$

Recall measures the proportion of total number of ground-truths that are correctly identified as positive detections, and is defined as:

$$recall = \frac{TP}{TP + FN} \tag{2.5}$$

IoU Whether a prediction is taken into account, is dependent on confidence and intersection over union (IoU) threshold. Confidence score or "objectness" score is the likelihood an anchor box contains an object of a certain class. IoU is defined as:

$$IoU = \frac{\text{area of overlap}}{\text{area of union}} = \tag{2.6}$$



In other words, confidence score measures if an object is detected and IoU measures the correctness of location. Lower (score) thresholds result in more detections thus increasing recall, whereas higher thresholds result in more accurate predictions at the expense of recall. The relation between precision and recall can be expressed with average precision (AP) as single scalar value, i.e. the precision averaged over all levels of recall of a certain class. average precision (AP) is computed by the area under the interpolated precision-recall curve [Zeng, 2018], illustrated in Figure 2.14.

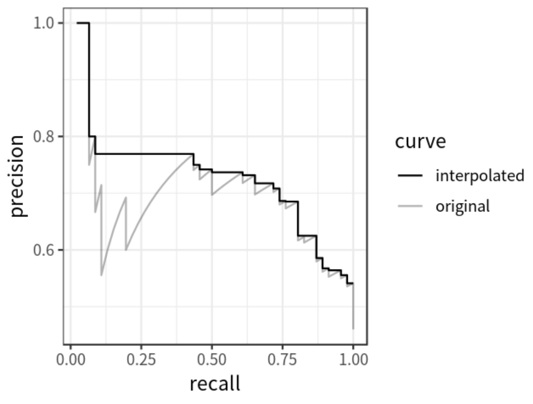
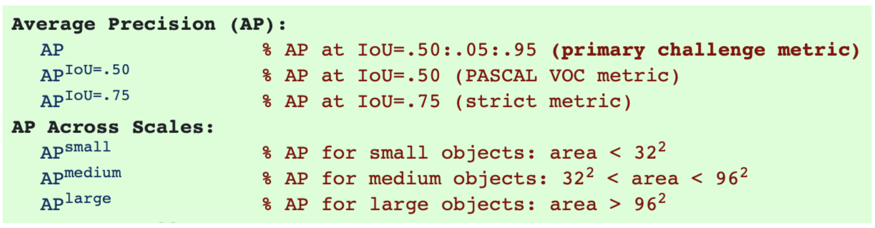


Figure 2.14.: The AP is calculated by the area under the interpolated precision-recall curve, which is the precision averaged over all unique recall levels [Zeng, 2018].

Traditionally, AP averaged over all classes is referred to as mAP. However, most object detection and semantic segmentation related research partake in computer vision challenges such as the Common Objects in Context (COCO) challenge [Lin et al., 2014]. The COCO challenge has defined their own metric, which we adopt for comparability. The COCO definition makes no distinction between AP and mAP, as AP is defined as AP averaged over 10 IoU values [0.50:0.05:0.95] [Lin et al., 2014]. As such, when referred to AP it is equivalent to mAP. Furthermore, AP^{small}, AP^{medium} and AP^{large} are interesting metrics to understand the detector's ability to detect different object sizes. Refer to Figure 2.15 for an overview of used COCO metrics.



```
Average Precision (AP):
   AP                     % AP at IoU=.50:.05:.95 (primary challenge metric)
   AP^IoU=.50             % AP at IoU=.50 (PASCAL VOC metric)
   AP^IoU=.75             % AP at IoU=.75 (strict metric)
AP Across Scales:
   AP^small               % AP for small objects: area < 32^2
   AP^medium              % AP for medium objects: 32^2 < area < 96^2
   AP^large               % AP for large objects: area > 96^2
```

Figure 2.15.: Overview of COCO's evaluation metrics and their corresponding definitions [Lin et al., 2014].

2.5.1. Mask R-CNN

For the objective of counting storeys, it is interesting to explore processing the outputs of both object detection and instance segmentation. One framework that allows both object detection and instance segmentation simultaneously, and proven effective in the application of façade parsing (see Section 2.6.1), is Mask R-CNN. In order to understand how Mask R-CNN works, a high-level overview of its underlying components is given.

Convolutional neural network (CNN) is the fundament of each method Mask R-CNN builds upon, which is an algorithm used for pattern recognition primarily in image data [O'Shea and Nash, 2015]. A conventional application is an image classifier, as illustrated in top left of Figure 2.13.

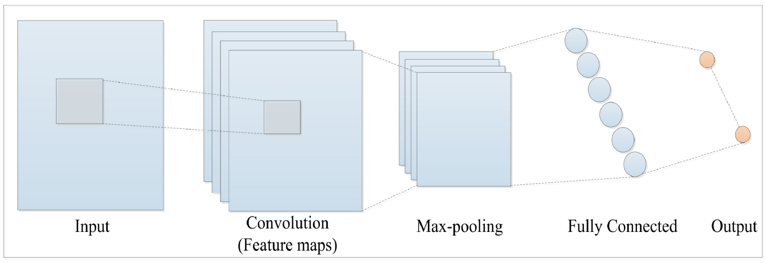


Figure 2.16.: A schematic overview of a typical CNN architecture. The three main layers of CNNs are; 1. Convolution layer, 2. Pooling layer and 3. Fully-connected layer. The layers help reduce computational complexity and overfitting of regular ANNs when processing high dimensional vector data such as image data Hussain et al. [2018].

A CNN is similar to a regular artificial neural network (ANN) being a collection of neurons that is structured as interconnected layers, where each neuron has learnable weights and biases. NNs are trained using stochastic gradient descent and are optimized by minimizing a cost function (loss), and back propagation allows for the weights to be updated during training. ANNs have the limitations of computational complexity and overfitting [O'Shea and Nash, 2015], especially when processing high dimensional vector data (i.e. image data). A CNN reduces the number of trainable parameters by a series of convolution and pooling layers. A convolution is the application of a kernel or filter on an input that leads to an activation. As such, a neuron is only connected to a small region of the previous layer through the receptive field of the filter (see Figure 2.17).

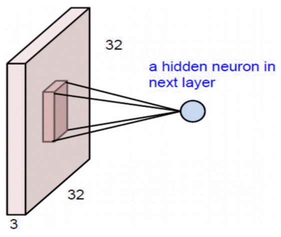


Figure 2.17.: The receptive field of one neuron (pixel) in the next convolution layer [Albawi et al., 2017].

The result is an activation map of feature representations. A simplified explanation can be illustrated with face detection in Figure 2.18, the first convolutions may detect edges or corners, later convolutions detect eyes or noses and final convolutions detect whole faces.

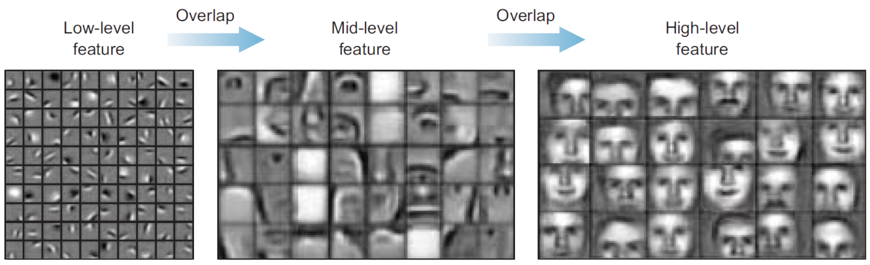


Figure 2.18.: A simplified explanation how a series of convolutions detects low-level features (e.g. lines and corners) in early layers to high-level features (e.g. faces) in final layers [Elgendy, 2020].

The pooling layer helps reducing the number of trainable parameters by reducing the spatial size of the activation maps. Usually, max-pooling is applied in CNNs where a small kernel (e.g. 2×2 with a stride of 2) applies a MAX-function, i.e. only the maximum value of the kernel's receptive field is mapped. Finally, a fully-connected layer with a loss function at the end outputs class scores as a regular ANN.

A drawback of the CNN is that spatial information is lost by convolving and pooling the input image, creating a localisation problem. Without localisation it is not possible to discern different objects in an image, and therefore unsuitable for counting objects (e.g. storeys).

Region-based CNN (R-CNN) solves CNN's localisation problem by employing (class independent) region proposals, creating object candidates that are independently classified by CNNs resulting in classed bounding boxes, i.e. object detection as illustrated in Figure 2.19. Another important contribution of the work of Girshick et al. [2014] is "supervised pre-training/domain-specific fine-tuning", also known as transfer learning.

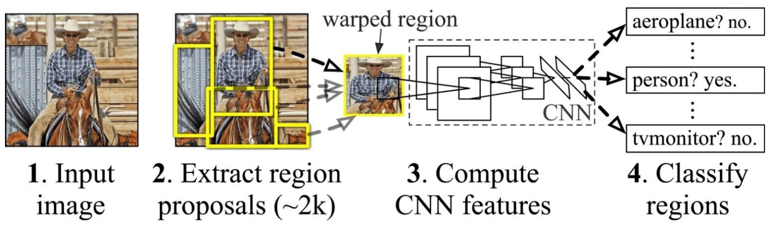


Figure 2.19.: The R-CNN architecture solved the lack of localisation in CNNs by achieving object detection through region proposals [Girshick et al., 2014].

Two limitations of R-CNNs are related to the region proposals, causing R-CNN-based architectures to suffer in speed, accuracy, or simplicity [Girshick, 2015]. First, many region proposals are extracted using "selective search" per input image and fed forward to the CNN. The original work of Girshick et al. [2014] extracted around 2000 region proposals per image. Second, the coarse localisation provided by the region proposals have to be processed through a refinement stage in order to attain precise localisation.

Fast R-CNN resolves the speed bottleneck in R-CNN, by simplifying the architecture. The simplification can be observed by comparing Figure 2.19 and 2.20. Instead of having a multi-stage architecture where many proposals are fed independently into a CNN , multiple regions of interest (RoIs) are fed to a CNN at once allowing the CNN to share computations. From the feature map, each region of interest (RoI) is pooled into a small fixed-size feature map as necessary for the fully connected (FC) layers it is fed to. The two FC layers perform both classification and bounding-box regression on the feature map simultaneously. This multi-task approach does not only improve speed, but the shared representation also improves accuracy [Girshick, 2015].

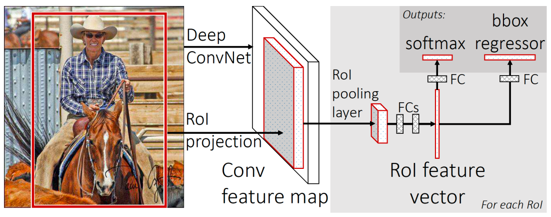


Figure 2.20.: Fast R-CNN simplified the architecture to resolve the speed bottleneck in R-CNN. RoIs are fed to a CNN at once, allowing the CNN to share computations [Girshick, 2015].

Yet, the same non-learning "selective search" algorithm is performed for finding RoIs, and is the remaining bottleneck in Fast R-CNN [Maity et al., 2021].

Faster R-CNN achieved near real-time object detection, by replacing the selective search algorithm for finding RoIs with the learning attention-module named Region Proposal Network (RPN) [Maity et al., 2021]. The RPN is a fully convolutional network (FCN) that saves computation by sharing its feature map with the detection network, essentially localising and classifying objects concurrently (see Figure 2.21). The Faster R-CNN framework allows for efficient object detection, allowing intraclass differentiation which is useful for counting instances. However, this does not leave the possibility to explore the counting of instances with pixel-level precision.

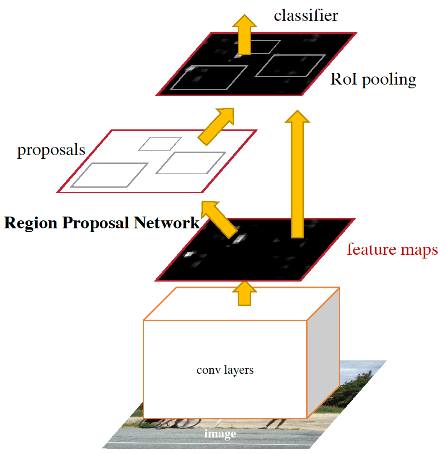


Figure 2.21.: Faster R-CNN improved Fast R-CNN by replacing the selective search algorithm with the learning attention-module RPN to find RoIs [Ren et al., 2015].

Mask R-CNN is a simple and flexible framework for object instance segmentation, developed by Meta AI Research, formerly known as Facebook AI Research (FAIR) [He et al., 2017]. It is an evolution of the Faster R-CNN framework for object detection [Maity et al., 2021]. The main contribution is the addition of a branch that predicts a segmentation mask for each object candidate, in addition to a bounding box and class label. The result is two outputs for each region of interest: a classed bounding box and an instance segmentation mask, as illustrated in Figure 2.22. Due to the two outputs, the Mask R-CNN is a two-stage network.

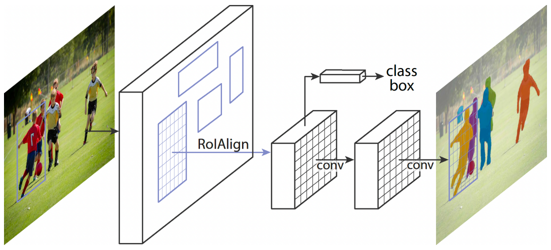


Figure 2.22.: Mask R-CNN added a branch to Faster R-CNN for the prediction of a segmentation mask per detection [He et al., 2017].

Backbone Mask R-CNN can instantiated with multiple architectures as backbone. The convolutional backbone architecture, is used for feature extraction over a whole image [He et al., 2017]. The backbone follows the *network-depth-features* naming structure. In the original paper, ResNet [He et al., 2016] and ResNeXt [Xie et al., 2017] are evaluated with network depths of 50 or 101 [He et al., 2017]. For features, the Feature Pyramid Network (FPN) [Lin et al., 2017] gives outstanding performance in terms of accuracy and speed according to He et al. [2017]. ResNet50-FPN en ResNet101-FPN are opted since these were evaluated on the Amsterdam Facade training dataset used in this work by Eijgenstein [2021], achieving good results in terms of AP in both object detection and instance segmentation as shown in Table 2.1.

Table 2.1.: Mask R-CNN object detection and instance segmentation results with ResNet50-FPN and ResNet101-FPN backbones on Amsterdam Facade dataset, by Eijgenstein [2021].

Backbone	AP^{bbox}	AP^{segm}
ResNet50-FPN	76.57	77.97
ResNet101-FPN	75.94	77.96

Loss functions The two stage Mask R-CNN has five loss functions. As a reminder, the first stage is Faster R-CNN's RPN with corresponding RPN classification and RPN bounding-box loss functions. The second stage adds three multi-task losses to the head of the added branch, for classification, bounding-box and mask defined as $L = L_{cls} + L_{box} + L_{mask}$ [He et al., 2017]. For the interest of gaining good performance on both object detection and instance segmentation, our implementation of Mask R-CNN is trained on minimizing the total loss which is the weighted sum of the five losses combined.

Multi task learning (MTL) In MTL a certain task in the final feature extraction is dependent on the context of other features. Say the first task is object detection, the second task is semantic segmentation of each object (i.e. Mask R-CNN) and the third task is detection of floor count based on the second task. The detection of floor count is called a downstream task as it follows from the first two and is the actual task of interest (see Fig. 2.23).

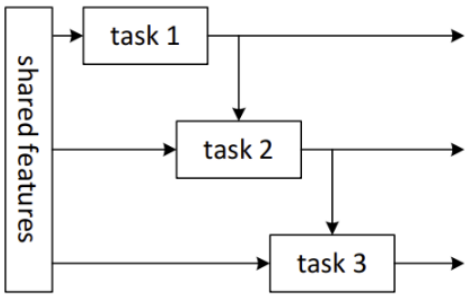


Figure 2.23.: Simplified visualisation of Multi Task Network Cascades [Dai et al., 2016].

2.6. Façade parsing

Façade parsing is referred to the semantic segmentation of a façade image [Mathias et al., 2016], into façade elements such as window, door, balcony, and so on [Liu et al., 2017]. Related work to façade parsing can be categorized into three principal strategies: grammar-based, image processing and learning-based [Sezen et al., 2022].

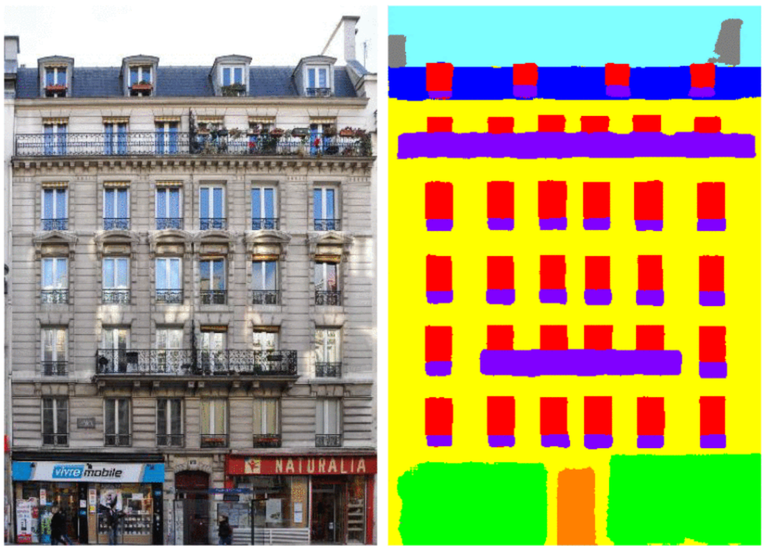


Figure 2.24.: DeepFacade: An influential work of Liu et al. [2020], that applies a learning-based approach to façade parsing which is treated as a semantic segmentation problem.

Grammar-based works assume prior knowledge: a façade adheres to a regular layout of rectangular shaped objects, organised by a set of man-made (grammar) rules [Teboul et al., 2011; Wang et al., 2022]. As the grammar-rules are constructed from certain (non-exhaustive set of) architectural styles, the approach does not generalize well and fails if the prior knowledge does not apply [Fathalla and Vogiatzis, 2017; Liu et al., 2017].

Image processing or basic computer vision related to pattern recognition (e.g. corner/edge detection) is dependent on local image values and sensitive to noise [Kong and Fan, 2020].

2.6.1. Learning-based façade parsing

Learning-based façade parsing does not rely solely on man-made rules of repetitive pattern recognition, but applies deep learning techniques to learn from data. Learning-based approaches are widely adopted to façade parsing in recent years for detection and segmentation of façade elements.

Schmitz and Mayer [2016] used a CNN for the semantic segmentation of façades yielding state-of-the-art performance on the eTRIMS dataset, without incorporation of any prior knowledge to the model. Liu et al. [2020] incorporated prior knowledge (i.e. rectangular symmetry of windows, doors and balconies) into the learning process with a novel symmetric loss function, such that their FCN penalized non-rectangular shapes when learning from the input. Their work, named DeepFacade, is still referred to as the state-of-the-art in recent papers [Zhang et al., 2022]. DeepFacade is a combination of FCN as main segmentation network with Mask R-CNN as its detector. For comparison, Mask R-CNN is also evaluated alongside the main segmentation network for semantic segmentation. Liu et al. [2020] reports 93.9% and 96.7% accuracy on classes of interest window and door with Mask R-CNN, respectively. Unsurprisingly, Mask R-CNN (with and without modifications) has been used as main façade parsing method in other works. Including extraction of façade details (i.e. windows and doors) for addition to LoD2 CityGML models [Zhang et al., 2019], window detection [Nordmark, 2021; Sun et al., 2022], large-scale façade parsing [Ayenew, 2021],and as façade parsing stage for window, door and sky object instance segmentation on the Amsterdam Facade dataset in Eijgenstein [2021].

