MSc thesis in Geomatics

Automatic reconstruction of 3D city models from historical maps

Camille Morlighem

2021

TU Delft

MSc thesis in Geomatics

Automatic reconstruction of 3D city models from historical maps

Camille Morlighem

June 2021

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

The work in this thesis was carried out in the:

3D geoinformation group
Delft University of Technology

Supervisors:	Hugo Ledoux
	Anna Labetski
Co-reader:	Francesca Noardo

Abstract

Historical 3D city models have been increasingly used for the preservation and communication of the cultural heritage to a wider and more diversified public. In the recent years, they have also been of a growing interest in other domains such as in urbanism or in economy. However, their potential for supporting new use cases has been restricted by the difficulty to generate these models. Historical 3D city models can only be reconstructed from historical sources, such as historical maps, and this means dealing with all sorts of constraints and inaccuracies. As a result, reconstructing historical 3D city models is a challenging process that is to date still essentially manual and time-consuming. This thesis investigates to what extent the reconstruction of historical 3D city models can be automated. Several existing methods for extracting building footprints from historical maps have been tested and compared so as to identify the pros, cons and use cases of each method and all the challenges of working with historical maps. Based on these experiments a fully automated methodology was developed. It relies on three main stages: (1) the processing of the historical maps to extract the building plots, (2) the subdivision of these building plots into individual building footprints and (3) the reconstruction of a LoD2 historical 3D city model using 3D procedural modelling. This methodology was implemented with historical maps from two different study areas, Delft and Brussels, and for different epochs in order to reconstruct a dynamic historical 3D city model for these cities. The results show that the methodology workflow developed in this thesis allows to reconstruct automatically historical 3D city models for different historical maps collections and for different study areas. The main differences between the two case studies, Delft and Brussels, regard the implementation details (i.e. data availability, running time and user-defined parameters) but similar results are obtained, which show the suitability of the methodology to be applied for other study areas. Two elements are identified as main factors influencing the quality of the results obtained: the quality of the scanning process and the symbology of the historical maps. For historical maps that were properly scanned, with sufficient spatial resolution and strict symbology rules, the methodology provides accurate results by identifying more than 84% of the building plots in the ground truth and classifying properly more than 89% of the building plots. In addition, all historical 3D city models reconstructed have their geometries valid at more than 99%. Overall, this thesis provides a methodology for reconstructing automatically historical 3D city models from historical maps along with guidance and hints about this process and about a series of other methods, so that any user can find the most suitable method for their needs.

Acknowledgements

I would like to express my gratitude to my supervisors, Dr. Hugo Ledoux and Anna Labetski, for their guidance, constant advice and support during this graduation project. I also thank them for their almost unlimited availability and the numerous meetings we had and that kept me very motivated during my thesis. I really enjoyed doing this graduation project with them as supervisors. In addition, I would like to thank my co-reader, Francesca Noardo, for the in-depth proof-reading of my thesis and her constructive suggestions to improve it. Special thanks also go to Jules Schoonman for the historical maps he provided and that have been very valuable to conduct this thesis. Finally, I would like to thank Lei Qu for her useful questions and comments.

I would also like to thank my parents and my sister for their continuous support during my entire studies and encouragements to never give up when faced with a new challenge.

Contents

List of Figures

List of Tables

Acronyms

1:n	one-to-many	
2PM	2D procedural modelling	30
AHN3	Actueel Hoogtebestand Nederland 3	48
BAG	Basisregistratie Adressen en Gebouwen	39
BCGA	Computer Generated Architecture for Blender	33
BnF	Bibliothèque nationale de France	47
CAD	computer-aided design	10
CFD	computational fluid dynamics	86
CGA	Computer Generated Architecture	33
CGAL	Computational Geometry Algorithms Library	
CRS	coordinate reference system	14
DEM	Digital Elevation Model	11
DEMs	Digital Elevation Models	
DP	Douglas-Peucker	9
DT	Delaunay triangulation	11
GDAL	Geospatial Data Abstraction Library	18
GIS	geographic information system	9
GISs	geographic information systems	
LiDAR	light detection and ranging	
LoD	Level of Detail	
LULC	land use/land cover	16
MBR	minimum bounding rectangle	23
NYPL	New York Public Library	9
OBIA	Object-based Image Analysis	14
RMSE	Root Mean Square Error	14
TU Delft	Technische Universiteit Delft	

1 Introduction

Historical maps represent an invaluable source of information in various domains such as urban studies, social sciences or economy, as they contain information that cannot be found elsewhere [Balletti and Guerra, 2016; Gobbi et al., 2019]. Often they are the only information available about the pre-satellites and digital maps era [Liu et al., 2019]. One important feature of historical maps is that they can help understand the current configuration of a landscape or of a city as this configuration relies on the past changes that took place [Nobajas and Nadal, 2015]. In addition, historical maps also find application in the health domain. For instance, Figure 1.1 represents the famous map of John Snow, showing cholera deaths during the cholera epidemic in 1854 in London [McLeod, 2000]. Back in the day, this map was an important first step in disease mapping [McLeod, 2000]. Nowadays, with the advance of technology, geomatics tools have the potential for making this kind of maps even more valuable, allowing to process them so that they can be used for present-day applications [Balletti and Guerra, 2016]. For instance, geomatics techniques could be used for raising John Snow's map to the third dimension, which could provide information about air circulation and thus give additional insights into how the disease spread in the city. This example demonstrates an application of historical 3D city models.