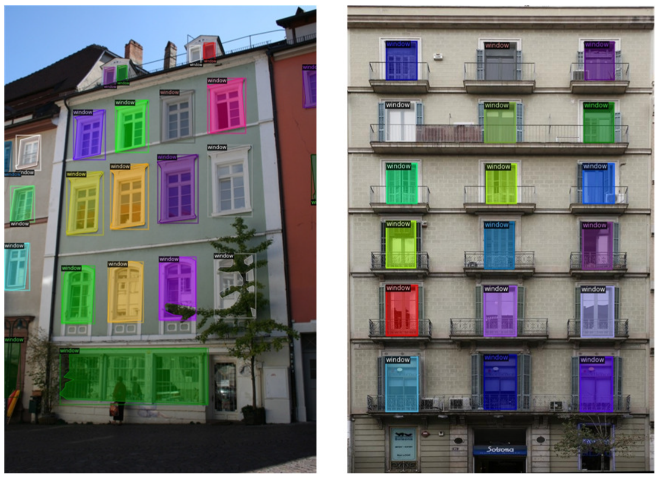


Figure 2.25.: Façade parsing results of DeepWindows [Sun et al., 2022]. A limiting factor for the determination of floor count would be the lack of door detection. In a few cases, the ground floor has a door but no windows.

Considering Mask R-CNN compares well to the state-of-the-art today, proven in the detection/segmentation of windows and doors, and widely available through open-source tooling, thus Mask R-CNN is the framework for both object detection as instance segmentation of windows and doors in this thesis.

2.7. Vertical clustering

Façade parsing yields detections and segmented instances, in our case of windows and doors for each image. The goal is to cluster these detections and segmented instances vertically, such that each cluster represents a storey.

2.7.1. HDBSCAN

In earlier experiments, façade parsing detections are clustered using hierarchical density-based spatial clustering of applications with noise (HDBSCAN). Regular density-based spatial clustering of applications with noise (DBSCAN) [Ester et al., 1996] is one of the most used algorithms for clustering. It works by finding areas that have a higher density (points per area) than surrounding points [Ledoux et al., 2022]. The density can be tuned with two defining parameters: ϵ radius and n_{min} minimum number of cluster points. The aim is to group areas that have close neighbors, as illustrated in Figure 2.26.

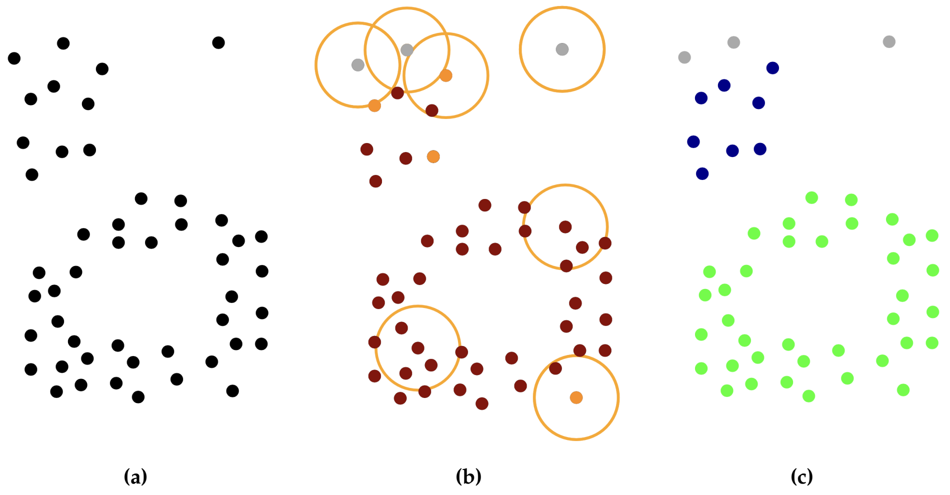


Figure 2.26.: DBSCAN for clustering points. a) The initial set of points. b) DBSCAN finds density clusters by recursively checking neighboring points if they are within the search radius ϵ. c) The resulting clusters of running DBSCAN [Ledoux et al., 2022].

Since the radius ϵ and minimum cluster size n_{min} have to be set, the method needs tuning for good results. This is not preferable in large scale analysis, with varying densities among samples. HDBSCAN performs DBSCAN with more stability by varying ϵ values and incorporating the result, meaning clusters of varying densities can be found [McInnes et al., 2017]. The method can be used fully automatic, which is useful in large scale analysis of varying densities.

2.7.2. Density estimation functions

KDE Based on the regularity assumption of façade elements and the observation that windows and doors are structured repetitively in vertical direction, the floor count is determined by estimation the underlying vertical (y) data distribution. A non-parametric approach to the problem is taken by performing a univariate analysis, and application of kernel density estimation (KDE) to estimate the probability density function $\hat{f}(x)$ at ordinate x. The problem is defined as follows [Silverman, 2018]:

> **Problem** Given n data points $\{x_1, ..., x_n\}$, estimate the underlying density by function $\hat{f}$.

where,

$$\hat{f}(x) = \frac{1}{nh} \sum_{i=1}^{n} K\left(\frac{x - x_i}{h}\right) \tag{2.7}$$

The principle is to place a kernel function K at each point x_i where the sum of n kernel functions (for n points) is the probability density function $\hat{f}$. The KDE plot reveals the curve of our estimated density function, where the bandwidth h acts as a smoothness parameter. This simple approach of computing KDE is also referred to as naive KDE.

FFTKDE Naive KDE performs well on small datasets, such as points extracted from windows and door detections (one point per detection). In the processing of the segmentation data, the size of the dataset grows to the number of pixels in the segmentation mask. The computational complexity of naive direct evaluation of KDE in Equation 2.7 is $O(n^2)$, given m evaluation points for n data points requiring $O(mn)$ kernel evaluations [Gramacki and Gramacki, 2017]. A very fast implementation is the fast fourier transform (FFT) based method, first described by Wand [1994]. The method consists of two steps.

The first step involves the linear binning (i.e. data discretisation [Wand, 1994]) of the input n data points, where each point is assigned to the neighboring grid point on a predetermined equidistant grid. As such, a grid count is computed for each grid point. In the second step, the kernel is evaluated once on $\leq n$ points, and then the grid counts and kernel weights are combined to obtain an approximation of the kernel estimate (KDE). The result is convolved using a series of discrete convolutions using the FFT, hence the name fast fourier transform kernel density estimation (FFTKDE). The computation complexity of the FFTKDE is $O(N2^d + nlogn)$, resulting in a runtime of 0.01 seconds for 10^6 points [Odland, 2022].

Manual optimisation is predominantly achieved by tuning bandwidth parameter h, given a kernel K with desired properties is selected. The resulting KDE curve is observed and judged with prior knowledge on the data. This empirical approach is suitable for exploratory analysis. Figure 2.27 illustrates how the selection of different bandwidths values result in different representations of the underlying data.

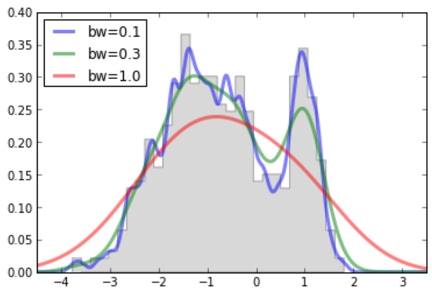


Figure 2.27.: Different bandwidths represent the underlying distribution of the data differently [VanderPlas, 2013].

Automatic optimisation is also referred to as the use of reference rules. Since the importance of bandwidth, most attention literature is given to automatic bandwidth selection. A review on fully automatic bandwidth selectors by Heidenreich et al. [2013] suggests a sample size $n > 100$, which is in line with sample size selection in original papers [Läuter, 1988; Sheather and Jones, 1991]. The most popular reference rules are Silverman's rule of thumb and improved Sheather Jones (ISJ).

Silverman's rule of thumb assumes that the true density is normally distributed unimodal data [Silverman, 2018], and performs well when data is somewhat symmetric and does not have fat tails [Heidenreich et al., 2010].

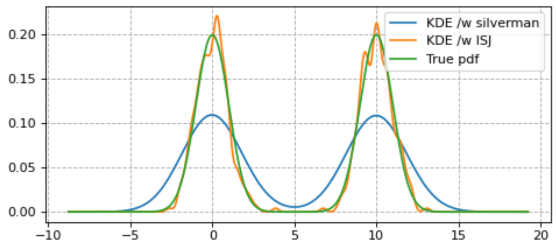


Figure 2.28.: The difference in data distribution representation as a result of using Silverman's rule of thumb compared to ISJ [Odland, 2022].

The underlying data of windows and doors do not fit unimodal data distribution. As a storey is expected to have a dense region of windows and doors, a façade with multiple storeys is expected as a multimodal data distribution. ISJ is a plug-in method that significantly outperforms Silverman's rule of thumb in the density estimation of multimodal data [Odland, 2022]. A plug-in method (e.g. Silverman's rule of thumb) requires *a priori* assumptions, often a preliminary normal model [Botev et al., 2010], on the unknown data distribution followed by a minimization of the asymptotic mean integrated square error (AMISE) of density estimator $\hat{f}$ [Chu et al., 2015]. The *a priori* assumption leads to significant bias, and is reduced in the ISJ method by introduction of a pilot density estimate $\|f''(x)\|^2$ that is free from normal reference rules [Botev et al., 2010]. However, when observing the differences between the two methods in Figure 2.28, the following should be considered. First, the resulting curve of applying ISJ represents the underlying data more closely in terms of density, but has a ragged curve. Second, the resulting curve of applying Silverman's rule of thumb results in a smooth curve, but a less precise representation of the underlying data distribution.

Maxima finding An elementary property of the KDE is that the function inherits the continuous properties of kernel K [Silverman, 2018]. Continuous functions on closed interval have the property to achieve maximum and minimum values [Fu and Yu, 2020]. Thus, the extremes of the KDE can be found where maxima are the most dense locations or "peaks", and minima are the least dense locations or "breaks" in the data. In our case, the peaks translate to rows of windows/doors, and breaks translate to the spaces in between the rows. Each peak/row is interpreted as a storey, which means that the sum of the peaks equals to the number of storeys or floor count.

In conclusion, we expect that the underlying vertical data distribution of windows and doors are separate regions that each represent a storey. The naive KDE with manual optimisation is a suitable approach for the exploration of vertical data distribution of smaller datasets. With larger datasets the faster FFTKDE is more suitable, and the larger dataset size allows for automatic bandwidth optimisation methods: Silverman's rule of thumb and ISJ. Either methods have different characteristics that have to be considered during the development of a floor count estimation method. Maxima finding will be performed as method to extract the number of vertical clusters, i.e. the estimated floor count.

2.7.3. Floor count evaluation

In order to asses the floor count estimation performance, we describe the following evaluation metrics.

Accuracy is arguably the most common metric to evaluate a model's overall performance on a scale from 0 to 1, often expressed as a percentage (%), and is defined as:

$$\text{Accuracy} = \frac{1}{n}\sum_{i=1}^{n} 1(y_i = \hat{y}_i) \tag{2.8}$$

The mean absolute error (MAE) is another commonly used metric to compare the predicted values with the ground truth values, and is defined as:

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} |y_i - \hat{y}_i| \tag{2.9}$$

In order to get more understanding on the sign of the error, we employ the mean error (ME) and normalised standard deviation of the error ($\sigma error$). The ME is defined as:

$$\text{ME} = \frac{\sum_{i=1}^{n} y_i - \hat{y}_i}{n} \tag{2.10}$$

Since we are dealing with an imbalanced dataset, accuracy score can be misleading about the performance of the method. Instead, we use harmonic mean of precision and recall, called f1-score and is defined as:

$$\text{F1} = 2 * \frac{precision * recall}{precision + recall} \tag{2.11}$$

The f1-score is given a value of 0 at worst, and a value of 1 at best. The class imbalance is accounted for by averaging the f1 score weighted by the support (number of ground truths per class).

3. Related Work

The focus of the related work regarding (large scale) floor count determination, is divided into three categories based on input data type: elevation data, attribute data and SVI data. The latter is most relevant to current study, thus described in more detail. Furthermore, research related to the detection and segmentation of floor line structures from SVI are described, since floor count information is (implicitly) present.

3.1. Floor count from elevation and attribute data

Elevation-based Many works exist on deriving building height from space-borne data, of which an overview is given by Iannelli and Dell'Acqua [2017]. From the building height, the floor count can then be estimated using geometric approaches which are described in Roy [2022]. In essence, the space-borne data are used to derive building heights (often as a by-product of a digital elevation model (DEM)/3D city model), and then the height is divided over a standardised or averaged storey height (e.g. 3 m). A purely geometric approach that was tested on a subset of a dataset containing 173,152 buildings' attributes by Roy [2022], achieved an accuracy of 69.9% and MAE of 0.31 on buildings with 5 or less storeys, and 47.5% accuracy and MAE of 0.70 on buildings with 5 or more storeys.

Attribute-based Building attribute data (e.g. floor count, height, construction year, use, etc.) are often embedded in cadastral datasets or 3D city models. For most building attributes there exist GIS standards, such as those described for floor count in Section 2.2. The general idea is that when the floor count attribute is missing (which is often the case), it can still be derived from (a combination of) other present attributes. A demonstration of a machine learning approach to infer floor count of building footprints (attributes) is given by Roy [2022]. The floor count determination is approached as a regression problem, which is less restrictive than a classification approach and allowing fractions (i.e. optional half storeys). Since it is a learning-based approach, most limitations are related to training data quality and characteristics. The best model achieved an accuracy of 94.50% and a MAE of 0.06 on buildings with 5 or less storeys, and 52.3% accuracy and MAE of 0.62 on buildings with 5 or more storeys.

Table 3.1.: Performance floor count determination comparison of machine learning (ML) approach and geometric approach by Roy [2022].

	Accuracy (%)		MAE	
	≤ 5	> 5	≤ 5	> 5
ML	94.50	52.3	0.06	0.62
Geometric	69.90	47.5	0.31	0.70

If (enough) attribute data is available, the demonstrated machine learning approach performs very well for floor count determination on lower storey buildings. The bias towards lower storey buildings is a limitation of the skew in the training data, which is reflected in the performance comparison in Table 3.1.

3.2. Floor count from SVI as classification problem

The following related works approach floor count from SVI as a classification problem with a deep learning approach. Each work is first introduced, followed by explanation of the approach, and finally limitations with relevant suggestions are described. Then, the observations on the related works are summarized with key takeaways. Thereafter, the fundamental differences between the related work and current study are articulated.

Iannelli and Dell'Acqua [2017] were the first to use deep learning to automatically infer the floor count from SVI. An overall performance in terms of accuracy of 85% was achieved on a test dataset of 430 images. Most failure cases were an under- or overestimation of one storey. Main benefits are feasibility and cost efficiency of the method when compared to labour costs of manual floor count determination.

The floor count determination is addressed as a multiclass classification problem, with five predefined classes: 0, 1, 2, 3 and 4+ storeys. Each façade image is fed to a pre-trained (i.e. ImageNet [Deng et al., 2009]) VGG-16 CNN-based architecture [Simonyan and Zisserman, 2014], then re-trained on their own annotated SVI dataset containing 600 images (acquired through the Google Maps API). Unfortunately, neither their dataset nor model are released to the public.

The approach has a couple of limitations. First, the decision for using the open-ended class of 4+ storeys, is based on the limitation of the vertical field of view in SVI from the Google Maps API response. Second limitation is related to heterogeneous class distribution in the training dataset, deemed unachievable in practice. Third, most error cases are *possibly* due to occlusions. Forth, main drawbacks of using SVI mentioned are limited coverage in some areas (albeit improving) and temporal quality issues. To improve the work, large-scale fusion SVI with satellite remote sensing data is suggested and use of improved CNN architectures.

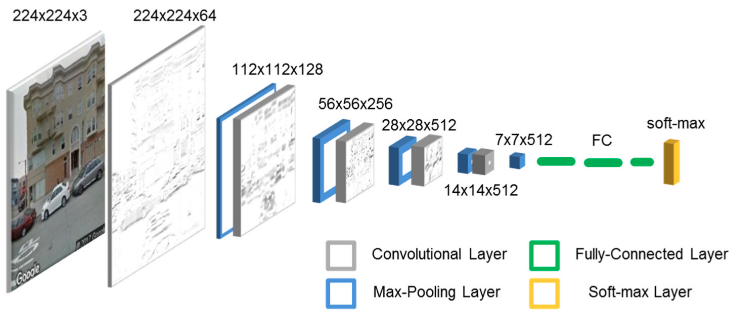


Figure 3.1.: The VGG-16 CNN pre-trained on the ImageNet dataset , used by Iannelli and Dell'Acqua [2017]. The network was retrained on 600 street view images to predict one of the five predefined floor count classes, achieving an accuracy of 85%.

Rosenfelder et al. [2021] proposed a deep learning model using both aerial and street view imagery, to predict building-level electricity consumption. Floor count is predicted as one of the intermediate features, achieving overall performance of 90.5% in terms of accuracy and 0.1 in terms of MAE on a test dataset containing 22,803 images. The main benefits of cost/time effectiveness and feasibility are also mentioned, replacing surveys or manual measurements.

The floor count determination part of the model is also addressed as a multiclass classification problem, with three predefined classes: 1, 2 and 3 storeys. A ResNet-34 CNN-based architecture is pre-trained on ImageNet, and then re-trained on their own annotated SVI dataset containing 843 images (acquired through the Google Maps API). Unfortunately, also their dataset and model are unpublished.

Limitations regarding floor count determination specifically are not mentioned, except for the 20% of gross errors due to a missing building in the image. Floor count determination is not the focus in their work, and the number of storeys is concluded to be of secondary importance in their sensitivity analysis, which could explain the lack of further details. Nevertheless, practices that improved the overall electricity consumption prediction performance, and perhaps the underlying floor count determination performance, of their machine learning pipeline are: BayesSearchCV for automatic hyperparameter optimisation and application of data augmentations. To improve the work, feature engineering and evaluation of other prediction methods are suggested.

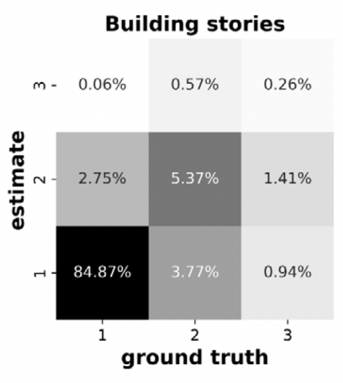


Figure 3.2.: The confusion matrix showing the floor count determination performance of the ResNet-34 CNN approach by Rosenfelder et al. [2021]. In this case the CNN predicts three predefined floor count classes.

Chen et al. [2022] proposed a CNN-based architecture that estimates building attributes (i.e. foundation height, foundation type, building type and *number of storeys*) simultaneously from GSV images for flood risk assessment. An overall floor count prediction performance of 93.5% in terms of accuracy and a 94.6 % F1 score of is achieved, on a test dataset containing 8,593 GSV images. Main benefit of the method mentioned is replacing manual labour.

The floor count determination problem is treated as a binary classification problem, of two predefined classes: 1 or 2+ storeys. Their CNN-based MTL, called Task Relation Encoding Network (TREncNet), is architecturally modified such that it can learn known relations among the three different attributes as illustrated in Figure 3.3. In addition, a feature fusion technique is applied that introduces metadata (e.g. camera-to-building distance and aspect ratio) to the MTL network. Similar to the two aforementioned works, their CNNwas pre-trained on the ImageNet datasets. The network was re-trained with 33,822 GSV images along with attribute data. The GSV images were obtained using the Google Maps API. Training was optimized with hyperparameter search and random augmentations (i.e. brightness, contrast, flipping, saturation and hue). TREncNet and the dataset used are unpublished.

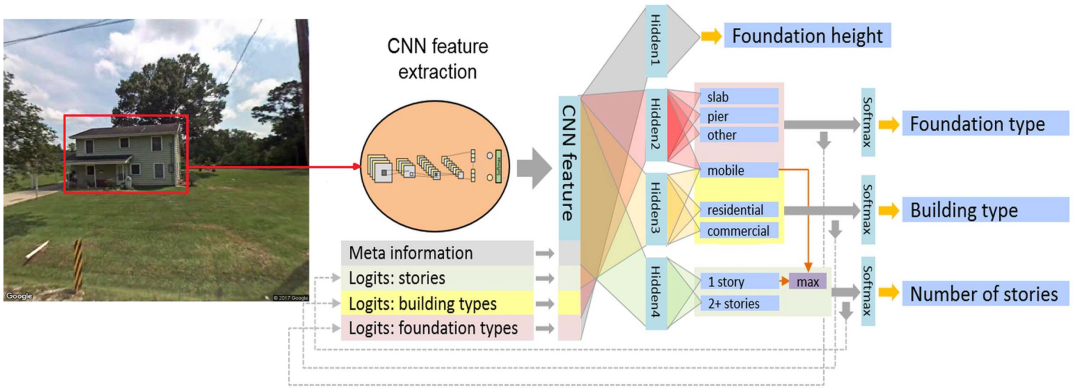


Figure 3.3.: Chen et al. [2022] developed a MTL CNN architecture called TREncNet, and enhanced by a feature fusion technique that introduces metadata into the learning process to predict multiple building attributes including floor count.

Main limitation is related to coverage issues, as roughly half of all building images through the Google Maps API were unsuitable or not available. A building image may be acquired at large distance when the distance between the road and building is large, more common is sparsely populated areas in Florida. Other limitations related to floor count classification are due to neighboring buildings that confuse the model, noise, ambiguity, occlusions, and false annotations in ground truth data.

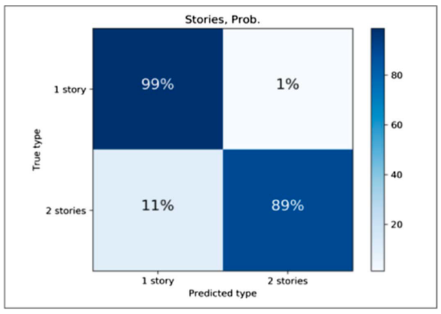


Figure 3.4.: The floor count determination performance of TREncNet shown in a confusion matrix. The performance is very good in terms of accuracy, but limited considering only 1 or 2 storeys can be detected [Chen et al., 2022].

Observations The related works of Iannelli and Dell'Acqua [2017]; Rosenfelder et al. [2021]; Chen et al. [2022] (summarised in Table 3.2) interface with current study in terms of learning-based approach to floor count determination, and use of SVI. On the one hand, it is expected that performance of learning-based models increase with the size of the training dataset (with significant but not extreme improvements), and with increasingly more sophisticated CNN-based models and supporting techniques. On the other hand, while training data sets increase the predefined floor count classes decrease, making the task arguably easier with each class less. What the works have in common is the use of transfer learning, use of GSV datasets, class imbalances, coverage and occlusion issues, and unpublished models and datasets. Key takeaways for this research are use of transfer learning, use of data augmentations in model training, taking into account coverage and occlusion issues, and publishing of the method developed and (enriched) dataset.

Table 3.2.: A comparison of the image-based floor count classification works of 1. Iannelli and Dell'Acqua [2017], 2. Rosenfelder et al. [2021], and 3. Chen et al. [2022].

	Architecture	Floor count classes	Pretrained	Accuracy (%)	Train/test images
1.	VGG-16	0, 1, 2, 3, 4+	Yes, ImageNet	85	600/430
2.	ResNet-34	1, 2, 3	Yes, ImageNet	90.5	843/22,803
3.	TREncNet	1, 2+	Yes, ImageNet	93.5	33,822/8,593

Differentiation The fundamental difference between the aforementioned works and current study, is the *approach* to the task of floor count determination. Instead of treating floor count determination as a classification problem, the floor count is estimated by utilising the underlying data distribution of the façade parsing outputs. The result is that the floor count is not reduced to a set of predefined classes, but is less restricted and more data-driven. Another difference is that the errors or failure cases could not be explained with satisfactory or concrete causes. This is due to the black-box working of the CNN-based approach (with many hidden layers), whereas the proposed method in this research strives to be more explainable and expose why floor count determination fails for each failure case.

3.3. Floor line structures from SVI

Floor(-level) lines are lines that are related to the structure of a storey. In related works these are referred to as floor-level lines or simply floor lines. Although floor count is not a direct concern in these works, the number of storeys is (implicitly) present.

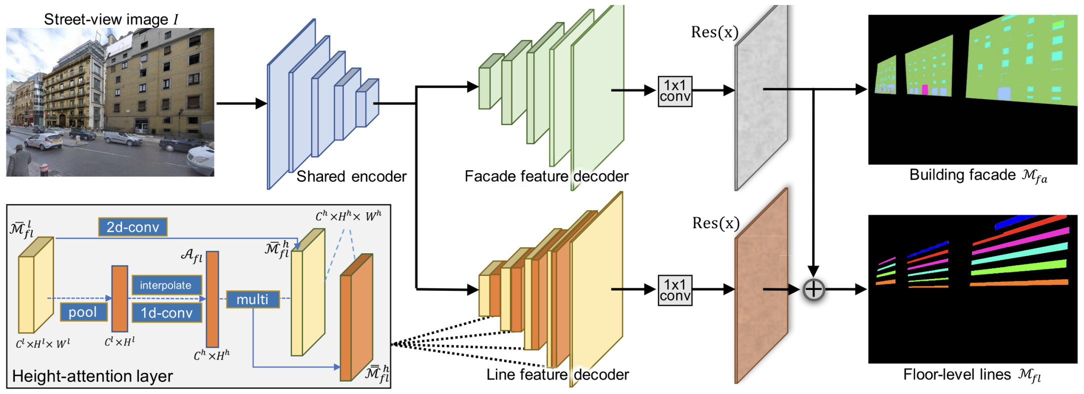

Street-view image I
Shared encoder
Facade feature decoder
Res(x)
1x1 conv
Building facade $\mathcal{M}_{fa}$
Res(x)
1x1 conv
Floor-level lines $\mathcal{M}_{fl}$
Line feature decoder
$\bar{\mathcal{M}}_{fl}^{l}$
2d-conv
$C^h \times H^h \times W^h$
$\mathcal{A}_{fl}$
$\bar{\mathcal{M}}_{fl}^{h}$
interpolate
pool
1d-conv
multi
$C^l \times H^l \times W^l$
$C^l \times H^l$
$C^h \times H^h$
$\bar{\bar{\mathcal{M}}}_{fl}^{h}$
Height-attention layer

Figure 3.5.: Overview of the FLN architecture: A MTL network that performs façade parsing and floor-level line detection simultaneously by hard parameter sharing. The input is an image, and the outputs are two mask for each of the performed tasks. Observe in the bottom left the implementation of the height-attention layer Wu et al. [2021].

FloorLevel-Net (FLN) [Wu et al., 2021] is designed to detect the floor-level lines of buildings from images (see Fig. 3.6). For the detection of separate storeys, two main observations are made. First, the task of detecting floor-level lines is dependent on the contextual information given by the semantic/instance segmentation. Second, to which floor each instance (i.e. window, door) belongs is dependent on the vertical distribution in the image. The same observations, the second in particular, could be made for the counting of storeys from façade parsing results. FLN takes these two observations into account by adoption of a MTL approach and incorporation of a height-attention module. In deep learning and in the case of CNNs, a channel attention module is implemented to guide the network what to focus on when learning/inferring the task of interest [Liu and Milanova, 2018]. In the case of FLN, the height-attention module is guiding to detect the floor-level lines in a regularly vertical distribution, resulting in better detection of floor-level lines.

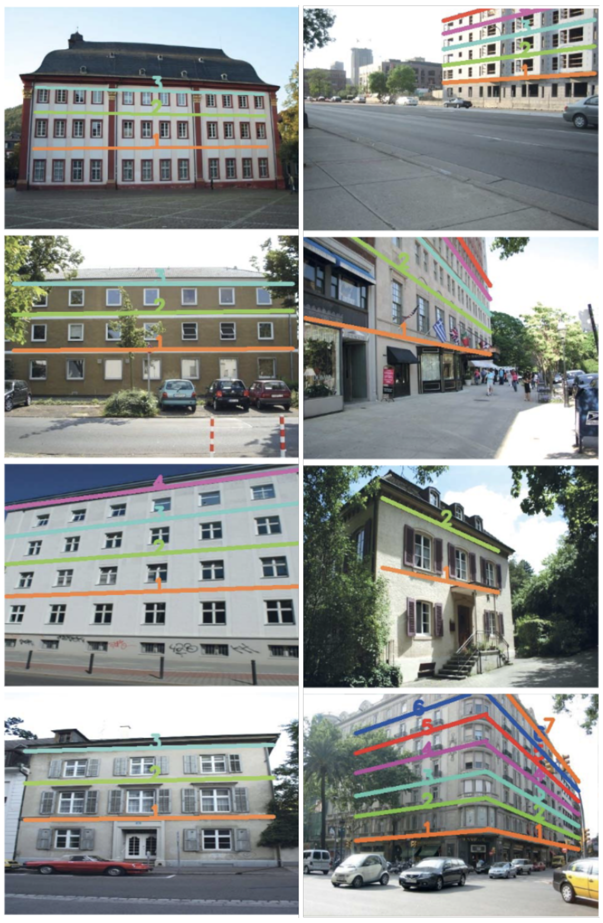


Figure 3.6.: The results of floor-level line detection with FloorLevel-Net, on more orthogonal view images (left) and perspective view images (right). Note how the floor-level line number corresponds to the number or count of the storey it is assigned to. In most cases, the highest floor-level line number corresponds to the floor count. Adapted from [Wu et al., 2021].

The parallel with floor count determination becomes clear when observing Figure 3.6. Each floor-level line is outputted by the algorithm as five-tuple. For the horizontal and vertical ranges of each façade, two endpoints are derived for each polyline: (x_s^i, y_s^i) and (x_e^i, y_e^i). Finally, the floor order l^i is added. So the five-tuple for each polyline is structured as follows: $\{ x_s^i, y_s^i, x_e^i, y_e^i, l^i \}$. For the floor count, the highest floor order corresponds to the floor count of a given façade. Given that the results in Figure 3.6 are robust to either orthogonal or perspective view imagery, the method looks promising for floor count determination on SVI.

Floor segmentation Håbrekke and Nordstad [2022] performed "floor segmentation" for the estimation of facade heights, where the number of storeys is used as an intermediate. The first step is façade parsing of classes window, door and balcony, and is approached as an object detection problem. The façade detection network used is a YOLO v3 architecture [Redmon and Farhadi, 2018], that was pre-trained on ImageNet [Deng et al., 2009] and re-trained on the custom FaçadeWHU dataset containing 900 street view images by Kong and Fan [2020]. As a second step, each object center is horizontally populated with three equally spaced points, as illustrated in Figure 3.7.

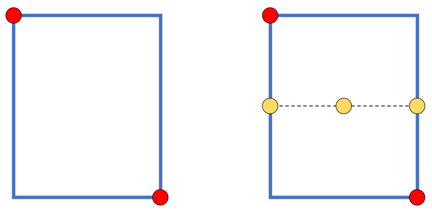
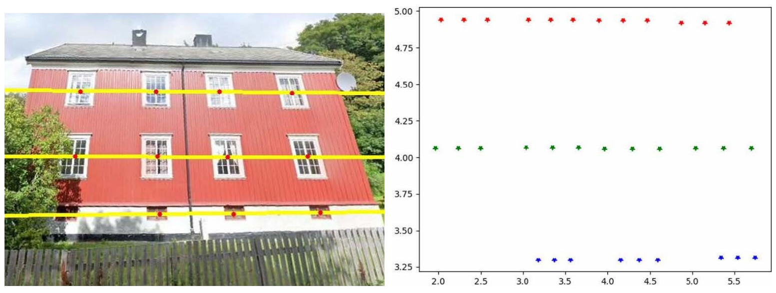


Figure 3.7.: Håbrekke and Nordstad [2022] populated each object center horizontally with three equally spaced points.

The result of populating the object centers horizontally, is that the point clusters are denser horizontally than vertically, forming horizontal "point lines" visualised in Figure 3.8b. This helps line fitting using multi-RANSAC [Zuliani et al., 2005] in the third step. Additional restrictive rules avoid lines to intersect, and misalignment or significant deviations. The "correctness" of the floor segmentation was measured in terms of number of storeys, i.e. floor count. Of 50 manually assessed façades, the floor count performance was 92% in terms of accuracy with an "mean error degree" (although not specifically defined, we assume this corresponds to MAE) of 1.25.



(a) **(b)**

Figure 3.8.: Floor segmentation with façade parsing, and multi-RANSAC applied on horizontally populated object centers [Håbrekke and Nordstad, 2022].

The difference of the method by Håbrekke and Nordstad [2022] and current study, is to leverage the regularity assumption more directly by first rectifying each image and then to vertically cluster the points, instead of applying RANSAC with restrictive rules. Although the results look promising on the 50 manually assessed samples, introducing restrictive rules can limit the potential to adapt to datasets with varying image quality and characteristics.

4. Experimental design and development

From the research objectives defined in Chapter 1, and the literature review conducted in Chapter 2 and 3, we propose a three staged methodology: 1. data preparation, 2. façade parsing, and 3. floor count estimation. A compact overview of the methodology is given below in Figure 4.1, and a comprehensive overview can be found in appendix A.1.

In the first stage, we explain the steps necessary to prepare the data for the following stages. Except for the dataset selection and creation, which are detailed in Chapter 5. In the second stage, the deep learning networks investigated and used are described, along with training and optimisation details. In the third and final stage, data processing steps and the various methods for vertical clustering, optimisation and floor count estimation are discussed.

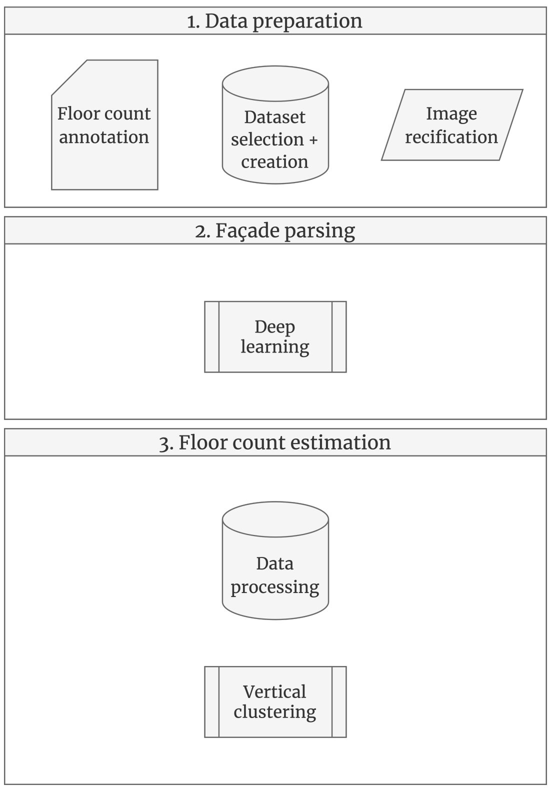


Figure 4.1.: A compact overview of the proposed three-stage method, which consists of: 1) data preparation, 2) façade parsing, and 3) floor count estimation.

4.1. Data preparation

In the data preparation stage, we explain the development steps in the floor count annotation and image rectification explored. Image rectification is considered to be the main pre-processing step to the façade parsing stage.

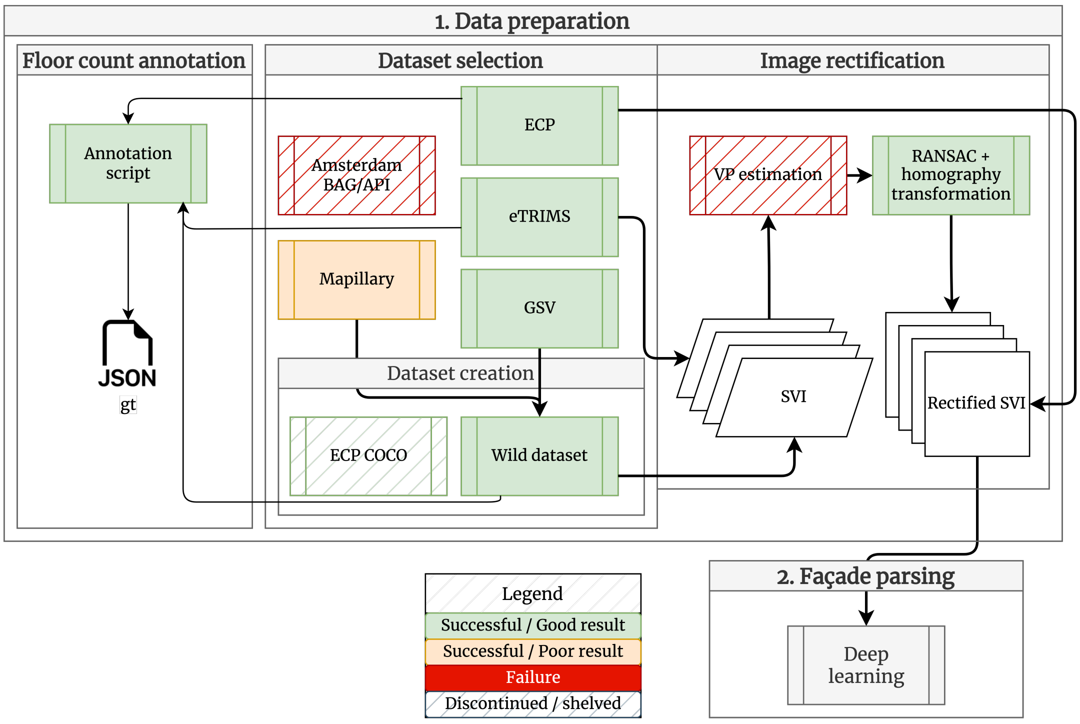


Figure 4.2.: Flowchart that details the development steps within the data preparation stage.

4.1.1. Floor count annotation

Ground-truth data (i.e. floor count annotations) are necessary for evaluation of the floor count determination method. To the best of our knowledge, there are no façade datasets that contain floor count annotations. Hence, floor count annotations are added manually to datasets used in this study (see Section 5.2). The floor count is determined by visual inspection of each façade image, according to the definition specified in Section 2.2.

For visual inspection of each façade image, a simple script is written in Python to loop through a file of images, plot each image and request input from the user to enter the number of storeys. Each record is encoded as `storeysAboveGround_gt` and the value is stored as a non-negative integer. The floor count annotations are written to a `JSON` file as follows:

```
{
   "image_name": {
      "storeysAboveGround_gt": int
   }
}
```

The benefits of a `JSON` file include convenient I/O operations, quick key-value lookup, and easy transformation to a dataframe for further processing and analysis. The `JSON` file is used later for storing predictions in the same file.