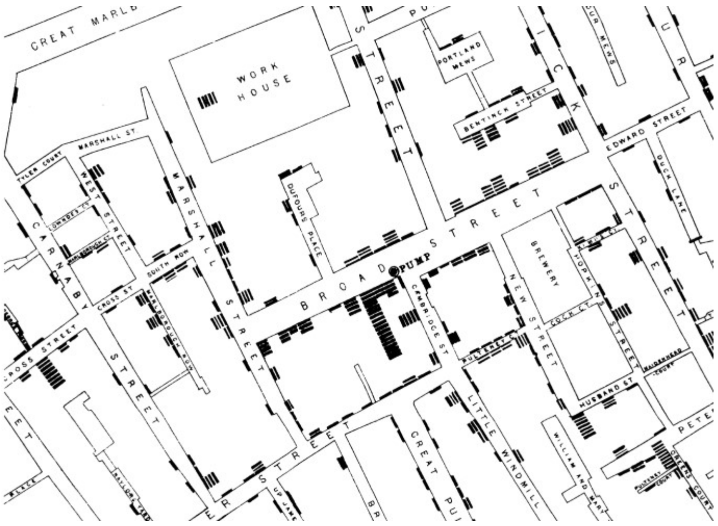


Figure 1.1: Cholera deaths during the cholera pandemic in London, 1854. The thick black dashes represent cholera-related deaths. Figure from McLeod [2000].

In the recent years, historical 3D city models have arisen as a new way for preserving historical heritage. Some studies have focused on reconstructing impressive, highly detailed, historical 3D city models [Frischer et al., 2008]. For example, the Municipality of Rotterdam has reconstructed a highly detailed historical 3D city model of Rotterdam in 1940[1] (Figure 1.2). A major downside of these models is that they can only be reconstructed manually and therefore they require lots of time, labour and money. For the historical 3D city model of Rotterdam in 1940, it required ten people working full-time and height months to produce it. Besides, the availability of historical data can make their reconstruction even more complex; it is rare and difficult to obtain a complete set of historical data covering the whole study area [de Boer, 2010; Isoda et al., 2009]. While the difficulty of producing such models limits their reconstruction, less detailed historical 3D city models can also find interesting application. They provide a way to explore historical sites in space but also in time and for a worldwide public if they are made available

[1]Available at `https://stadsarchief.rotterdam.nl/over-ons/projecten/rotterdam-3d/index.xml`.

over the Internet [Balletti and Guerra, 2016; Guidi and Russo, 2011; Maiwald et al., 2019]. Furthermore, they can also be used for more pragmatic applications such as the estimation of the population size of a city at times where census or other population data were not available [Biljecki et al., 2016a]. Figure 1.3 shows an example of simple historical 3D city model which represents Cruquius in the 17th century.

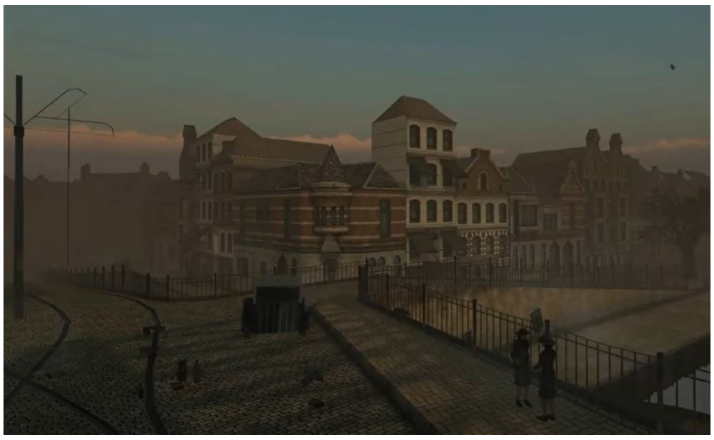


Figure 1.2: Historical 3D city model of Rotterdam in 1940

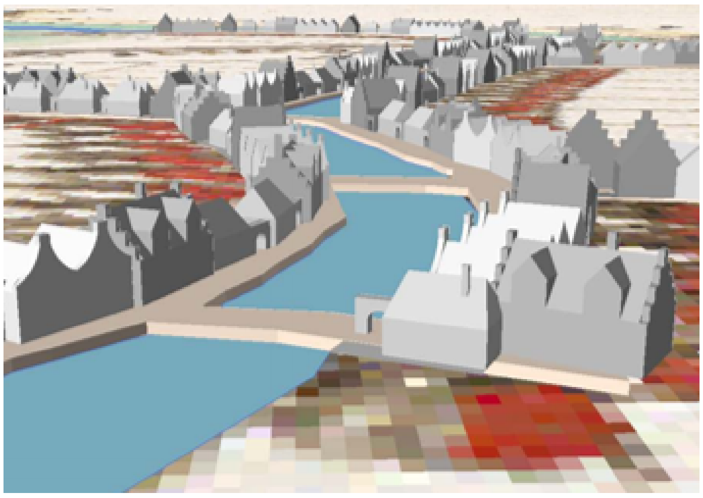


Figure 1.3: Simple historical 3D city model of Cruquius in the 17th century. Figure from de Boer [2010].

Although simpler historical 3D city models are easier to reconstruct, they cannot be generated with the same modern techniques as current 3D city models. Thus, they rely on extracting information from historical maps. Data imperfection in historical maps makes this task particularly challenging [Kersten et al., 2012]. Historical maps are usually affected by geometric and chronometric inaccuracies because of the way they were created back in the day, using triangulation networks. Those maps were often influenced by the cartographers' subjectivity [Laycock et al., 2011; de Boer, 2010; Heitzler and Hurni, 2020]. Therefore, 3D modelling of historical cities is a challenging process, that is time-consuming and manual, and to date it has not been possible to fully automate it [Kersten et al., 2012; Riche, 2020]. Automating the reconstruction of historical 3D city models can spare historians and other domains researchers the tedious task of manually producing them. Moreover, it could make it possible to easily create dynamic 3D city models showing the evolution of a city over a certain time period.