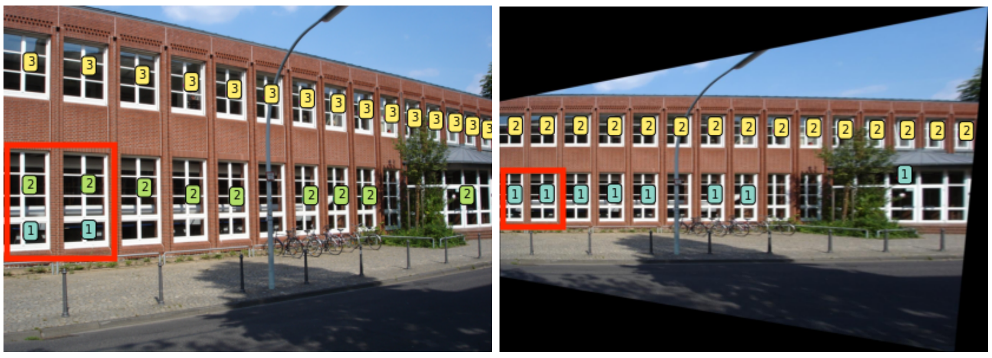
During floor count annotation, the datasets are observed with special attention to image quality aspects, image characteristics and floor count ambiguities.

4.1.2. Image rectification

The rectification is applied on the eTRIMS and custom "in the wild" datasets, as both represent SVI containing perspective view images. The rectification is not needed for the ECP and Amsterdam Facade datasets since both are pre-rectified. After data preparation, the (pre-)rectified images are passed onto the façade parsing stage.

VP estimation In earlier experiments, image rectification by VP estimation is applied, similar to the approach mentioned by Liu [2011] as described in Section 2.4. Instead of using specific horizontal and vertical line segment detection, we use Canny edge detection [Canny, 1986], and apply VP estimation with RANSAC. The homography is computed, and applied to the image which warps the image towards a fronto parallel view with orthogonal axes. The tests for this method are only conducted on the "wild dataset", as the dataset is smaller and the perspective view angles larger. The image rectification quality is assessed qualitatively.

RANSAC and homography transform Instead of finding VPs and then compute the homography, the implementation by Affara et al. [2016] as described in Section 2.4 is applied. We run the automatic implementation in MATLAB on the eTRIMS and wild datasets. Image rectification results are assessed qualitatively, and quantitatively in relation to floor count estimation improvement.



(a) Perspective view image. **(b)** Rectified image.

Figure 4.3.: Rectification on a perspective view image from the eTRIMS dataset in 4.3a. Observe how the rectification result ensures the applicability of the regularity assumption of façade-elements in 4.3b. As a result, the grouping of storey elements is improved.

4.2. Façade parsing

After the data preparation stage, we use learning-based façade parsing to localize windows and doors in rectified images. In the following sections we explain which networks are investigated and how they are implemented.

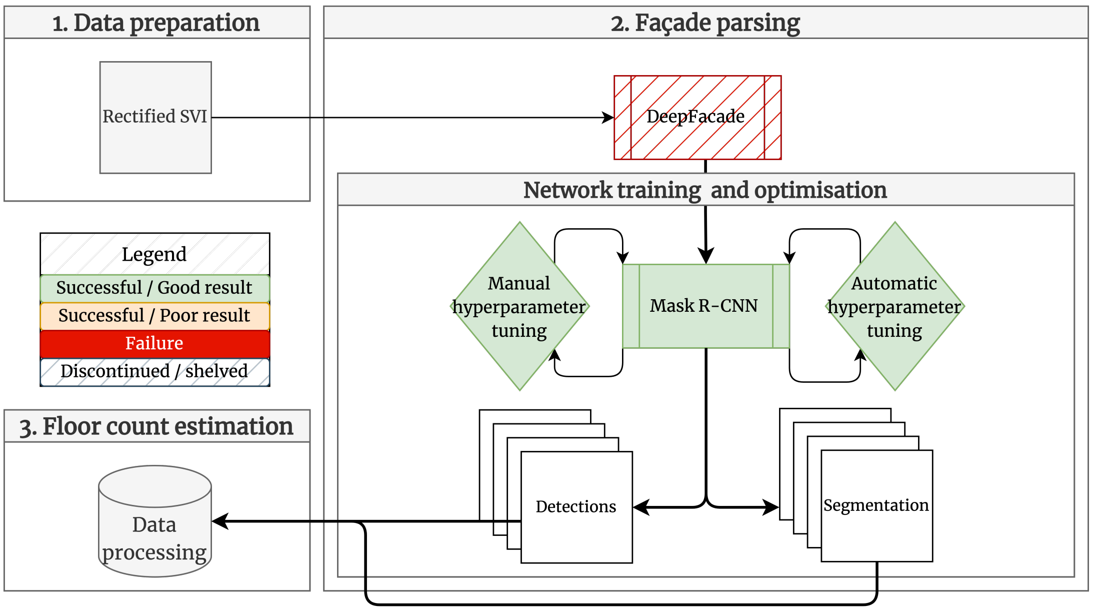


Figure 4.4.: Flowchart that details the development steps within the façade parsing stage.

4.2.1. DeepFacade reproduction

Initially, we attempted to reproduce the original work of DeepFacade by Liu et al. [2020] for the façade parsing stage, which is published on GitHub [Liu, 2020]. Even though the repository was already mentioned to be "somewhat deprecated" around time of publishing in 2020 by the author, considerable time is put into debugging the code with multiple machines and operating systems (Mac OS Monterey, Linux Ubuntu 18.04 and Windows 10). Alas, two and a half years after publishing the code is severely deprecated and not maintained by the authors, or anyone for that matter. Further experimentation on DeepFacade is discontinued in current research. For future reference, in order to reproduce DeepFacade the main model has to be ported to a supported version of the `PyTorch` library Paszke et al. [2019] along with all other dependencies. Preferably on an intel-based Linux (virtual) machine under a stable release of Ubuntu with a NVIDIA GPU supporting CUDA drivers [Nickolls et al., 2008], which is important for training.

4.2.2. Mask R-CNN training and optimisation

Initialisation of network For Mask R-CNN we make use of transfer learning, as observed to be useful in training on a relatively small dataset in Chapter 3. The pre-trained weights are loaded, from a model that is trained on the Common Objects in Context (COCO) dataset. As the name suggests, the model is pre-trained on common objects, which do not include the window and door class. Therefore, we have to re-train the model on our domain specific classes. We select the window and door as foreground classes, and sky as background class.

First network optimisation As the model is initialised, we start manually configuring the hyperparameters. Starting from the default hyperparameters, we tune one hyperparameter at the time with a few training iterations (epochs) to get a feeling for the most important hyperparameters for the combination of Mask R-CNN and our dataset. In order to understand whether the change had an effect, we observe the performance (i.e. accuracy) and optimisation curves (training and validation loss). Accuracy is maximised, and the losses are minimised. All progressions are logged, and training weights are saved with a filename that translates to the changed hyperparameter and value.

Backbone comparison Once we obtain acceptable results, i.e. accuracy is $> 70\%$, we keep the hyperparameters constant and compare the two backbones discussed in Section 2.5. Both backbones are pre-trained on the COCO dataset containing 300K+ images and 2.5M labeled instances [Lin et al., 2014].

Final optimisation For final optimisation we select the best performing backbone and continue optimisation of the model's performance, both manually and automatically.

For manual optimisation, the set of the most performant hyperparameters are selected and we run a training routine with a high number of iterations to capture the full learning curve. With the full learning curve we can observe at what number of iteration the model starts overfitting and select that number as final learning rate (LR) parameter.

Then, with the knowledge of the best configuration of hyperparameters so far we move onto automatic hyperparameter tuning. We define the set of hyperparameters that we would like to be 'searched', and define the maximum number of routines. This is done exhaustively in the beginning with a smaller subset of hyperparameters, but this is computationally expensive. Both Bayesian and random search are applied instead, which are found to lead to better searches in less routines. Although they are more efficient, the total number of automatic hyperparameter searches are still limited due to computational costs.

4.3. Floor count estimation

After the façade parsing stage, we arrive at the floor count estimation stage. This stage has three main lines (see Figure 4.5). FLN, bivariate approach with HDBSCAN, and the univariate approach with (FFT)KDE. The three main lines are each explained with their additional development steps (if applicable) in the following sections.

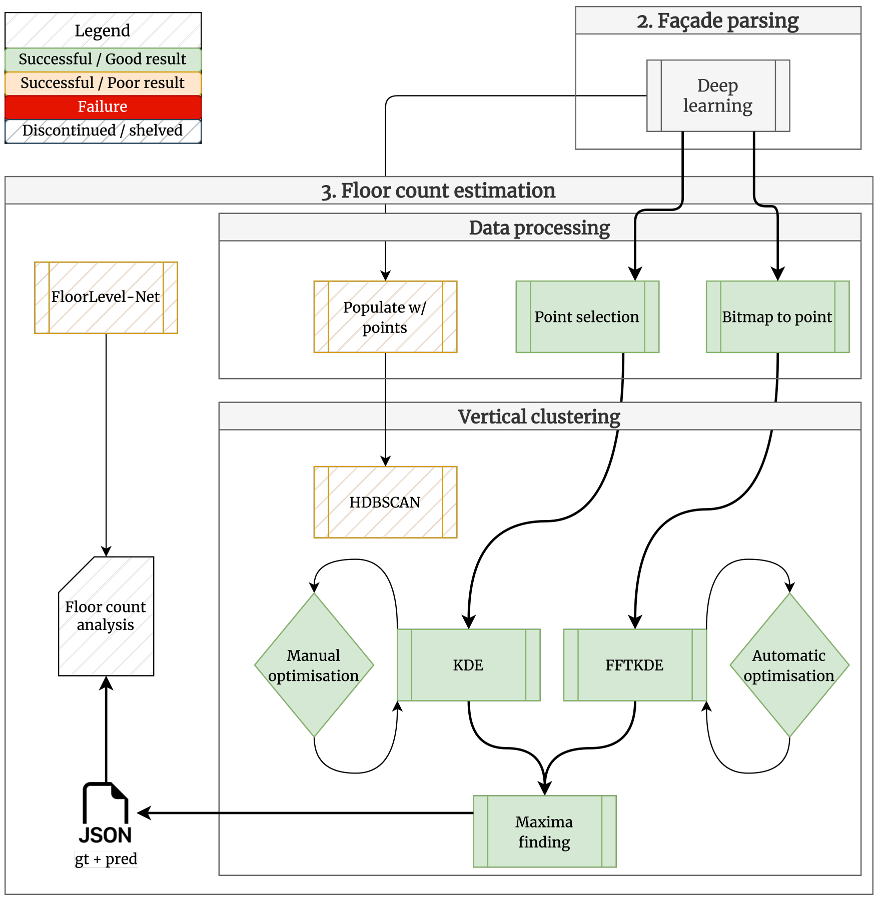


Figure 4.5.: Flowchart that details the development steps within the floor count estimation stage.

4.3.1. FloorLevel-Net

As observed in Section 3.3, FLN orders the the detected floor-level lines with an order number, where the maximum order number should in most cases correspond to the total number of storeys. The code of the model is published by the authors along with model weights and their custom dataset, which facilitates and encourages reproduction. The custom data is a combination of their own GSV images and the CMP dataset. We run inference of their model on their data and the ECP dataset, and write the highest floor order number to the ground truth file for later analysis.

4.3.2. Bivariate approach

The second line of development is called the bivariate approach, since we attempt to find horizontal clusters from the detection pixel-coordinate (x, y) data.

Populate detections with points The location of each detection is represented as a bounding-box encoded by its spatial extent, i.e. the minimum and maximum pixel-coordinate values in a four-tuple: $bbox = (x_0, y_0, x_1, y_1)$. Similar to the floor segmentation approach of Håbrekke and Nordstad [2022] discussed in Section 3.3, we populate each center line horizontally with three equally spaced points.

First, we create horizontal line segments that are denoted by boundary representation (b-rep), i.e. two bounding points. We get the center from each detection with a built-in function from Detectron2. We combine the y value of the center with both the minimum and maximum x values of the bounding box into horizontal center line $= (x_0, y_{center}, x_1, y_{center})$.

Second, we create a function that allows for variable point density in between the lines. The initial approach is to have 3 equally spaced points per line. Other attempts include uniform density over entire image width, or proportional between detection and image width.

HDBSCAN The densified horizontal lines are fed to the next stage, HDBSCAN. We run HDBSCAN on default parameters, since it is designed to vary search radius ϵ automatically (a desired feature) as discussed in Section 2.7.

The clustering results are mapped to the pixel-coordinate values and plotted, to qualitatively assess the clustering performance while tuning the line densification function.

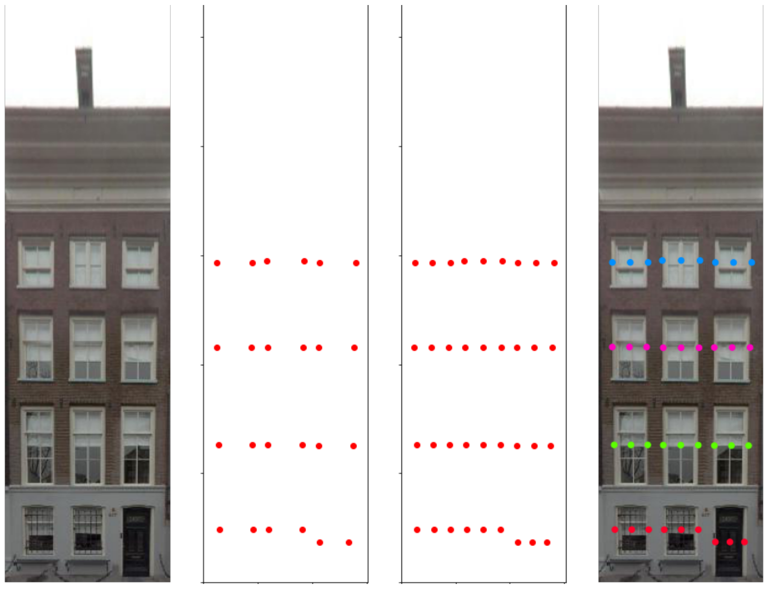


Figure 4.6.: An illustration how bivariate clustering is implemented, by first finding lines, then populating each line, and finally applying HDBSCAN.

4.3.3. Univariate approach

The bivariate approach is visually intuitive. However, the information that reveals where the concentration of window and door rows are, is contained in the vertical distribution of the data. By analogy, we often count storeys by counting the number of windows in one column. As such, the vertical data distribution is of primary interest. We isolate the y data and approach the problem by performing a univariate analysis as defined in Equation 2.7. First we describe the development line from detection-based data, followed by the development line from the segmentation-based data.

Detection-based

Point selection As previously mentioned, Mask R-CNN outputs detections as bounding-boxes. The KDE requires an array of data points as input, so from each detection one y-point is extracted and appended to an array. The array is sorted in ascending order, which improves the KDE performance.

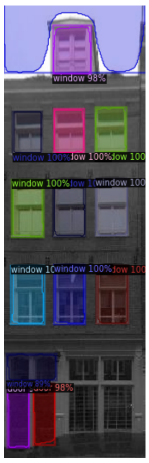
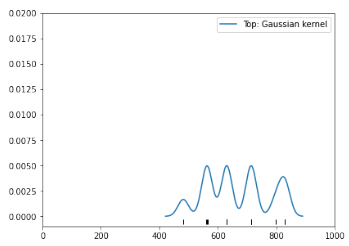
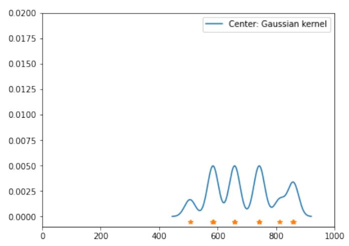


Figure 4.7.: For this particular façade, the point selection matters for the outcome of the floor count determination. Note the smaller windows above the doors (bottom left).

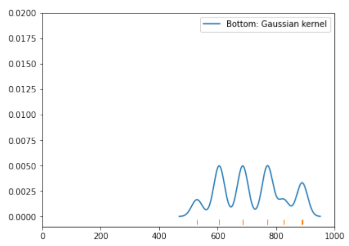
The question arises: which point position from the detection bounding-box should be selected? Although the location of windows related to symmetry/regularity in façades is extensively discussed in literature, no statistical analysis is found on the "center (point) of gravity" of façade elements concerning symmetry and regularity. Therefore, the top, center and bottom y-points are extracted and fed to separate KDEs, while bandwidth and kernel are kept constant. By comparing the three different plots in assessment, we gain a better understanding of the effect of point position selection on the floor count estimation performance.



(a) Top



(b) Center



(c) Bottom

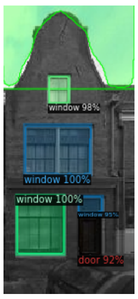
Figure 4.8.: The KDEs derived from top (4.8a), center (4.8b) and bottom (4.8c) point selection from façade detections in Fig. 4.7. The smaller windows above the doors result in less clearly separated peaks in 4.8b and 4.8c, which is an undesired effect as it leads to false positives in more severe cases.

KDE and manual optimisation After point selection, the sorted array of y-points is fed to the (naive) KDE. The KDE takes two parameters, kernel K and bandwidth h. The bandwidth optimisation is done manually for the detection-based data. Each parameter is tuned separately, while continuously assessed.

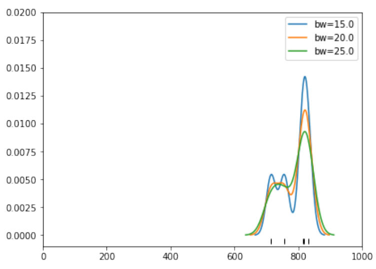
For bandwidth optimisation, a range of bandwidths are plotted, with kernel and point selection kept constant. The curves are compared to one another in terms of curve shape. Special attention is given to the ability of each corresponding curve to approximate the correct number of maxima (which will be elaborated on later in this section).

For kernel optimisation, three different kernels with the continuity property are selected: Gaussian, tricube and cosine. This time, the bandwidth and point selection are kept constant. The rest of the optimisation procedure is similar. The curves from different kernels are compared with one another, where special attention is given to the ability of each curve to approximate the correct number of maxima.

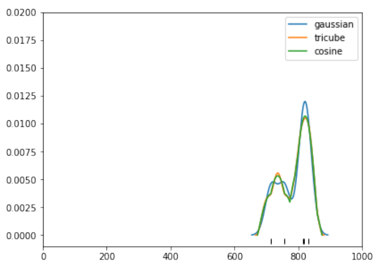
Automatic bandwidth optimisation, e.g. Silverman's rule of thumb or ISJ, did not work for our detection-based data points. Most input images in our test set (Amsterdam Facade dataset) have between 5-20 detections, resulting in the equal amount of data points. Examples in literature show sample sizes $n > 100$ for the use of automatic bandwidth optimisation.



(a) Top



(b) Center



(c) Bottom

Figure 4.9.: The bandwidth and kernel optimisation with the help of plots showing bandwidth range and three different kernels. Note how bandwidth has a stronger influence on the curve shape.

Segmentation-based

Bitmap to points Mask R-CNN outputs each instance segmentation in the form of a bitmap. In our case, each bitmap is an array the size of the input image. Each pixel that contains the instance is encoded as `True`, all other pixels are encoded as `False`. As mentioned before, the KDE requires a one dimensional array containing coordinate data, meaning the bitmaps must be processed before passing onto the KDE. First, an array of zeros in the same shape as input image is initiated. Each segmentation is concatenated to the zero array, where each `True` value is converted to pixel value of `255`. The resulting array is gray-scaled and split into an x- and y-array. The y-array has a larger range and size than the detection-based array, leading to more resolution in the distribution of points. Compared to the detection-based array, the segmentation-based data is "spread out". Noise may be estimated as differences in the data distribution by the KDE. The method to find the extremes on this data led to erroneous errors, especially near the tails of the KDE function. Data normalisation alleviates this side effect. The results are compared by plotting both the KDE functions of both normalised and non-normalised segmentation-based data.

FFTKDE and automatic optimisation The size n of the y-array is significantly larger than the detections-based point arrays. The segmentation-based arrays are up to the size of the input image, in many cases $n \approx 1000$. Conventional KDE runs somewhat slow (seconds per image), which is not desirable when developing a pipeline towards large scale and automatic processing of SVI. Therefore a fast convolution-based implementation of KDE using the FFT (described in Section 2.7) is implemented with the `KDEpy` library [Odland, 2022].

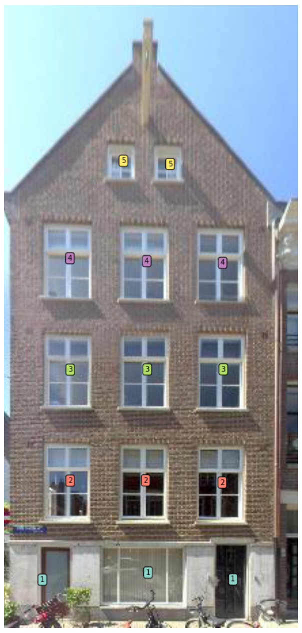
Another consequence of the larger size of the array, is the option to utilise automatic optimisation. The optimisation methods described in Section 2.7, i.e. Silverman's rule of thumb and ISJ, are implemented with a constant kernel (Gaussian). The oversmoothing effect of applying Silverman's on multimodal data is compared with the curve of ISJ, which is supposed to outperform Silverman's with multimodal distributions. Both plots are judged in relation to maxima finding and its performance in floor count estimation.

4.3.4. Maxima finding

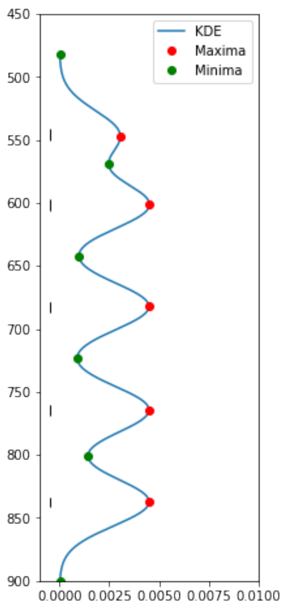
The output of both KDE and FFTKDE is a curve (of function $\hat{f}$), in the format of a grid of points (array). The implementation of finding the (relative) extremes is done with a SciPy library named `argrelextrema` [Kramer, 2016]. It finds the extremes by traversing the array output of the KDE and comparing neighboring values. A local maximum in an array `a` is found when: `a(maximum) = a[n-1] < a[n] > a[n+1]`. A local minima is found when: `a(minimum) = a[n-1] > a[n] < a[n+1]`.

The maxima are the most important since the sum is the estimated floor count, which is written to the corresponding ground-truth file for evaluation. The minima are used in conjunction with the maxima to find "breaks" in the data, as explained in Section 2.7. The data that is in between minima is assigned to the maximum in the middle, which enables mapping the storey number to the corresponding detections. Remember that upon sorting, the indices of the array are stored for this purpose. Though not necessary for floor count estimation, it helps visualise the results which in turn is useful for error analysis.

The results of the experiments are evaluated both quantitatively with metrics defined in Section 2.7 and qualitatively with side-by-side plots of density estimation curves and input image.



(a) Input image with mapped floor count information.



(b) Corresponding KDE, plotted with found extremes.

Figure 4.10.: The extremes of the KDE are used to find the most dense locations of data, called maxima or peaks. The peaks are where most data points are representing windows and doors, which are interpreted as storeys. As such, the sum of the peaks is the floor count. The example shows a detection-based floor count determination.

5. Implementation and Experiments

5.1. Software and tools

Python All development of code is written in Python, unless stated otherwise.

Google Colaboratory For the implementation of the façade parsing and floor count estimation stage, an interactive Python Google Colaboratory Pro environment is employed. The interactive environment allows for quick prototyping, and the pro license allows the use of a single NVIDIA A100 SXM4 40 GB GPU for fast training and inference of the façade parsing model.

Detectron2 Mask R-CNN is built with Meta's Detectron2 platform for AI research [Wu et al., 2019]. The framework offers faster training than others [Meta AI Research, 2020], includes a backbone "library", is open-source, and is designed to be integrated into a Google Colaboratory environment.

MATLAB For to automatic rectification, a MATLAB implementation is employed provided by Affara et al. [2016]. Currently, the implementation is limited to Windows due to a included MEX file for the line segment detection part of the rectification method. This makes the image rectification part of the data preparation stage separate from the rest of the pipeline. In future development, the MATLAB implementation could be converted into a Python-package such that the entire pipeline is in Python.

Repository All developed code, model weights and datasets created and used in this research will be made available online and is open-source. The repository is pending, and will be published on the following page: `https://github.com/Dobberzoon/Facade2Floorcount`.

5.2. Datasets

Amsterdam Facade dataset The Amsterdam Facade dataset is central is this thesis as it is used for training and testing the façade parsing network, as well as for the development of the floor count determination method. Reasons considered for selecting this dataset are open data, annotations, and the image and dataset characteristics. Open data makes research more accessible and reproducible. The annotations in the dataset adhere to the minimum requirement of façade elements needed to identify a storey; window and door. The annotation style used, COCO-style, is convenient as it is supported by the Detectron2 library used in this research.

More important are the characteristics of the images in the dataset, as these will significantly influence how and what the façade parsing network will learn. The image characteristics considered are quality, variety and source. Many of the datasets used in façade parsing studies are often manually rectified, taken with DSLR cameras, small in dataset size and/or too homogeneous.

Although manual rectification leads to good rectification quality, the quality is not representative of the (lower) quality from automatic image rectification as is the case for images in the Amsterdam Facade dataset. Image (acquisition) source will also affect image quality. DSLR cameras generally take higher quality images with less distortion, when compared to panoramic cameras found in large scale acquisition of SVI. Moreover, the DSLR camera can be positioned more consciously, resulting in a front-view perspective. The Amsterdam Facade dataset is generated with the Amsterdam Panoramabeelden (Panoramic imagery) API, that can be used to retrieve SVI from Amsterdam. The SVI is acquired with an omnidirectional mobile imaging system, shown in Figure 5.1. The images are closer to real world SVI in terms of image quality and characteristics.

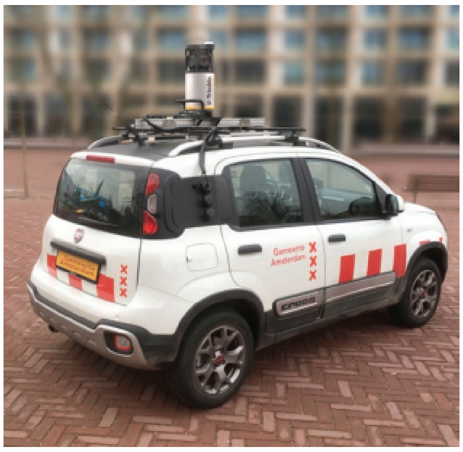


Figure 5.1.: The car of the City of Amsterdam that is used to capture SVI from Amsterdam, equipped with a 360° Trible MX7 mobile imaging system. Source: [Geospatial, 2022]

Considerations related to the characteristics of the dataset itself, are the size and variety. With 910 annotated images, the dataset is larger than most datasets used in other façade parsing studies. In terms of variety, the images originate from the North and West of Amsterdam which are chosen due to their relative architectural variation [Eijgenstein, 2020]. Both size and variety are beneficial for the façade parsing network to learn, which can translate to better generalisation of the network.

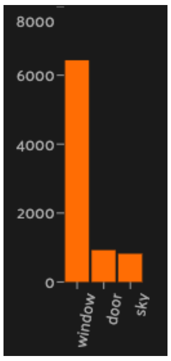


Figure 5.2.: Class imbalances in Amsterdam Facade dataset, containing 910 façade images.

One observation made on the Amsterdam Facade dataset is the class imbalance in terms of number of annotated instances, illustrated in Figure 5.2. The window-class is by far the most dominant class, which naturally inherent to the real-world occurrence ratio between windows and doors. This can result in bias towards the majority class (window) in the façade parsing model.

Evaluation datasets For further evaluation of the proposed method, we included two benchmark façade parsing datasets: Ecole Centrale Paris (ECP) dataset [Teboul et al., 2010] and the eTRIMS dataset [Korc and Förstner, 2009]. The ECP dataset contains 104 annotated, rectified and cropped images of Haussmannian style façades in Paris, including seven labels: *window, door*, wall, sky, shop, balcony, roof, and chimney. The images are taken in varying light and weather conditions, which poses a good challenge for the proposed method. However, the effectivity of the automatic rectification method cannot be measured since the images are pre-rectified. Hence, the eTRIMS dataset containing 60 annotated, non-rectified (i.e. perspective view) images are added to the evaluation. The eight-class variant of the dataset contains labels: *window, door*, building, car, pavement, road, sky and vegetation. The evaluation of floor count performance is conducted on eTRIMS before and after the automatic rectification.

Wild dataset The aforementioned datasets exhibit different image characteristics, which makes for a well rounded collection of datasets for façade parsing and testing of the automatic rectification. However, none of the datasets are designed for floor count determination, nor do they reflect the wide variety of different architectural styles or building types that are encountered in real-world SVI.

In development of the floor count determination prototype, it was found the floor count values are imbalanced or limited in range. This can lead to bias in the floor count method or limit thorough assessment. Another notable finding is a certain degree of homogeneity of architectural styles and building types within each dataset. Although combining the datasets increases heterogeneity, the combination of datasets still lacks taller buildings with more than seven storeys, modern architecture (e.g. predominance of glass), or challenging SVI-related anomalies. An example of the latter is related to a limitation of road-bounded SVI acquisition, which can result in moderate to severe scaling and perspective-angle variance.

For reasons mentioned, a small custom dataset is created that includes taller buildings (up until 14 storeys), modern architecture with a predominance of glass and various perspective-angles and image scales are included. The façades are cropped to contain only a single building, as multiple-building images are already covered in the eTRIMS dataset. The images are mostly collected through GSV, and some from Mapillary for comparison. For GSV images, the automatic pose estimation is used as much as possible to replicate an automatic pipeline.

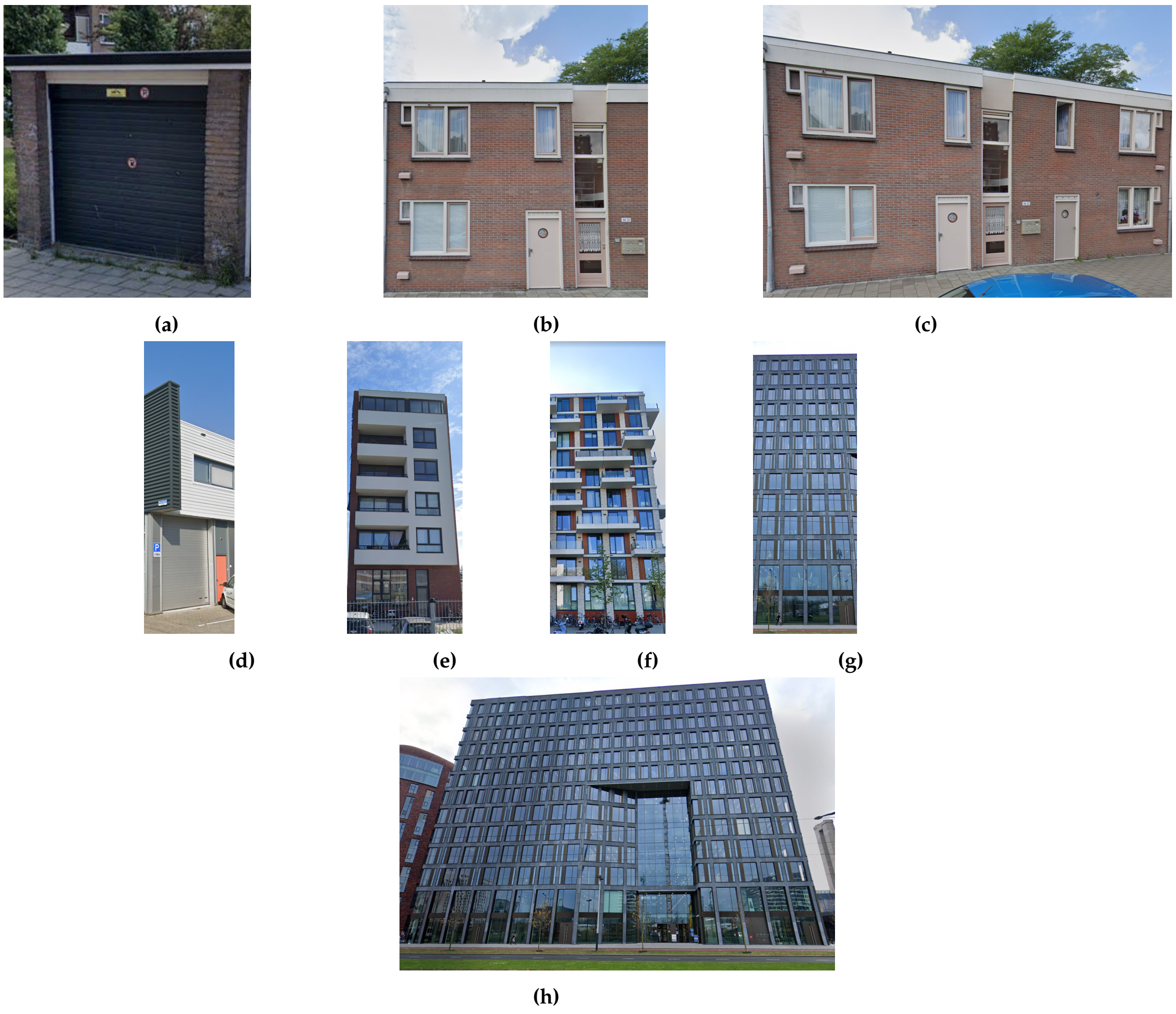


Figure 5.3.: Some examples of the custom wild images dataset, including taller buildings (with more storeys), modern architecture and varying image scales and perspectives.

5.3. Mask R-CNN training and optimisation experiments

The Mask R-CNN model is trained on the Amsterdam Facade dataset (see Section 5.2). A 80/20 split is used for training and test subset of the dataset. For optimising the training routine of the model, both manual and automatic hyperparameter tuning are explored.

Manual hyperparameter tuning was performed to get familiar with a few the Mask R-CNN hyperparameters. Most notable are learning rate (LR), weight decay and warm-up iterations. These were found to affect the training the most. LR, arguably the most important hyperparameter, is the factor that controls how much the model changes at each iteration while minimising a loss function [Goodfellow et al., 2016]. With all other hyperparameters kept constant, an LR between 0.00025-0.0001 is found to be effective.

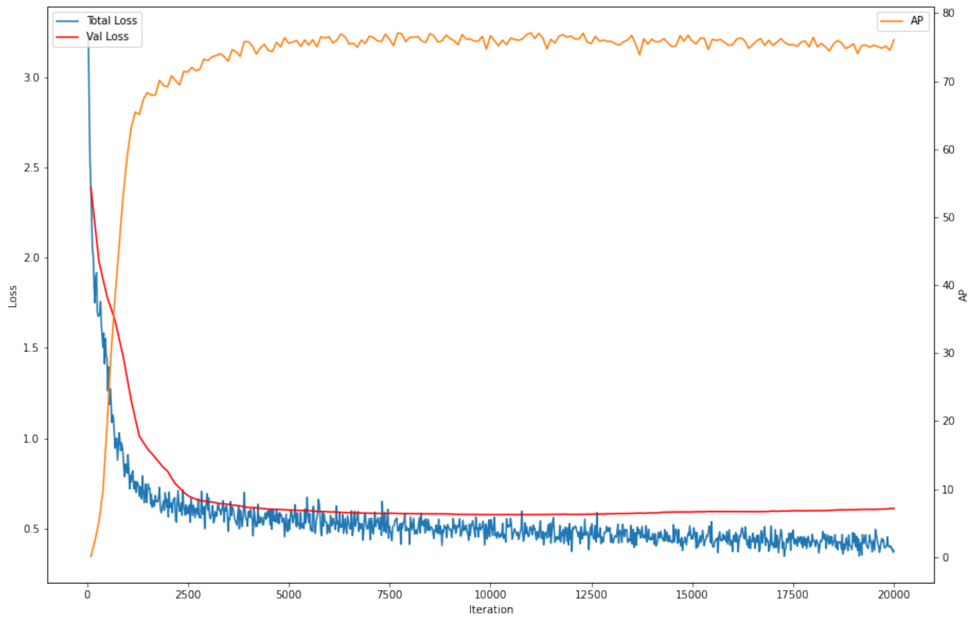


Figure 5.4.: The full learning curve is plotted in manual hyperparameter tuning, with a LR of 0.0001 for 50 epochs. With a batch size of 2, one epoch is 392 iterations. Note how the model starts overfitting slightly beyond 25 epochs.

The problem found with this approach is that more than 10K iterations or 25 epochs are needed for convergence. It was found that weight decay and warm-up iterations can help to reach convergence in less iterations. Weight decay is the incremental decrease of LR over a given set of checkpoints, and warm-up iterations is a routine where the LR is built up from 0 to the base LR in the first few iterations.

Automatic hyperparameter tuning Since resources are limited with the Google Colaboratory Pro platform, especially when training with the premium GPU as specified, only so many training routines can be performed before hitting computing limits. In search for a better set of hyperparameters that result in better performance with less training, a more sophisticated approach for hyperparameter tuning is employed. The Weights & Biases library [Biewald, 2020] is used for an automated hyperparameter search, to explore the space of possible models. Instead of the classic grid search or parameter sweep, where a manually specified subset of hyperparameters searched exhaustively, Bayesian and random search are employed.

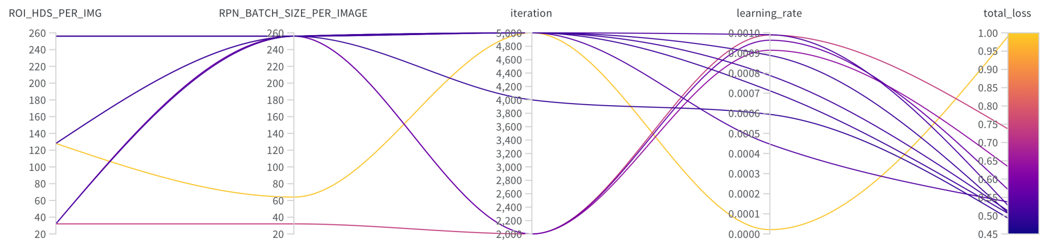


Figure 5.5.: The sweep configuration chart that gives an overview of the hyperparameters searched, and which combination resulted in the lowest total loss.

A marginal increase in performance is achieved with automatic tuning compared to manual tuning as shown in Table 5.1. The performance difference as a result of manual versus automatic tuning on floor count determination performance is measured in Chapter 6.

Table 5.1.: Performance comparison of façade parsing model, manual versus automatic hyperparameter tuning. The automatic tuning resulted in marginally better performance in terms of AP in both object detection (bbox) and instance segmentation (segm).

Tuning	AP^{bbox}	AP^{segm}
Manual	76	77
Automatic	**78**	**80**

Backbone comparison Two conventional backbones, ResNet-50 FPN or ResNet-101 FPN, are trained and compared in terms of AP in Table 5.2. In our experiments, the ResNet-50 FPN performed marginally better than the ResNet-101 FPN while also being computationally cheaper to train. Hence, it is the backbone used in further experimentation unless specified otherwise.