For these reasons, the goal of this research is to explore whether and how the reconstruction of simple historical 3D city models can be automated. This study focuses on two main axis: the processing of the historical maps to derive vector data and the 3D reconstruction of the historical cities from these vector data. For the first axis, different existing methods in the literature are implemented and compared. As for the second axis, it is based on procedural modelling, a process that allows reconstructing automatically large 3D city models from vector data. This study focuses on the case study of Delft, with the aim of automatically reconstructing historical 3D city models of this city for different time periods between 1880 and 2000. In addition, Brussels is also used as a case study for testing the methodology. Figure 1.4 shows an example of historical map that is used in this study. It represents the city of Delft in 1880. All data, source codes and the output models reconstructed in this thesis are freely available at https://github.com/CamilleMorlighem/histo3d.

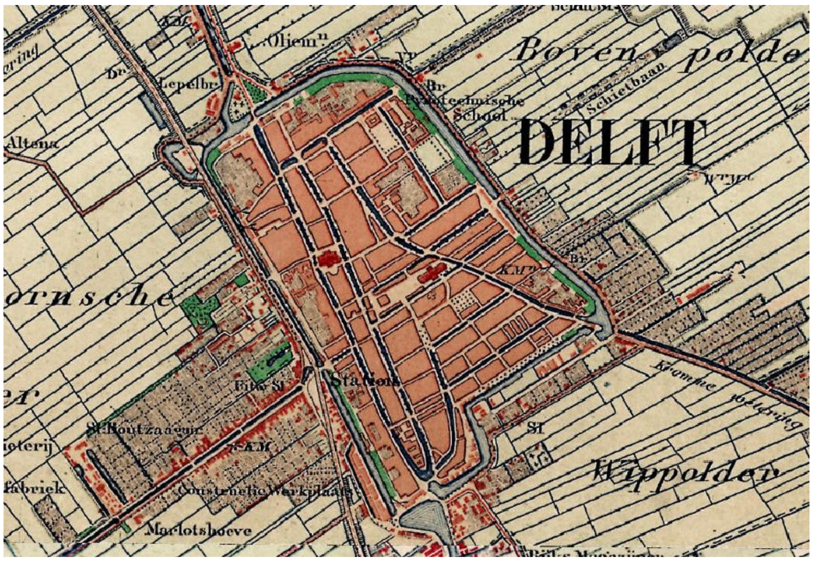


Figure 1.4: Historical map of Delft in 1880. Available from https://www.topotijdreis.nl/.

1.1 Objectives and research questions

The main objective of this research is to develop a methodology for automatically reconstructing historical 3D city models from historical maps and avoid as much as possible using manual processing. The focus of this study is thus the automation of the whole process from the processing of the input historical maps to the generation of the 3D buildings. Developing that methodology also requires as sub-objective the census, assessment and comparison of existing methodologies so as to provide information about which methodology is suitable for which application. Eventually, the resulting methodology will be implemented with historical maps of different epochs in order to reconstruct a dynamic 3D city model.

Special care must be taken to define the type of historical 3D city models this study aims to produce, as it will lead the development of the methodology. The output historical 3D city models should be made of LoD2 buildings and stored in CityJSON format. Historical 3D city models are usually reconstructed with two different approaches: the first one aims to achieve an historical 3D city model as much realistic as possible, while the second one aims to obtain an historical 3D city model as much accurate as possible [de Boer, 2010; Nijhuis et al., 2011]. The main difference between these two approaches is that they support different applications. Realistic 3D city models are prone to lots of approximations as they mainly aim at visualisation purposes while accurate 3D city models support more pragmatic applications and use cases, where geometric accuracy is required. In this study, the aim is to reconstruct the historical 3D city models the most accurate as possible. However, as we are limited by the amount of historical information available, educated guesses and assumptions must be made.

To answer those objectives, the whole study is guided by the following research question:

"To what extent can be automated the process of reconstructing simple 3D city models from historical maps?"

To answer this main research question, additional sub-questions are also relevant to be answered. They are as follow:

- What are the difficulties in the process of automatically reconstructing 3D city models from historical maps?
- What results do we achieve with existing methods for extracting building footprints from historical maps? What are the pros and cons of these methods?
- Which degree of automation can be achieved in the process of reconstructing 3D city models from historical maps? Are some manual interventions necessary?
- How much accurate are the historical 3D city models reconstructed?
- In what ways and for which applications are the simple historical 3D city models obtained relevant in comparison to highly detailed manually derived historical 3D city models?

1.2 Scope

The following clarifies the scope of this thesis:

- This study focuses on the reconstruction of historical 3D city models containing 3D buildings but the reconstruction of other features such as roads, canals or landscape features will not be investigated. Thus, the different methods implemented are compared and assessed on how well they work for reconstructing building features and their performance for other types of features is not taken in consideration. Different feature categories on historical maps actually require different specific processing methods to be automatically extracted [Herrault et al., 2013b].
- In this research, the only historical sources used for reconstructing historical 3D city models are historical maps. Old photographs, paintings or other types of historical sources are thus not used. Some studies made use of old photographs for historical 3D modelling for adding textures [Younes et al., 2013; Maiwald et al., 2019]. Since the aim here is to reconstruct simple historical 3D city models, texturing is out of scope of this thesis.
- The historical maps used in this study are already scanned and thus digitally available. This step is thus not considered in this study. Besides, the georeferencing of the historical maps is not part of the automation process. The historical maps used are either already georeferenced or georeferenced manually. Some research has focused on automating the georeferencing step [Sun et al., 2020], but even manually this step is not always straightforward and prone to errors [Nobajas and Nadal, 2015]. As the results of the methodology implemented in this study strongly rely on the georeferencing step, the latter was made manually in order to derive georeferenced maps as much accurate as possible.
- The aim of this thesis is not to reinvent the wheel. Thus, this study starts from existing methods, assesses and combines them before creating new ones.