Table 5.2.: Comparison of the ResNet-50 FPN and ResNet-101 FPN backbone performance in object detection (bbox) and instance segmentation (segm) in terms of AP. Their best found configuration was found using the same automatic hyperparameter tuning protocol.

	AP^{bbox}	AP^{segm}
ResNet-50	**77.81**	**79.79**
ResNet-101	77.27	79.16

6. Results and Analysis

6.1. Image rectification experiments

6.1.1. VP estimation

The results of image rectification by VP estimation and image warping, are shown in Figure 6.1. The results are paired with the original, so the difference can clearly be observed. The rectification results in the top row are reasonable, and the regularity assumption is enforced. However, note how the rectification result 6.1d is substantially warped which changes aspect ratio. The results in the bottom row are worse. The image in 6.1e is from far away, some perspective angle and the façade is tall. A challenging façade, but not extreme. Yet, the image rectification result thereof is extreme. The warping of the image is severe, such that the rectified image is worse than the original and the regularity assumption is not enforced. The rectification result in 6.1h has improved regularity, but also in this example the warping deteriorated the image quality.

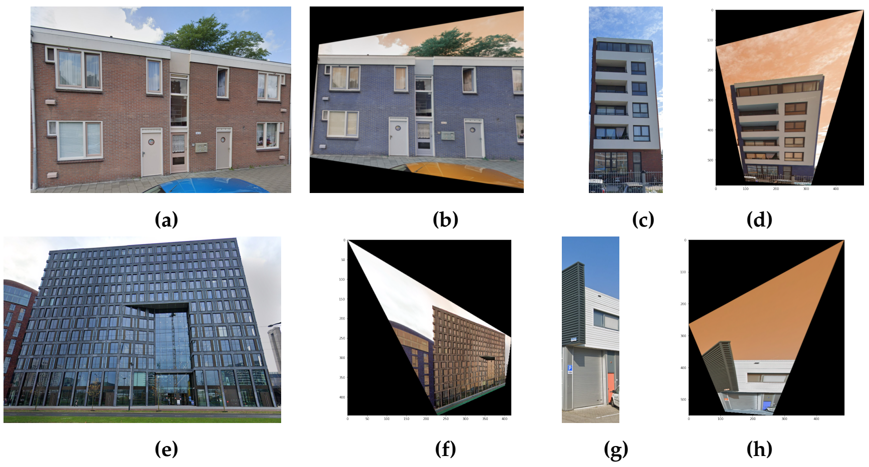


Figure 6.1.: Image rectification results by VP estimation.

6.1.2. RANSAC and homography transform

The image rectification results by direct homography transform are shown in Figure 6.2. The top row are the same images shown in the previous image rectification method (see Figure 6.1). Most of the results by this method are impressive, when considering enforcement of the regularity assumption, varying input image scales and qualities, and image warping whilst keeping aspect ratios within reason.

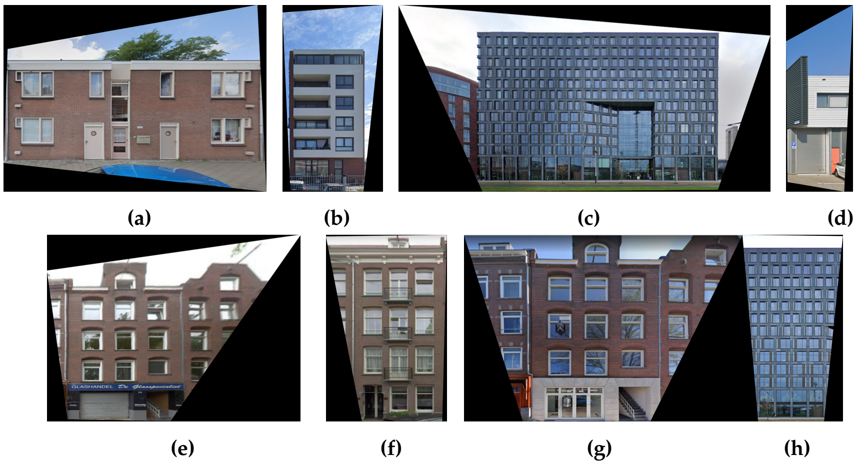
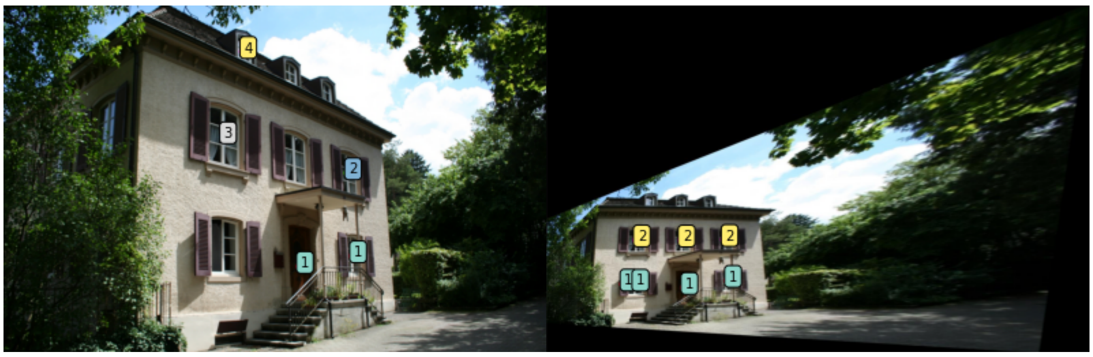


Figure 6.2.: Image rectification results by direct homography transform.

It is only in cases where the perspective view angle around $> 30°$, that this image rectification method starts introducing severe warping that interferes with the aspect ratio of the image. In the example shown in Figure 6.3, the warping causes the model to fail to detect the top window. That said, it also improves detection quality on the first two storeys, which is due to the windows appearing more rectangular in the rectified image.



(a) Perspective view image. **(b)** Rectified image.

Figure 6.3.: Here we can see that in cases of large angle perspective view, the rectification distorts the image severely and the top window is no longer detected. This is due to the model having mediocre performance on smaller objects.

6.2. Façade parsing results

Overall, with a confidence threshold of 0.8 in inference, good façade parsing results are achieved with the Mask R-CNN as illustrated in Figure 6.4. The results look promising as input for the next stage, floor count determination.

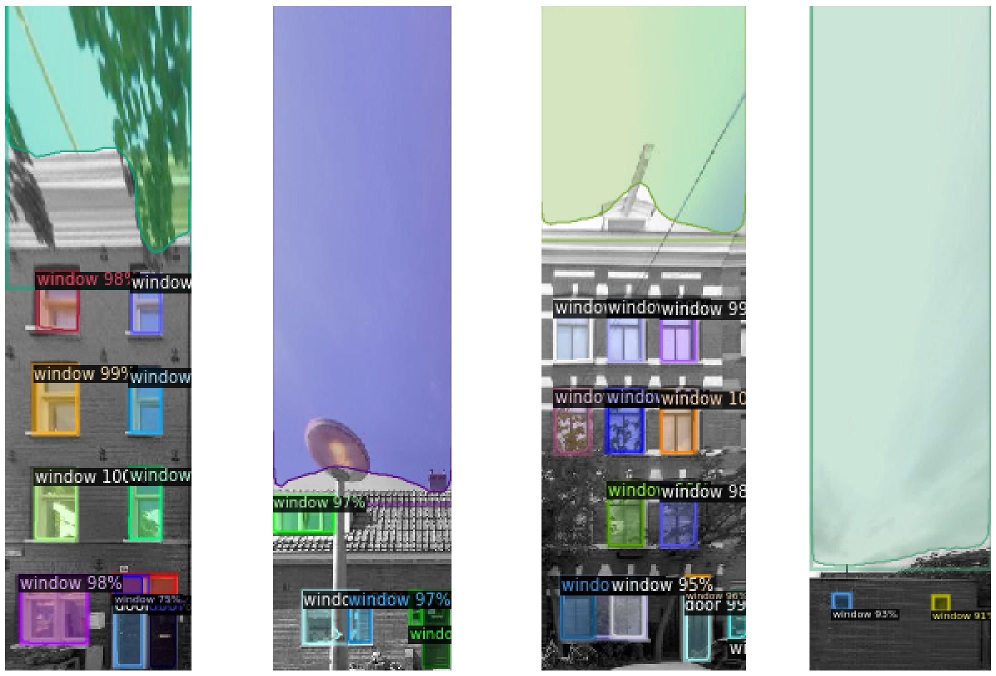


Figure 6.4.: A few façade parsing results with Mask R-CNN. A confidence threshold of 0.8 results in few false positives, still detecting and segmenting instances that are partly occluded.

6.2.1. Detection and segmentation evaluation

For the side-by-side evaluation of the detection and segmentation results, inference is performed on the test subset of the Amsterdam Facade dataset with the automatically tuned model. The object classes evaluated are window, door and sky in Table 6.1. The latter is included as background-class, increasing the performance of the model overall.

Table 6.1.: A comparison of the object detection and instance segmentation performance on classes window, door and sky. The performance is measured in terms of class-specific AP.

	Window	Door	Sky
Detection	71	67	95
Segmentation	72	69	98

The Mask R-CNN performs better on the majority class (window) in both detection (71%) and segmentation (72%), compared to minority class (door) in detection (67%) or segmentation (69%). This is expected with a substantial class imbalance in the training dataset, though the marginal performance difference is within reason.

Table 6.2.: A comparison of the object detection and instance segmentation performance across different detection sizes and thresholds.

	AP_s	AP_m	AP_l	AP75	AP50	AP
Detection	55	71	95	88	96	78
Segmentation	57	73	98	89	97	80

From Table 6.2 we can conclude that the façade parsing model performs significantly better on larger objects (APm, APl) compared to smaller object (APs). This is problematic for our automatically rectified datasets, as in some cases severe warping causes object sizes (e.g. windows) to decrease in size. If the model fails to detect a whole row of windows, the subsequent floor count determination method will detect a storey less. This is exemplified in the Figure 6.3.

6.3. Floor count estimation experiments

6.3.1. FloorLevel-Net evaluation

The reproduction of FLN was successful, and run on a subset of the CMP dataset and the complete ECP dataset. The results are shown in Figure 6.5. In the top row, we observe the results on the CMP dataset. FLN is partly trained on the CMP dataset, so the results are good as expected. However, note how the network detects a false positive in 6.5d. In the bottom row, we observe the results on the ECP dataset. The results are poor, and unusable for floor count detection. An interesting finding, is that in most cases the model predicts between 3 and 5 floor lines, regardless how many storeys the façade has. This indicates that the network has a strong bias, and training dataset should be diversified for the goal of counting storeys.

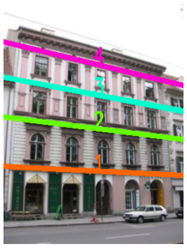
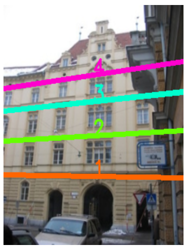
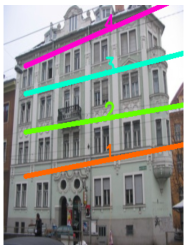
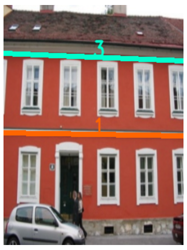
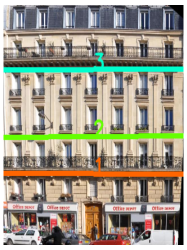
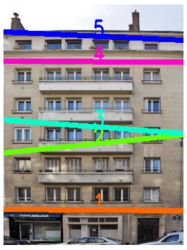
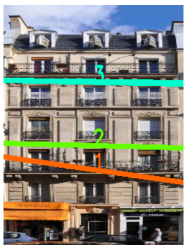
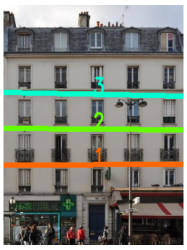

(a)

(b)

(c)

(d)

(e)

(f)

(g)

(h)

Figure 6.5.: Results of the FloorLevel-Net reproduction, run on the CMP (top) and ECP (bottom) dataset.

6.3.2. Bivariate vertical clustering evaluation

The bivariate vertical clustering experiments are run on the Amsterdam Facade datasets, as shown in Figure 6.6. In the top row are successful examples. In the bottom row are a few failure cases, which exposes the limitations of the approach. Generally, the method struggles with detections that are not vertically aligned well, or smaller vertical differences from intraclass variance. As an example, in 6.6h we see that the small windows on top of the door are detected as a separate cluster from the door and windows to the left. All of these should belong to the same cluster. Varying in point density did not help, as it would fix some cases but cause new issues in others. Notice how clusters on similar vertical position, can still be put into a different cluster due to horizontal separation. This observation led to the isolation of the y-data.

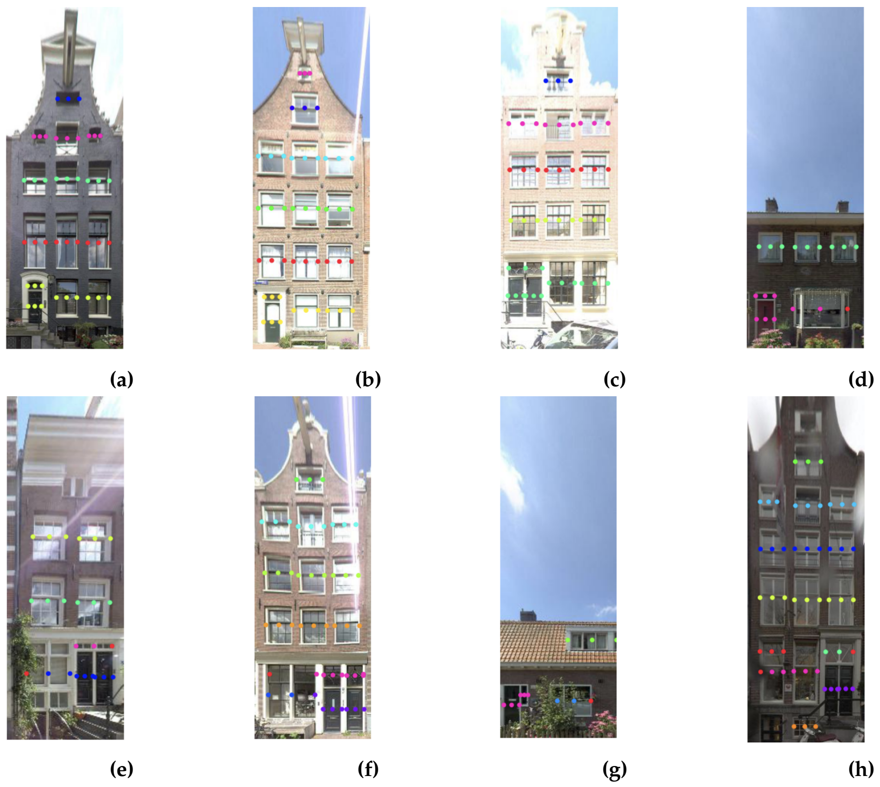
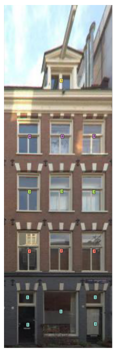


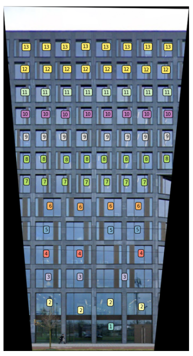
Figure 6.6.: Results of the bivariate approach. On the top row are results that are correct. The bottom row shows failure cases.

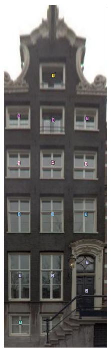
6.3.3. Univariate vertical clustering evaluation

The experiments of univariate vertical clustering were most extensive, as the method showed the most potential early on. In Figure 6.7 are the results of floor count estimation across five different datasets. The method shows it works with different datasets, meaning it is robust to varying image quality and characteristics. From the comparison on the bottom row, we can observe the impact of the rectification pre-processing step on the detection and floor count estimation performance. Weaknesses are also exposed in the bottom row, such are occlusions (red car) and surrounding façades that are also detected thus resulting in an incorrect floor count.

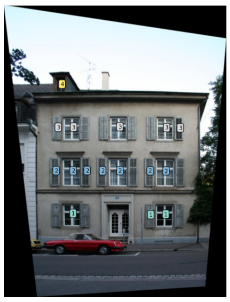

(a)

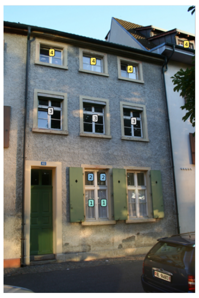
(b)


(c)


(d)

(e)


(f)


(g)

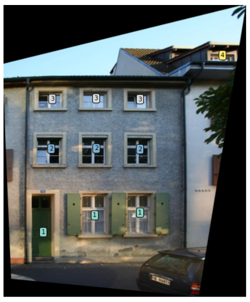

(h)

Figure 6.7.: The results of univariate approach to floor count estimation across five different datasets: Amsterdam Facade (6.7a, 6.7d), ECP 6.7b, eTRIMS (6.7e, 6.7g) en eTRIMS rectified (6.7f, 6.7h).

6.3.4. Floor count performance evaluation

The evaluation of floor count determination performance is summarised in Table 6.3 and 6.4, to compare between the manually and the automatically optimised façade parsing methods, respectively. Each table contains the summary of metrics across all datasets mentioned in Section 5.2, of the detection-based, segmentation-based and normalised segmentation-based methods.

Table 6.3.: Floor count determination performance from the manually optimized façade parsing model, across all datasets in terms of MAE, mean error, standard deviation (σ) error, F1-score weighted average and accuracy. Best results in **bold**, $\downarrow$= lower is better, $\uparrow$= higher is better.

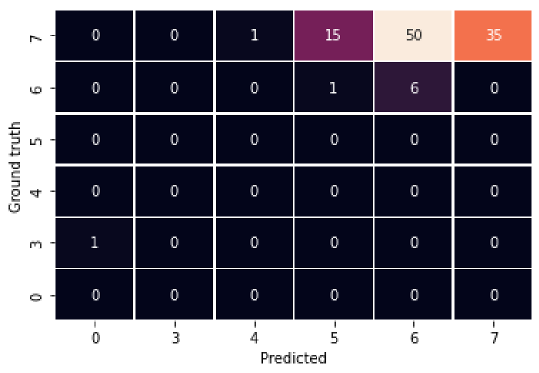
		Ams. Façade	ECP	eTRIMS	eTRIMS rect	wild	wild rect
Detection based data	MAE $\downarrow$	**0.17**	0.80	0.65	0.5	2.24	2.36
	ME $\downarrow$	**-0.17**	-0.80	0.32	**0.17**	-1.57	-2.18
	σ error $\downarrow$	**0.38**	0.74	0.93	0.74	3.78	3.45
	f1 $\uparrow$	**0.83**	0.49	0.49	0.53	0.21	0.35
	Accuracy $\uparrow$	**0.83**	0.38	0.48	0.53	0.24	0.32
Segmentation based data	MAE $\downarrow$	**0.66**	0.88	1.92	7.9	2.10	2.86
	ME $\downarrow$	**0.30**	-0.86	1.68	7.73	-0.76	0.32
	σ error $\downarrow$	1.77	**0.70**	4.31	20.65	4.05	5.06
	f1 $\uparrow$	**0.63**	0.38	0.36	0.41	0.44	0.19
	Accuracy $\uparrow$	**0.64**	0.28	0.35	0.38	0.43	0.23
Segmentation based data (normalised)	MAE $\downarrow$	**0.20**	0.95	0.60	0.65	1.90	2.23
	ME $\downarrow$	**-0.20**	-0.95	0.30	0.42	-1.90	-2.14
	σ error $\downarrow$	**0.40**	0.77	0.83	1.11	3.39	3.37
	f1 $\uparrow$	**0.79**	0.36	0.49	0.50	0.48	0.28
	Accuracy $\uparrow$	**0.80**	0.27	0.48	0.50	0.48	0.32

Table 6.4.: Floor count determination performance from the automatically optimized façade parsing model, across all datasets in terms of MAE, mean error, standard deviation (σ) error, F1-score weighted average and accuracy. Best results in **bold**, $\downarrow$= lower is better, $\uparrow$= higher is better.

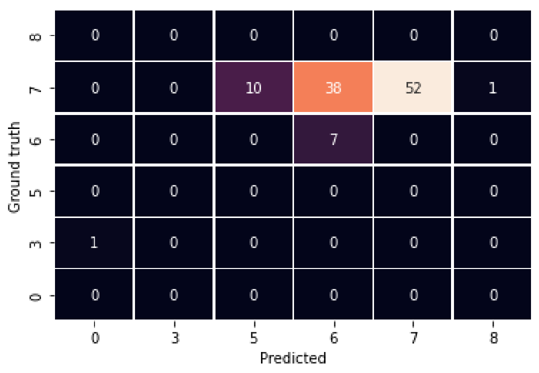
		Ams Façade	ECP	eTRIMS	eTRIMS rect	wild	wild rect
Detection based data	MAE $\downarrow$	**0.19**	0.57	0.45	0.34	2.32	2.86
	ME $\downarrow$	-0.19	-0.55	0.15	**0.03**	-1.68	-2.00
	σ error	**0.42**	0.71	0.84	0.64	3.46	3.21
	f1$\uparrow$	**0.83**	0.65	0.67	0.69	0.23	0.23
	Accuracy $\uparrow$	**0.82**	0.54	0.67	0.70	0.23	0.24
Segmentation based data	MAE $\downarrow$	1.0	**0.86**	2.62	3.28	2.55	5.33
	ME $\downarrow$	0.66	**-0.33**	2.42	3.15	0.36	3.90
	σ error $\downarrow$	3.20	**2.40**	5.06	7.44	5.35	10.83
	f1$\uparrow$	**0.63**	0.61	0.35	0.51	0.35	0.30
	Accuracy $\uparrow$	**0.63**	0.49	0.35	0.50	0.32	0.29
Segmentation based data (normalised)	MAE $\downarrow$	**0.23**	0.66	0.47	0.48	2.00	2.10
	ME $\downarrow$	-0.17	-0.66	0.20	**0.15**	-1.91	-1.90
	σ error $\downarrow$	**0.48**	0.71	0.80	0.80	2.69	3.05
	f1 $\uparrow$	**0.77**	0.59	0.64	0.58	0.26	0.27
	Accuracy $\uparrow$	**0.78**	0.48	0.63	0.58	0.27	0.29

As expected, the method performs the best for the test dataset the façade parsing model is trained on and the floor count determination method is developed with. The detection-based method performs best with an MAE of 0.17 and accuracy of 0.83, followed by the normalised segmentation-based method with an MAE of 0.20 and accuracy of 0.80. The model tends to undershoot (negative ME scores), which can be due to the strict confidence threshold of 0.8 in the inference module. Considering detection-based and normalised segmentation-based for both manual and automatic optimised façade parsing methods, the results for all datasets but wild, the MAE is under 1 storey.

What is interesting regarding the comparison between rectified and non-rectified data, is that results vary both positively and negatively. This can be explained by the fact that although rectification enforces regularity, the warping as seen in Figure 6.3 can cause severe distortion causing the model to fail to detect crucial windows or doors.



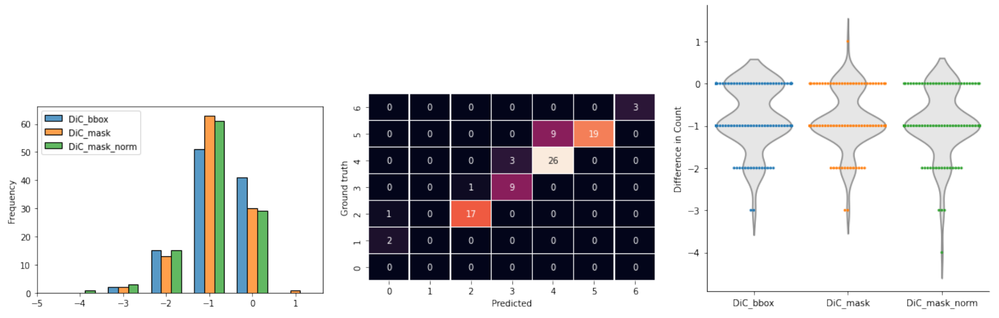
(a) Confusion matrix of ECP dataset, with a manually optimised façade parsing network.



(b) Confusion matrix of ECP dataset, with a automatically optimised façade parsing network.

Figure 6.8.: The effect that automatic hyperparameter tuning had was surprising. as the performance gain in terms of AP were marginal. It means that manual hyperparameter tuning is relevant to the training and test dataset, but automatic hyperparameter tuning can improve the generalisability of your model.

When comparing Table 6.3 and 6.4, a general difference is that the former (manually tuned) excels on the Amsterdam Facade dataset, whereas the latter (automatically tuned) performs more evenly. Again considering the detection-based and normalised segmentation-based methods, the automatically tuned model performs significantly better on ECP, eTRIMS and eTRIMS rectified and only marginally worse on the Amsterdam Facade and wild datasets. The comparison of the confusion matrices in Figure 6.8 from experiments on the ECP dataset show from mostly undershooting floor count determination in Figure 6.8a to mostly correct floor count determination in Figure 6.8b.



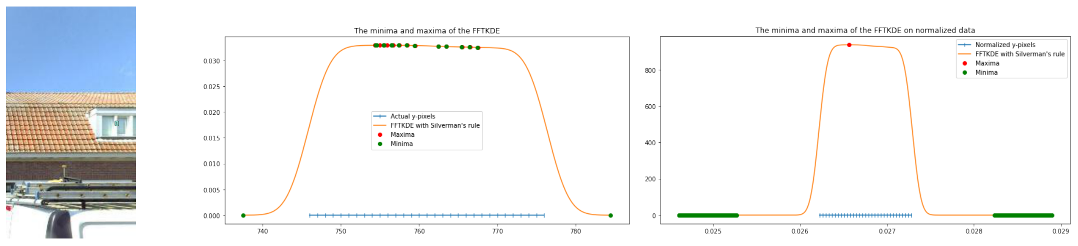
(a) (b) (c)

Figure 6.9.: The best results visualised in a histogram in 6.9a, a confusion matrix in 6.9b and a violin plot in 6.9c. Note, DiC (difference in count) is equivalent to ME.

Concluding the results in Figure 6.9, the results look promising with its main problem being undershooting the floor count by one storey. It seems a marginal increase in façade parsing performance already reduced the undershooting problem significantly.

6.3.5. Failure cases

It is clear from comparing segmentation-based and normalised segmentation-based results that normalisation is a necessary processing step for the floor count determination method to work properly, which is especially noticeable in the eTRIMS rectified results in Table 6.3. The phenomenon can be observed in Figure 6.10.



(a) (b) (c)

Figure 6.10.: The automatic ISJ bandwidth optimisation method finds many maxima in non-normalised data, as the data is relatively spread out for the one storey (see 6.10c). Normalisation "compresses" the data, such that a single maxima is determined.

The undershooting from floor count determination estimation can be due to non-detection of smaller windows (e.g. of semi-basements or attic windows), or when façades have a shop as ground floor. Note that if the minority class *door* was detected in Figure 6.11a, an additional storey would have been counted thus would have a correct floor count estimation. Alas, a car are occluding the door partially, which may confuse the model as the door is non-rectangular as a result.

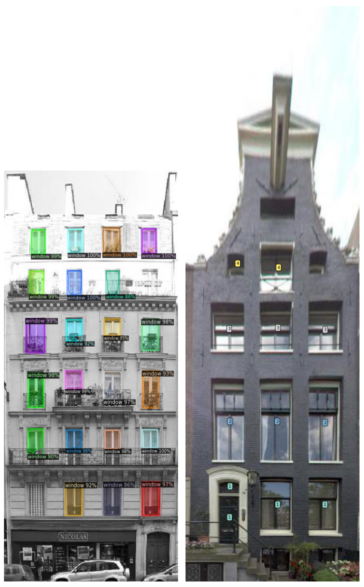


Figure 6.11.: The general undershooting of the floor count determination method is often due to non-detection of smaller windows or unknown class shop.

7. Conclusions

7.1. Research overview

The focus of this thesis was to develop an experimental pipeline that explores how automatically extract floor count information with the use of façade parsing from street view imagery (SVI). Other intentions that were set during the study include large scale application, using open-source data and software. The interest and relevance of the research in the field of geomatics is to tap into a vast and fast growing form of spatial data, SVI. This can help fill the gap of missing floor count information in GIS datasets and 3D city models, that is useful to many applications in urban analytics. Deep learning has shown its potential to gain new insights from data and achieve feats in fields such as computer vision, which gave reason to believe it could solve a rather intuitive human task that is known as counting the number of storeys in images. This raised the following question:

How to determine floor count in an image with the use of learning-based façade parsing?

Façade parsing detects and segments the façade elements, which can be used to find vertical clusters that corresponds to storeys, under the assumption that façade design harnesses regularity. The best method to find vertical clusters is by means of computing the kernel density estimation function, and perform maxima finding on the function which corresponds to the floor count. Image rectification can improve the floor count performance, though the façade parsing model should be trained to detect smaller objects as well in cases of significant warping of the image.

Floor count evaluation on the full pipeline show an 83 % accuracy and a MAE of 0.17 on detection-based data. On (normalised) segmentation-based data, an accuracy of 80 % and MAE of 0.20 are achieved. Considering the relatively small dataset used for training and development, and with no discrimination made between lower or taller buildings, the method proves façade parsing is a promising approach for extracting floor count information from SVI. The following sub-questions will provide more nuance to different aspects of the main question.

1. **How can we use deep learning for façade parsing?**

   Deep learning has successfully been applied to achieve façade parsing. The Mask R-CNN framework provided both object detections and instance segmentations. Important considerations are dataset selection, transfer learning and hyperparameter tuning. The dataset selected to train and test the network was as close as possible to the application in mind. A pre-rectified dataset from an SVI source, with the distortions that are to be expected in similar data. Transfer learning is crucial when little data is available, as it allows for the use of a general trained network to be re-trained on a relatively small dataset. If smaller dataset fits the target application, good results can be achieved. Finally, hyperparameter tuning is an essential process to optimise learning from the dataset. Manual hyper parameter tuning can provide insights on what parameters are most responsive in the model given the data. Preliminary findings can be transferred to automatic hyperparameter tuning to further improve performance. Furthermore, automatic hyper parameter tuning is beneficial for model generalisation and making the process more reproducible.

One of the main limitation is related to undershooting, that is caused by the façade parsing model to fail to detect small or occluded windows/doors. This shows that major improvements in the floor count determination performance can be made if the façade parsing model is trained to detect smaller objects, and non-rectangular windows and doors (due to occlusion).

2. **How should façade parsing outputs be processed for floor count determination?**

   In order to prepare façade parsing outputs for floor count determinations with density estimation, the outputs should be processed into pixel coordinate-data. First, the most relevant classes (e.g. window and door) that are available in the training dataset should be filtered for both detection-based and segmentation-based data. For both outputs, the x and y dimension have to be split as well. The detections contain spatial information in the form of spatial extent. In this study, single y-point extraction was the chosen method to represent each detection, as this allowed for simple input for the subsequent KDE. It was found empirically that the top-point was the most robust point for floor count determination performance. The segmentation instances are bitmaps that need to be converted to pixel coordinate-data. Each segmentation bitmap is concatenated, and the result is split into a x and y array. The y array can be passed onto the KDE. However, findings show that the segmentation-data needs normalisation for better floor count determination performance via maxima finding.

3. **How can we vertically group façade parsing outputs, such that the groups represent countable storeys?**

   For the vertical grouping of façade parsing outputs, a univariate distribution analysis is performed with KDE on both detection- and segmentation-based data. The two most important parameters for KDE optimisation are bandwidth and kernel. There are both manual and automatic optimisation of bandwidth. The empirical findings show that bandwidth is more crucial, as long as a kernel with continuous property is chosen. From the KDE, maxima are extracted by traversing the KDE output with a "greater than" comparator. A general finding is that the method worked better on detection-based data, across all datasets. One reason is the fact that the KDE was manually tuned on the detection-based data, whereas for the segmentation-based data an automatic method was chosen for bandwidth optimisation. The difference in optimisation shows that manual tuning can leverage domain knowledge. However, the performance difference is acceptable considering the time saved in the optimisation process. The time-saving aspect can contribute to more widespread applicability of the method in terms of different datasets, and used as a tool for method calibration onto other datasets.

4. **What pre- and/or post-processing steps do we need to improve floor count determination performance?**

   In terms of pre-processing, our findings show that image rectification improves performance in the eTRIMS dataset marginally overall, whereas the performance on our wild dataset is marginally worse (excluding non-normalised segmentation-based data). The reason why rectification did not lead in a substantial improvements in all cases is related to façade parsing performance. The rectification introduces more distortion, in some cases severe. The distortions can make object in the images smaller, which is what the façade parsing model scores lowest on in terms of AP. A possible solution is re-scaling of the rectified images, another solution is to add data augmentations to the training procedure to better deal with different scaling. More improvements can be made in the data annotations as well. The dataset contains many train-images that lack the annotation of small windows. This further explains the mediocre performance of the façade parsing model on smaller objects. Moreover, some of the annotated images lack annotations on neighboring façades. This can confuse the model, as the pixels are very similar to the ones annotated, but will falsely be penalised when predicted.

In terms of post-processing, the results show that segmentation-based façade parsing outputs need normalisation of the pixel-coordinate data to result in a KDE function that is not sensitive to noise.

7.2. Discussion

7.2.1. Contributions

This research builds upon learning-based façade parsing works, and contributes by combining existing methods to approach the problem of floor count from SVI in an alternative way. None of the existing works on (indirect) floor count determination with learning-based façade parsing, were evaluated on benchmark datasets before.

- Deep learning pipeline for façade parsing that can perform both detection and segmentation, trained on a open-source domain specific dataset, and floor count estimation module which are all published for further use and development.
- A small dataset that can be used as starting point for dedicated SVI facade parsing and floor count dataset, along with floor count annotations for well known façade image datasets.
- Review of and experimentation on other methods that can be useful for how floor count determination can be achieved from SVI and/or other data.

7.2.2. Limitations

During research and development of the floor count determination method, weaknesses and limitations of the approach were exposed, which we outline as follows:

- **Dataset**: The dataset used was not designed with a floor count application in mind. The dataset was too uniform in the sense of architectural and geographic variability, image quality, façade morphology. Neither of the datasets used had floor count ground-truth data available, and floor count annotations were made manually by visual inspection (by the author) of the images. However, cases of ambiguity, bias and non-complete knowledge of the buildings make some of the annotations debatable.

  Another limitation of the dataset used was inconsistent window and door annotations, from smaller windows or objects on neighboring façades. This observation was made when assessing the results, unfortunately too late in the process to improve the annotations. It is highly recommended to improve the annotations before using the Amsterdam Facade dataset.
- **Breadth of research**: Since the exploratory nature of this thesis and the breadth of development options, there are a couple of investigated methods that were shelved

or discontinued due to time constraints in the project. In retrospect, the populated horizontal line clustering could be further improved, quantitatively assessed and tested on other datasets. The RANSAC method described by Håbrekke and Nordstad [2022] should also be employed and evaluated on the same data, to get a better understand how these works compare.

- **SVI coverage and practicality of large scale application**: During the creation of a SVI dataset and selection of other SVI /façade datasets, literature and observations indicated there are significant limitations in the use of SVI for analysing building data. Hence, the problem of counting storeys was simplified. For instance, not all buildings are close to an open road, many areas in developing countries are not covered yet, buildings narrow streets and high risers might be practically impossible to capture completely. Another practical issue, which is not formally investigated in this research, is the use of APIs to create SVI datasets, which come with its own set of challenges. One of which is targeting the façade of interest and isolating it from the rest of the image.
- **Computation limitations**: There was no formal budget for cloud computing, meaning small scale cloud computing services were rented. The computation time was limited as a result, which means the training routines (both manual and automatic) were conservative.

7.2.3. Recommendation for future work

Based on the limitations and observations made during this research, we formulate the following recommendations:

- **Dataset creation**: Since a small performance gain in façade parsing performance, given that most errors are caused by failure of detection, and that the dataset used for training has the aforementioned limitations, it is strongly suggested to create a dedicated SVI façade parsing dataset with an API from an open-source (e.g. Mapillary, OSM/Overpass Turbo or otherwise) that is geographically and socio-economically stratified on a global scale, has a high degree of architectural and structural variability, contains as much ground-truth data on building attributes as possible, and regards all quality aspects mentioned in Section 2.3 or Hou and Biljecki [2022]. It is strongly recommended to open source the dataset and code to create it.

  If one is interested in working with the GSV API, one could apply for Cloud Credit funding for academia [Google, 2023]. However, terms and regulations should be thoroughly considered.
- **Automatic façade retrieval and isolation** In relation to dataset creation, a good study would be on automatic acquisition of SVI data with an API and targeting the correct building by façade isolation as mentioned in Ogawa et al. [2021].
- **Conduct a literature review**: on the definitions, regulations, and architectural and structural design of storeys, and storey variants, such as semi-basements, half-storeys, mezzanines, attics, roof shapes, and so on. In addition, conduct a corresponding review on GIS standards that regard storeys and (its relation to) floor count.
- **More sophisticated model design**: FloorLevel-Net reproduction was not a complete failure. The model was very complex and time consuming to reproduce, but inference was run successfully. Training the model was not accomplished, and it would be

interesting to investigate the method further for its capability to detect floor-level lines in perspective view SVI. The initial results shown do reveal there is some potential for use in floor count determination from SVI, and deserves a quantitative assessment after training on appropriate data.

Another recommendation from using FLN, is to investigate more sophisticated façade parsing models that employ height and/or symmetry attention modules. The modules guide the learning process, which can result in achieving higher level semantic beyond singular elements.

A final recommendation in this context is a research focus on model (inference) speed. Very few papers, if any, in relation to façade parsing for urban analytics and related applications mention real-time performance of the model. With large scale applications in mind, speed is essential. Another opportunity is real-time acquisition of information as the street view car patrolling around cities.

- **Improve vertical clustering**: The manual and automatic optimisation of the KDE and FFTKDE can be further optimised. For instance, a parameter search can be performed in maximisation of accuracy or minimisation of MAE. The automatic bandwidth optimisation can give a good initial guess, and used to configure a range of bandwidth for the parameter search. Other optimisation method to improve vertical clustering is regularisation of the detections, as described by Wang [2022].

A. Appendix

A.1. Experimental design and development

Page left intentionally blank, figure on the next page.