1.3 Thesis outline

The present thesis is structured as follows:

Chapter 2 gives an overview of the scientific research related to this thesis and provides conclusions about the literature review conducted.

Chapter 3 provides the whole methodology workflow developed to answer the research objectives and theoretical background about the different processes used or implemented.

Chapter 4 presents the datasets used to implement and test the methodology and the programming softwares and tools on which it relies.

Chapter 5 describes and analyses the results obtained with the methodology workflow.

Chapter 6 discusses the results obtained before providing the conclusions of this thesis and the future work to which it opened a window.

2 Related Work

In this chapter, an overview of the related work is provided. First, research and methods used for the digitalisation of historical maps are described (Section 2.1). Then, existing methods for extracting building footprints from historical maps (Section 2.2) and reconstructing historical 3D city models are addressed (Section 2.3). Lastly, this chapter concludes on the observations and findings made from this literature review (Section 2.4).

2.1 Digitalisation of historical maps

Map digitalisation is the process of recreating a vector or a raster map representing the geographic features depicted on an input paper map [Gobbi et al., 2019]. Map digitalisation usually comprises three main steps. The first step deals with scanning the paper map to make it digitally available. Once the map is scanned, the following step is to georeference it, which means placing the map geographic features within a geographic datum. Finally, the geographic features can be classified into different categories to produce a digitised map [Gobbi et al., 2019].

Lots of studies have focused on applying the digitalisation process to historical maps. Color is usually one of the most important features used to distinguish geographic features from an historical map as color-based symbology was already used in early cartography [Liu et al., 2019]. Three main categories of methods can be identified: histogram thresholding, machine learning classification and deep learning classification.

2.1.1 Histogram thresholding

Histogram thresholding deals with analysing the color histogram of the map to detect the peaks in every color layer. Based on those peaks, thresholds can be defined such that all pixels comprised between specific thresholds represent the same category of geographic features [Liu et al., 2019]. Sometimes, some additional processing as histogram stretching or transforming the RGB image into another color space is used to increase the difference between the different color layers and make the peaks more visible [Liu et al., 2019]. For instance, Pezeshk and Tutwiler [2008] applied some morphological operators to increase the difference between the color layers and then used histogram thresholding for classifying scanned topographic maps. Similarly, Ebi et al. [1994] used histogram thresholding in the u'v' color space. One downside of these methods is that they only use color information as unique feature to distinguish the different categories of geographic features and do not consider spatial autocorrelation [Gobbi et al., 2019; Liu et al., 2019]. As a result they are sensitive to effects introduced by the scanning process such as aliasing and false color. Aliasing is due to the fact that scanning a map implies sampling and pixels located at the boundary between two different geographic features will thus be mixed. This phenomenon common to current topographic maps is even reinforced on historical maps where bleaching sometimes leads to high color variations in the map. As for the false color effect, it happens when the three RGB planes are not perfectly aligned during the scanning process. This leads to color variation inside the same geographic feature [Liu et al., 2019]. Because of these two phenomena, histogram thresholding produces good results for good quality historical maps but does not prove sufficient for poor quality ones [Liu et al., 2019].

2.1.2 Machine learning classification

Classifying a map in machine learning refers to the process of categorising the features on the map into different classes, using training data. From the training data, the spectral signature of the different geographic features can be derived and it is then used to perform the classification. Two main approaches can be used: either the classification is made at the pixel level or it is made using patches of pixels, which are usually called segments or "objects" [Gobbi et al., 2019]. An example of pixelwise classification (made at the pixel level) is for instance the work of Henderson et al. [2009]. They compared unsupervised classification algorithms, such as the expectation maximization algorithm and the K-means clustering algorithm, to automatically classify raster maps. In the same way, Herrault et al. [2013b] made use of a K-means clustering algorithm for classifying historical maps. One downside of pixelwise classifications is that, as histogram thresholding, they do not cope for the aliasing and false color effects as they are only based on color information and do not consider spatial contiguity. Contrarily, segmentation (classification made at the segment level) allows taking into consideration color but also shape properties for forming and classifying the different segments. For instance, Gobbi et al. [2019] made use of a region-growing algorithm to segment historical maps and then used training data to classify the segments based on shape and color properties.

2.1.3 Deep learning classification

Deep learning for map classification comprises a series of machine learning techniques in which neural network models are trained to classify the geographic features of a map [LeCun et al., 2015]. Except for written-text recognition, the use of deep learning for classifying historical maps has been until now limited, as shown by the review papers of Ares Oliveira et al. [2017] and Ignjatić et al. [2018]. The main reason behind this is that the training dataset needed is usually expensive to produce in terms of time and money, while the reuse possibilities are limited. Indeed, historical maps usually have their own map symbology, colors, hues and contrasts [Sun et al., 2020], such that it is unlikely that using the same training data produces good results for two different historical maps [Ignjatić et al., 2018]. The results of Ares Oliveira et al. [2017] show indeed a reduction in performance from 99% to 10% when using the same pre-trained model for two different historical maps. Besides, even for the same historical map, the same deep learning model could perform differently for two different map sheets if the map is affected by bleaching or was damaged with time [Heitzler and Hurni, 2020]. The lack of sufficient training data available can also be a limitation for using deep learning [Ignjatić et al., 2018]. For instance, in the work of Liu et al. [2017], they used a full training dataset made of 100,000 unannotated test images and 870 human-annotated images for classifying historical cadastral plans. Creating such a big dataset for classifying an historical map is not only time-consuming but it might also simply be impossible depending on the size of the original historical map [Ignjatić et al., 2018].

2.2 Existing methods for building footprints extraction

Different types of geographic features can be extracted from historical maps. For instance, one may want to extract roads, contour lines, textual features or building footprints. To be automatically extracted, different feature categories actually require different processing techniques [Herrault et al., 2013b]. In this thesis, as the aim is to reconstruct historical 3D city models made of buildings, we focus solely on building footprints extraction techniques. Lots of research have been done on developing automatic extraction methods of building footprints from aerial imagery. However, these methods cannot be applied to historical maps as the latter are usually of lower quality and come with different constraints [Sun et al., 2020]. After a literature review of existing methodologies specific to historical maps, these can be classified into four main categories: smart crowdsourcing, color-based image processing, machine and deep learning and edge detection.

2.2.1 Smart crowdsourcing

Some crowdsourcing projects are sometimes introduced for extracting building footprints from historical maps. For instance, the New York Public Library (NYPL) has developed a crowdsourcing project for extracting building footprints from atlases of the 19th and the 20th centuries. Staff and volunteers manually digitised the building polygons on the historical maps from a crowdsourcing website[1]. With this process, it required three years for digitising 170,000 building footprints from four New York atlases [Budig et al., 2016; Arteaga, 2013]. Recently the NYPL has started automating the process with image processing tools for extracting the polygons such that staff and volunteers are only needed for validating and correcting the extracted polygons. A challenge in this new process is that the same building footprint is usually corrected by many people and there is then a need for finding an averaged polygon. Budig et al. [2016] introduced a method for deriving a consensus polygon from all digitised polygons representing the same building. Their methodology relies on a vertex voting algorithm. After filtering the outlier polygons, the algorithm starts by clustering the vertices of the remaining polygons and identifies the path through the different clusters that is used by most polygons. Crowdsourcing projects making use of such solutions can be qualified as smart crowdsourcing [Budig et al., 2016].

2.2.2 Color-based image processing

Other building footprints extraction methods rely on digital image processing techniques using color space information. For instance, Drolias and Tziokas [2020] proposed an automatic method in a geographic information system (GIS), which starts from manual histogram thresholding to create a binary raster from the input historical map. After removing isolated pixels, the patches of pixels representing building footprints are vectorised. Eventually, the building footprints are smoothed using a Douglas-Peucker (DP) algorithm. Similarly, Arteaga [2013] developed a methodology relying on open source tools for extracting building footprints from historical cadastral plans. Their methodology relies on a calibration step which consists in enumerating all the colors present in the input map and the class they represent. By playing with the brightness and contrasts of the original image, a black and white image is created and patches of pixels are vectorised and smoothed using an α-shape operator. Lastly, each resulting polygon is cropped with the input historical map and it is kept if the average color of the cropped image matches with the building color.

2.2.3 Machine and deep learning

Machine and deep learning methods have also been implemented for automatically extracting building footprints from historical maps. Oliveira et al. [2017] extracted automatically building footprints from historical cadastral maps using a graph-based segmentation approach. Superpixels are created as patches of pixels sharing color properties. The superpixels are then used as the vertices of a graph where edges are weighted based on the similarity between neighbouring superpixels. By removing edges representing high dissimilarity, homogeneous regions are created and they are then classified into textual features, background and contour using a supervised machine learning algorithm. The final building footprints are obtained using a flood fill algorithm filling background-labelled regions until the contour on their boundary is met. Another methodology based on deep learning was implemented by Heitzler and Hurni [2020]. Their methodology makes use of a neural network architecture to classify an input historical map and produce a binary map. The binary map is then vectorised using a corner point detection algorithm and perpendicularity is enforced in the output building footprints.

2.2.4 Edge detection

A few building footprints extraction methods rely solely on edge detection algorithms. One downside of this category of methods is that they can usually only be applied with monochromatic historical maps

[1]Available at `http://buildinginspector.nypl.org/`.

[Gobbi et al., 2019; Riche, 2020]. For instance, Riche [2020] uses an edge detection algorithm called Otsu's binarization to extract the contour of the building footprints from historical maps. Their methodology is not fully automated as a manual processing step is taken afterwards for cleaning up and correcting the extracted building footprints. Laycock et al. [2011] also developed a methodology based on edge detection but it makes it possible to extract building footprints from multicolored historical maps. A seed point is first manually added in all building footprints of the map to serve as starting point for a flood fill algorithm. After implementing the flood fill algorithm, the resulting building footprints are vectorised. Eventually, edge detection is performed on the original historical map and the vectorised polygons are adjusted by forcing their contour to lie on the building edge pixels of the historical map.

2.3 Existing methods for historical 3D city models reconstruction

Modern measuring techniques such as laser scanning or photogrammetry used for reconstructing current 3D city models cannot be applied for historical 3D city modelling as they only allow to represent the current city configuration [Laycock et al., 2011]. Therefore, the 3D reconstruction of historical cities relies mainly on historical documents such as paintings, drawings, maps or photographs [Kersten et al., 2012]. The initial attempts for modelling in 3D past landscapes or cities date back to the 1980s but 3D historical modelling was truly made possible from the 1990s with the evolution of 3D GIS and computer-aided design (CAD) softwares [de Boer, 2010]. For instance, the Rome Reborn Project, which started in 1997, aims at reconstructing historical 3D city models showing the evolution of Rome from the Bronze Age to the Middle Ages (Figure 2.1) [Frischer et al., 2008].