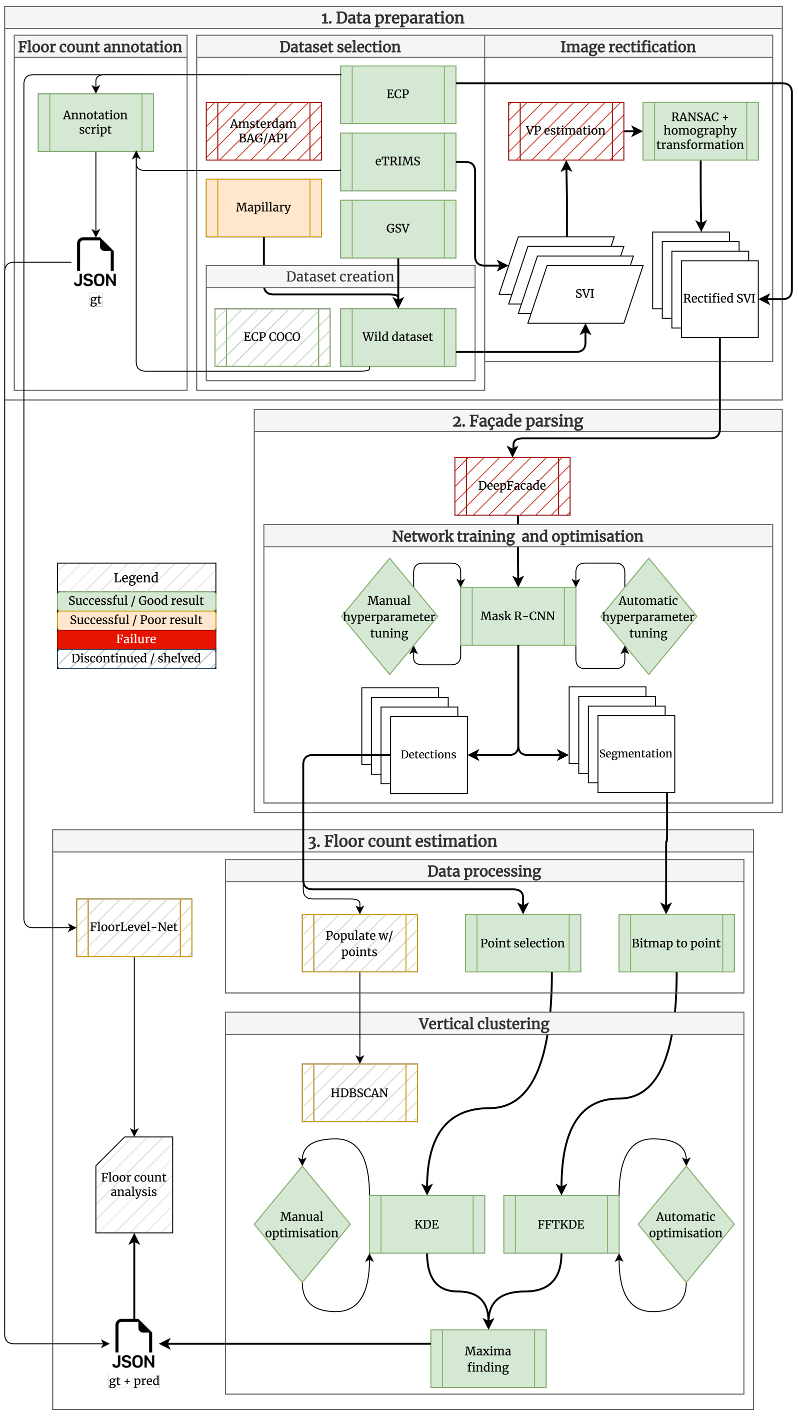


Figure A.1.: The complete and detailed overview of the proposed three-staged method and developments steps, which consists of: 1. data preparation, where floor count annotation files are made, datasets are reviewed, selected and created, and images are rectified; 2. façade parsing, where two deep learning networks are investigated, of which Mask R-CNN is trained and optimised; 3. floor count estimation, where detections and segmentations are processed, and followed by vertical clustering experiments, of which KDE/FFTKDE are further developed and optimised. Finally, all successful experiments are chained into a full pipeline (follow bold lines), then tested on multiple datasets and analysed.

Bibliography

Affara, L., Nan, L., Ghanem, B., and Wonka, P. (2016). Large scale asset extraction for urban images. In *European Conference on Computer Vision*, pages 437–452. Springer.

Albawi, S., Mohammed, T. A., and Al-Zawi, S. (2017). Understanding of a convolutional neural network. In *2017 international conference on engineering and technology (ICET)*, pages 1–6. Ieee.

Anguelov, D., Dulong, C., Filip, D., Frueh, C., Lafon, S., Lyon, R., Ogale, A., Vincent, L., and Weaver, J. (2010). Google street view: Capturing the world at street level. *Computer*, 43(6):32–38.

Aydin, Y. C. and Mirzaei, P. A. (2020). A novel mathematical model to measure individuals' perception of the symmetry level of building facades. *Architectural Engineering and Design Management*, pages 1–18.

Ayenew, M. (2021). Towards large scale façade parsing: A deep learning pipeline using mask r-cnn.

Biewald, L. (2020). Experiment tracking with weights and biases. Software available from wandb.com.

Biljecki, F. (2017). *Level of detail in 3D city models*. PhD thesis, Delft University of Technology, Delft, the Netherlands.

Biljecki, F. and Ito, K. (2021). Street view imagery in urban analytics and gis: A review. *Landscape and Urban Planning*, 215:104217.

Botev, Z. I., Grotowski, J. F., and Kroese, D. P. (2010). Kernel density estimation via diffusion. *The annals of Statistics*, 38(5):2916–2957.

Campkin, B. and Ross, R. (2016). Negotiating the city through google street view. In *Camera Constructs*, pages 169–180. Routledge.

Canny, J. (1986). A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698.

Chen, F.-C., Subedi, A., Jahanshahi, M. R., Johnson, D. R., and Delp, E. J. (2022). Deep learning–based building attribute estimation from google street view images for flood risk assessment using feature fusion and task relation encoding. *Journal of Computing in Civil Engineering*, 36(6):04022031.

Chu, C.-Y., Henderson, D. J., and Parmeter, C. F. (2015). Plug-in bandwidth selection for kernel density estimation with discrete data. *Econometrics*, 3(2):199–214.

Cohen, A., Oswald, M. R., Liu, Y., and Pollefeys, M. (2017). Symmetry-aware facade parsing with occlusions. In *2017 International Conference on 3D Vision (3DV)*, pages 393–401. IEEE.

Dai, J., He, K., and Sun, J. (2016). Instance-aware semantic segmentation via multi-task network cascades. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3150–3158.

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. Ieee.

Eijgenstein, C. (2020). *Chrise96/3D_building_reconstruction*. https://github.com/chrise96/3D_building_reconstruction [Accessed: 10-8-2022].

Eijgenstein, C. (2021). Automatically enhance citygml lod2 buildings with facade details, by using a panoramic image sequence and building footprint data. Master's thesis, Vrije Universiteit Amsterdam.

Elgendy, M. (2020). *Deep learning for vision systems*. Simon and Schuster.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., et al. (1996). A density-based algorithm for discovering clusters in large spatial databases with noise. In *kdd*, volume 96, pages 226–231.

Fathalla, R. and Vogiatzis, G. (2017). A deep learning pipeline for semantic facade segmentation.

Fu, Y. and Yu, W. (2020). A formalization of properties of continuous functions on closed intervals. In *International Congress on Mathematical Software*, pages 272–280. Springer.

Gaw, L., Chen, S., Chow, Y., Lee, K., and Biljecki, F. (2022). Comparing street view imagery and aerial perspectives in the built environment. *ISPRS Annals of Photogrammetry, Remote Sensing & Spatial Information Sciences*, 10.

Gemeente Amsterdam (2007). *Kenmerk bouwlaag verblijfsobject*. https://www.amsterdam.nl/stelselpedia/bag-index/catalogus-bag/objectklasse-vbo/kenmerk-bouwlaag/ [Accessed: 3-1-2023].

Geospatial, T. (2022). Amsterdam boosts efficiency amp; access to spatial data through trimble-enabled mobile mapping.

Gilge, C. (2016). Google street view and the image as experience. *GeoHumanities*, 2(2):469–484.

Girshick, R. (2015). Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448.

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.

Goel, R., Garcia, L. M., Goodman, A., Johnson, R., Aldred, R., Murugesan, M., Brage, S., Bhalla, K., and Woodcock, J. (2018). Estimating city-level travel patterns using street imagery: A case study of using google street view in britain. *PloS one*, 13(5):e0196521.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep learning*. MIT press.

Google (2023). *Get Cloud credits for your students*. https://edu.google.com/programs/credits/teaching/?modal_active=none [Accessed: 10-1-2023].

Gramacki, A. and Gramacki, J. (2017). Fft-based fast computation of multivariate kernel density estimators with unconstrained bandwidth matrices. *Journal of Computational and Graphical Statistics*, 26(2):459–462.

Hartmann, A., Meinel, G., Hecht, R., and Behnisch, M. (2016). A workflow for automatic quantification of structure and dynamic of the german building stock using official spatial data. *ISPRS International Journal of Geo-Information*, 5(8):142.

He, K., Gkioxari, G., Dollár, P., and Girshick, R. (2017). Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969.

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.

Heidenreich, N.-B., Schindler, A., and Sperlich, S. (2010). Bandwidth selection methods for kernel density estimation-a review of performance. *Available at SSRN 1726428*.

Heidenreich, N.-B., Schindler, A., and Sperlich, S. (2013). Bandwidth selection for kernel density estimation: a review of fully automatic selectors. *AStA Advances in Statistical Analysis*, 97(4):403–433.

Hou, Y. and Biljecki, F. (2022). A comprehensive framework for evaluating the quality of street view imagery. *International Journal of Applied Earth Observation and Geoinformation*, 115:103094.

Hussain, S., Anwar, S. M., and Majid, M. (2018). Segmentation of glioma tumors in brain using deep convolutional neural network. *Neurocomputing*, 282:248–261.

Håbrekke, and Nordstad, F. D. (2022). Estimating the height of facades with street-level imagery using facade parsing, floor segmentation, and urban rules. Master's thesis, Norwegian University of Science and Technology.

Iannelli, G. C. and Dell'Acqua, F. (2017). Extensive exposure mapping in urban areas through deep analysis of street-level pictures for floor count determination. *Urban Science*, 1(2):16.

Jiang, H., Yan, D.-M., Dong, W., Wu, F., Nan, L., and Zhang, X. (2016). Symmetrization of facade layouts. *Graphical Models*, 85:11–21.

Kong, G. and Fan, H. (2020). Enhanced facade parsing for street-level images using convolutional neural networks. *IEEE Transactions on Geoscience and Remote Sensing*, 59(12):10519–10531.

Korc, F. and Förstner, W. (2009). etrims image database for interpreting images of man-made scenes. *Dept. of Photogrammetry, University of Bonn, Tech. Rep. TR-IGG-P-2009-01*.

Kramer, O. (2016). Scikit-learn. In *Machine learning for evolution strategies*, pages 45–53. Springer.

Kutzner, T., Chaturvedi, K., and Kolbe, T. H. (2020). Citygml 3.0: New functions open up new applications. *PFG–Journal of Photogrammetry, Remote Sensing and Geoinformation Science*, 88(1):43–61.

Laakso, M., Kiviniemi, A., et al. (2012). The ifc standard: A review of history, development, and standardization, information technology. *ITcon*, 17(9):134–161.

Läuter, H. (1988). Silverman, bw: Density estimation for statistics and data analysis. chapman & hall, london–new york 1986, 175 pp.,£ 12.—.

Ledoux, H., Arroyo Ohori, K., Peters, R., and Pronk, M. (2022). *Computational modelling of terrains*. Self-published.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., and Belongie, S. (2017). Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2117–2125.

Lin, T.-Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer.

Liu, C. (2011). Automatic facade image rectification and extraction using line segment features. In *VISAPP*, pages 104–111.

Liu, H. (2020). *DeepFacade*. `https://github.com/liuhantang/DeepFacade` [Accessed: 20-5-2022].

Liu, H., Xu, Y., Zhang, J., Zhu, J., Li, Y., and Hoi, S. C. (2020). Deepfacade: A deep learning approach to facade parsing with symmetric loss. *IEEE Transactions on Multimedia*, 22:3153–3165.

Liu, H., Zhang, J., Zhu, J., Hoi, S. C. H., Liu, H. ., Zhang, J. ., and Zhu, J. . (2017). Deepfacade: A deep learning approach to facade parsing.

Liu, X. and Milanova, M. (2018). Visual attention in deep learning: a review. *Int Rob Auto J*, 4(3):154–155.

Liu, Y., Hel-Or, H., Kaplan, C. S., Van Gool, L., et al. (2010). Computational symmetry in computer vision and computer graphics. *Foundations and Trends® in Computer Graphics and Vision*, 5(1–2):1–195.

Mahabir, R., Schuchard, R., Crooks, A., Croitoru, A., and Stefanidis, A. (2020). Crowdsourcing street view imagery: a comparison of mapillary and openstreetcam. *ISPRS International Journal of Geo-Information*, 9(6):341.

Maity, M., Banerjee, S., and Chaudhuri, S. S. (2021). Faster r-cnn and yolo based vehicle detection: A survey. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pages 1442–1447. IEEE.

Mathias, M., Martinović, A., and Van Gool, L. (2016). Atlas: A three-layered approach to facade parsing. *International Journal of Computer Vision*, 118(1):22–48.

McInnes, L., Healy, J., and Astels, S. (2017). hdbscan: Hierarchical density based clustering. *J. Open Source Softw.*, 2(11):205.

Meta AI Research (2020). *Benchmarks*. `https://detectron2.readthedocs.io/en/latest/notes/benchmarks.html` [Accessed: 21-6-2022].

Müller, P., Zeng, G., Wonka, P., and Van Gool, L. (2007). Image-based procedural modeling of facades. *ACM Trans. Graph.*, 26(3):85.

Nickolls, J., Buck, I., Garland, M., and Skadron, K. (2008). Scalable parallel programming with cuda: Is cuda the parallel programming model that application developers have been waiting for? *Queue*, 6(2):40–53.

Ning, H., Ye, X., Chen, Z., Liu, T., and Cao, T. (2022). Sidewalk extraction using aerial and street view images. *Environment and Planning B: Urban Analytics and City Science*, 49(1):7–22.

Nordmark, N. (2021). Window detection in facade imagery: A deep learning approach using mask r-cnn.

Odland, T. (2022). KDEpy. https://kdepy.readthedocs.io/en/latest/index.html.

Ogawa, Y., Oki, T., Chen, S., and Sekimoto, Y. (2021). Joining street-view images and building footprint gis data. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on Searching and Mining Large Collections of Geospatial Data*, GeoSearch'21, page 18–24, New York, NY, USA. Association for Computing Machinery.

OpenStreetMap Wiki (2022). *Key:building:levels — OpenStreetMap Wiki*. https://wiki.openstreetmap.org/w/index.php?title=Key:building:levels&oldid=2268226 [Accessed: 5-11-2022].

O'Shea, K. and Nash, R. (2015). An introduction to convolutional neural networks. *arXiv preprint arXiv:1511.08458*.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Pauly, M., Mitra, N. J., Wallner, J., Pottmann, H., and Guibas, L. J. (2008). Discovering structural regularity in 3d geometry. In *ACM SIGGRAPH 2008 papers*, pages 1–11.

Redmon, J. and Farhadi, A. (2018). Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

Ren, S., He, K., Girshick, R., and Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28.

Rosenfelder, M., Wussow, M., Gust, G., Cremades, R., and Neumann, D. (2021). Predicting residential electricity consumption using aerial and street view images. *Applied Energy*, 301:117407.

Roy, E. (2022). Inferring the number of floors of building footprints in the netherlands. Master's thesis, Delft University of Technology.

Roy, E., Pronk, M., Agugiaro, G., and Ledoux, H. (2022). Inferring the number of floors for residential buildings. *International Journal of Geographical Information Science*, pages 1–25.

Schmitz, M. and Mayer, H. (2016). A convolutional network for semantic facade segmentation and interpretation. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 41:709.

Sezen, G., Cakir, M., Atik, M., and Duran, Z. (2022). Deep learning-based door and window detection from building façade. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 43:315–320.

Sharma, R., Saqib, M., Lin, C., and Blumenstein, M. (2022). A survey on object instance segmentation. *SN Computer Science*, 3(6):1–23.

Sheather, S. J. and Jones, M. C. (1991). A reliable data-based bandwidth selection method for kernel density estimation. *Journal of the Royal Statistical Society: Series B (Methodological)*, 53(3):683–690.

Silverman, B. (2018). *Density Estimation for Statistics and Data Analysis*. Routledge.

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

Sun, Y., Malihi, S., Li, H., and Maboudi, M. (2022). Deepwindows: Windows instance segmentation through an improved mask r-cnn using spatial attention and relation modules. *ISPRS International Journal of Geo-Information*, 11(3):162.

Teboul, O., Kokkinos, I., Simon, L., Koutsourakis, P., and Paragios, N. (2011). Shape grammar parsing via reinforcement learning. In *CVPR 2011*, pages 2273–2280. IEEE.

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.

Tsironis, V., Tranou, A., Vythoulkas, A., Psalta, A., Petsa, E., and Karras, G. (2017). Automatic rectification of building façades. *The International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences*, 42:645.

Tyleček, R. and Šára, R. (2010). A weak structure model for regular pattern recognition applied to facade images. In *Asian Conference on Computer Vision*, pages 450–463. Springer.

VanderPlas, J. (2013). *Kernel Density Estimation in Python*. `https://jakevdp.github.io/blog/2013/12/01/kernel-density-estimation/` [Accessed: 10-1-2023].

Wand, M. (1994). Fast computation of multivariate kernel estimators. *Journal of Computational and Graphical Statistics*, 3(4):433–445.

Wang, L. (2022). Detailed facade reconstruction for mahattan-world buildings. Master's thesis, TU Delft Architecture and the Built Environment.

Wang, S., Kang, Q., She, R., Tay, W. P., Navarro, D. N., and Hartmannsgruber, A. (2022). Building facade parsing r-cnn. *arXiv preprint arXiv:2205.05912*.

Wu, C., Frahm, J.-M., and Pollefeys, M. (2010). Detecting large repetitive structures with salient boundaries. In *European conference on computer vision*, pages 142–155. Springer.

Wu, M., Zeng, W., and Fu, C.-W. (2021). Floorlevel-net: Recognizing floor-level lines with height-attention-guided multi-task learning. *IEEE Transactions on Image Processing*, 30:6686–6699.

Wu, Y., Kirillov, A., Massa, F., Lo, W.-Y., and Girshick, R. (2019). Detectron2. `https://github.com/facebookresearch/detectron2`.

Xie, S., Girshick, R., Dollár, P., Tu, Z., and He, K. (2017). Aggregated residual transformations for deep neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1492–1500.

Zeng, N. (2018). *An introduction to evaluation metrics for object detection*. https://blog.zenggyu.com/en/post/2018-12-16/an-introduction-to-evaluation-metrics-for-object-detection/ [Accessed: 10-11-2022].

Zhang, G., Pan, Y., and Zhang, L. (2022). Deep learning for detecting building façade elements from images considering prior knowledge. *Automation in Construction*, 133:104016.

Zhang, H., Xu, K., Jiang, W., Lin, J., Cohen-Or, D., and Chen, B. (2013). Layered analysis of irregular facades via symmetry maximization. *ACM Trans. Graph.*, 32(4):121–1.

Zhang, X., Lippoldt, F., Chen, K., Johan, H., Erdt, M., Zhang, X., Lippoldt, F., Chen, K., Johan, H., and Erdt, M. (2019). A data-driven approach for adding facade details to textured lod2 citygml models. In *VISIGRAPP (1: GRAPP)*, pages 294–301.

Zuliani, M., Kenney, C. S., and Manjunath, B. (2005). The multiransac algorithm and its application to detect planar homographies. In *IEEE International Conference on Image Processing 2005*, volume 3, pages III–153. IEEE.

Colophon

This document was typeset using LaTeX, using the KOMA-Script class `scrbook`. The main font is Palatino.

TU Delft