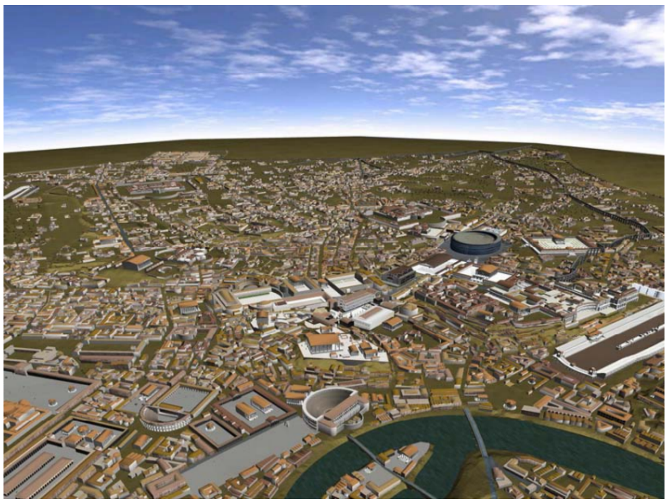


Figure 2.1: Historical 3D city model from the Rome Reborn Project. Figure from Guidi et al. [2012].

In the recent years, there has been more research on developing methodologies to reconstruct historical 3D city models from historical maps in combination with other historical sources. Nobajas and Nadal [2015] developed a method which consists in manually digitising historical cadastral maps into different land cover layers. The historical 3D city model was then reconstructed from the building footprints layer in a desktop GIS using a generic house model for all buildings. Another solution was proposed by Kersten et al. [2012] for reconstructing an historical 4D city model. Their methodology relies on recreating a first historical 3D city model by manually digitising historical cadastral plans and further extruding the building footprints. Building height was derived from historical photographs. A second historical 3D city model was created from the recording of an historical wooden model for deriving a point cloud and further processing it to reconstruct 3D buildings. Eventually, from those two historical 3D city models and historical maps of past epochs, they derived historical 3D city models for other epochs. The complete reconstruction of the final historical 4D city model required 800 hours in total, with half of the time spent in manual processing steps. Balletti and Guerra [2016] also worked on a methodology for reconstructing a dynamic historical 3D city model from a collection of historical maps, drawings and other historical documents. Their methodology is based on the combination of database archiving for storing and managing the different historical sources and GISs for the 2D and 3D modelling of the city from these historical documents. After reconstructing a 3D city model for the current city situation, they modified each building based on the historical data available to recreate 3D city models at past epochs.

Although these methods produce very good results, they rely strongly on manual processing steps. Some attempts have been made for automating the process. In that way, de Boer [2010] proposed a GIS-based method for automating the reconstruction of historical 3D city models. Their methodology relies on the selection of all pixels of the building color on the historical maps. All building pixels are then vectorised into point features and a Delaunay triangulation (DT) is built. Triangles with a large area are removed and adjacent polygons are merged into one connected area to create the different building footprints. Using Google Sketchup, they recreated a generic house model from historical drawings and paintings and they overlaid it over each recovered building footprint. One interesting feature of their methodology is that, unlike usual studies, they reconstructed an historical Digital Elevation Model (DEM) and used it as ground building height in the historical 3D city model. To do so, they compared historical and current land use maps to identify the unchanged areas. By extracting current elevation points in these unchanged areas and interpolating them, they were able to reconstruct an historical DEM. Other research has been carried out for automating the reconstruction of historical 3D city models of the Edo era in Japan. In that way, Isoda et al. [2009] used historical paintings and remaining historic houses of the Edo period to create 3D parametric models of the different types of houses. Current cadastral plans were overlaid onto an historical street map to recreate historical building lots. Eventually, an historical 3D city model was built by substituting the coordinates of the 2D lots into the 3D parametric house models. Suzuki and Chikatsu [2003] also focused on the Edo period using a similar approach. Instead of using current cadastral plans, they recovered building footprints from historical maps using a corner point detection algorithm. However, their approach only makes it possible to detect rectangular building footprints.

2.4 Observations from all methods

Some conclusions can be drawn from the literature review and all the methods described in this chapter. In all the methodologies developed for reconstructing historical 3D city models, the main challenge regards the digitalisation of the historical maps and the extraction of the building footprints, which essentially stay manual. Some research has tried to automate the building footprints extraction step using very different approaches, from image processing techniques to GIS-based methods. However, most of the existing methods still rely on some manual processing or seem to be specific to certain types of historical maps, as it was also noticed in the review paper of Liu et al. [2019]. In that way, the input historical maps used by the different studies often drive the main development of the methodology. For instance, some methods require as input monochromatic historical maps [Riche, 2020] or historical maps with high color contrasts [Herrault et al., 2013b]. Some research suggests that it is not possible to find a unique methodology working for any collection of historical maps, as they all have their own map symbology, colors, hues and contrasts [Sun et al., 2020]. Different map collections would thus require different methodologies [Sun et al., 2020; Liu et al., 2019; Herrault et al., 2013b]. Therefore, there is a need for investigating and comparing these different methodologies to determine which one is effective for which type of historical maps.

Furthermore, the use of training datasets is sometimes considered as an obstacle to automation, especially in deep learning-based methodologies [Ignjatić et al., 2018]. However, using training data might be a compromise between automating the methodology and making a methodology working for any collection of historical maps. As a consequence, it would be relevant to investigate how well methodologies based on training data perform for different collections of historical maps.

3 Methodology

This chapter focuses on the methodology developed to automatically reconstruct 3D city models from historical maps. Figure 3.1 displays the general methodology workflow. It relies on three main stages: (1) the processing of the historical maps to extract the building plots, (2) the subdivision of these building plots into individual building footprints and (3) the reconstruction of LoD2 buildings from these building footprints. There are two versions of the methodology: a complete and a shortened version. The complete version comprises all the steps given in Figure 3.1 including the optional ones, while the shortened version skips these optional steps. Choosing one version or the other actually depends on the data availability about the study area as the optional steps rely on datasets with specific attributes (Section 4.1.2). In the following sections, the complete version is described and more details about the skipping of the optional steps are provided in their own sections. This chapter starts by presenting the data preparation steps (Section 3.1). Then, the building plots extraction method is described along with the assessment conducted to evaluate it (Section 3.2). Next, the processes used to reconstruct individual building footprints from the historical building plots are presented (Section 3.3). The last section focuses on the steps to take for reconstructing LoD2 buildings from the historical building footprints (Section 3.4).

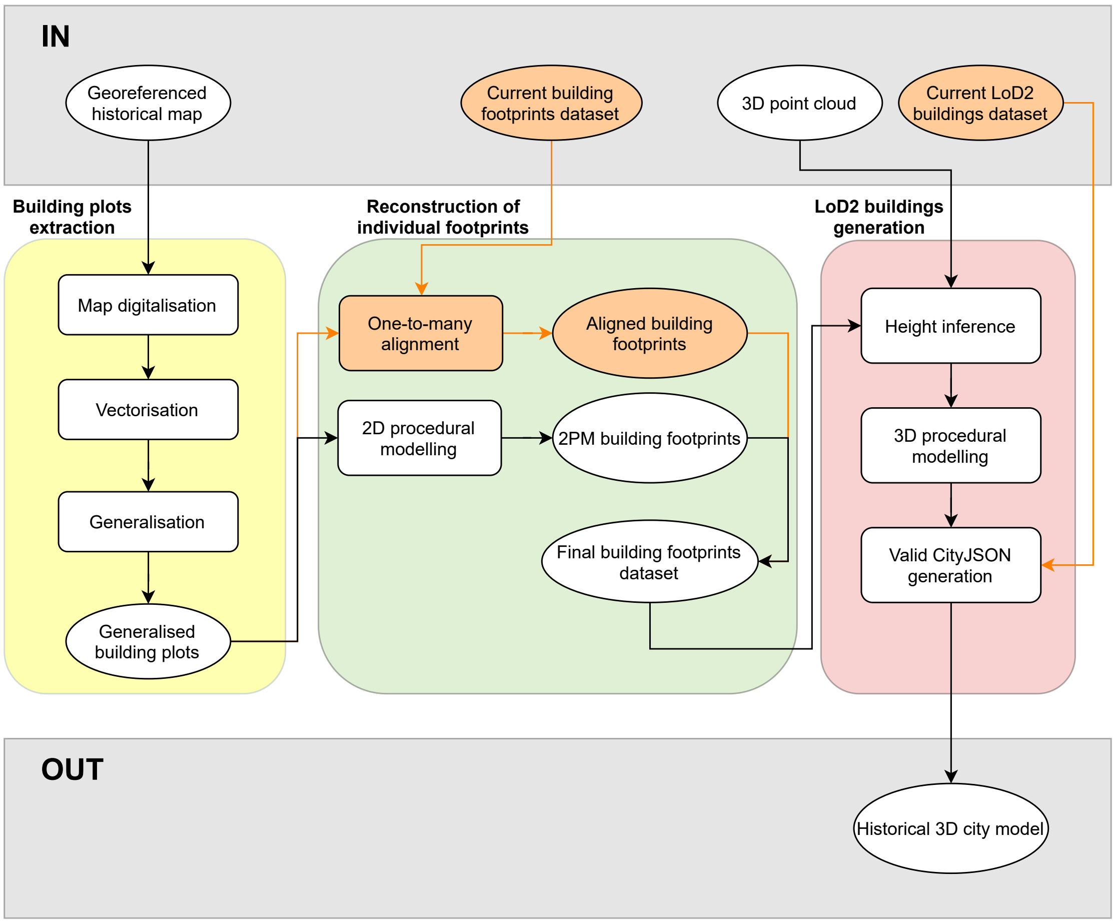

IN
Georeferenced historical map
Current building footprints dataset
3D point cloud
Current LoD2 buildings dataset
Building plots extraction
Reconstruction of individual footprints
LoD2 buildings generation
Map digitalisation
Vectorisation
Generalisation
Generalised building plots
One-to-many alignment
Aligned building footprints
2D procedural modelling
2PM building footprints
Final building footprints dataset
Height inference
3D procedural modelling
Valid CityJSON generation
OUT
Historical 3D city model

Figure 3.1: General methodology workflow. Rectangular boxes represent methodology steps and oval boxes represent datasets. Optional datasets and steps are depicted in orange.

3.1 Data preparation

3.1.1 Map projections

The methodology developed requires the input historical map to be georeferenced into a projected coordinate reference system (CRS) as different steps of the methodology rely on computing distance and other measures in meters. If the input historical map is not available that way, there are two possible operations to implement:

- **Map reprojection:** If the input historical map is not available into a projected CRS but is instead georeferenced in a geographic CRS, it must be reprojected. A transformation matrix is applied to change the coordinates of all the map geographic features from the geographic CRS to a projected one.

- **Georeferencing:** If the input historical map is not already available as georeferenced, it is manually georeferenced in a GIS. Pairs of control points are created on distinctive locations (e.g. road intersections or field corners) on the historical map and on a georeferenced map of the same study area. Different types of transformations can be applied for finding the coordinates of all geographic features in the historical map. Three types of transformations are usually used: local transformations, global transformations and global transformations locally sensitive (based on kernels) [Follin et al., 2016; Herrault et al., 2013a]. With these transformations, residuals are created and the best transformation is the one that minimizes them. The Root Mean Square Error (RMSE) can help assess the quality of the georeferencing. This process is not always straightforward, even more with historical maps as they are affected by geometric and chronometric inaccuracies. Depending on the input historical map, this can greatly influence the quality of the georeferencing [Nobajas and Nadal, 2015]. Thus, in this study, different transformations are tested and compared in order to obtain the georeferenced historical maps the most accurate as possible, as the other steps of the methodology heavily rely on it.

3.2 Building plots extraction

The building plots extraction step consists in extracting and vectorising the building plots from the input historical map. This part of the workflow relies on different steps: (i) digitalising the input historical map to generate a binary or classified raster map, (ii) vectorising and generalising the rasterised building plots and (iii) computing different quality metrics to assess the results. In this thesis, I have used, implemented and compared different building footprints extraction methods described in Chapter 2 to determine the most suitable one for performing this task. I report in Section 5.2.1 on the results of these comparisons. The following subsections detail the combination of methods which provided the best results.

3.2.1 Map digitalisation

The map digitalisation step deals with the processing of the input historical map to create a classified raster map. Different methods can be followed for this purpose, using either a pixelwise or an object-oriented approach. After different experiments (Section 5.2.1), it was chosen to use the object-oriented method of Gobbi et al. [2019]. They designed a methodology in three main steps: segmentation, Object-based Image Analysis (OBIA) classification and text removal.

Segmentation

First, a segmentation is applied on the input historical map. In that way, the map is segmented such that patches of pixels sharing color and shape properties are formed. The segmentation is based on a region-growing algorithm which requires two input parameters for managing the creation of the segments. The first parameter is a threshold value comprised between 0 and 1 and that represents the maximum dissimilarity between neighbouring pixels belonging to the same segment. The higher the value, the more the radiometric variations between neighbouring pixels are considered. Thus, for a threshold value of 0, all pixels are considered different and the segmented map is exactly the same as the input map, while for a value of 1, only one segment is created containing all pixels of the map. The second parameter is the minimum number of pixels included in a single segment. Low values for that parameter lead to a high number of segments and smaller objects are better identified. But if the value is too high, big objects are split into many small segments and this could cause an issue for later regrouping them.

Hyperparameters tuning can help define the best combination of parameters to perform the segmentation. The algorithm for tuning the parameters is called unsupervised segment parameter optimization [Espindola et al., 2006]. It measures for different random combinations of parameters the intra-segment homogeneity, looking at the segment variance weighted by the segment size (VW), and the inter-segment heterogeneity using a spatial autocorrelation index (SA). If VW is high, it means that the input map was undersegmented while a high value of SA translates an oversegmentation. The best segmentation is the one maximising both the intra-segment homogeneity and the inter-segment heterogeneity, which means lowering both VW and SA [Espindola et al., 2006]. To measure the performance of each segmentation, the algorithm normalises VW and SA by substituting them to X in Equation 3.1.

$$X_{norm} = \frac{X_{max} - X}{X_{max} - X_{min}} \tag{3.1}$$

Then, the two normalised indices are combined into an overall index either by summing them or by computing a function of the two where they can be assigned different weights (Equation 3.2). In Equation 3.2, α represents the relative weight of SA and VW normalised. The higher the value of the function, the higher the quality of the segmentation.

$$F = (1 + \alpha^2)\frac{AS_{norm}VW_{norm}}{\alpha^2 AS_{norm} + VW_{norm}} \tag{3.2}$$

OBIA classification

The second step deals with performing an Object-based Image Analysis (OBIA) classification to obtain a classified raster map from the segmented map. The classification algorithm uses radiometric and geometric properties to classify the different segments. In that way, different radiometric statistics are computed for all segments in the three RGB bands composing the map. These include the mean, the median, the variance, the first and third quartile. As for the geometric properties, the compact square (CS), the compact circle (CC) and the fractal dimension (FD) are used. CS and CC respectively compare the compactness of a segment with a square and a circle. The higher the value, the more the segment shape resembles a square or a circle [Wirth, 2004]. As for FD, it represents the rate at which the perimeter of the segments increases when the measurement scale is reduced (Figure 3.2). It is a measure of the irregularity of the boundary of a segment [Wirth, 2004]. Those three measures are computed as follows:

$$CS = 4\frac{\sqrt{area}}{perimeter}$$

$$CC = \frac{perimeter}{2\sqrt{\pi area}} \tag{3.3}$$

$$FD = 2\frac{\log(perimeter)}{\log(area + 0.0001)}$$

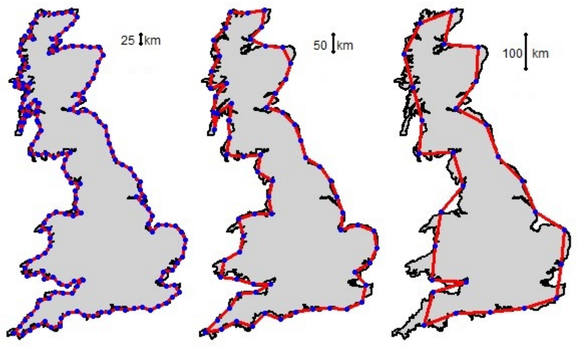


Figure 3.2: Measure of the fractal dimension of the Great Britain's border. From right to left, the fractal dimension is the rate at which the perimeter increases while the measurement scale is reduced. The distance between two adjacent blue dots represents the length of the measurement scale. The colored outlines represent the new perimeters after changing the measurement scale. Figure from Gurung [2017].

As a supervised machine learning approach is used, training data are required. To facilitate the creation of the training map, the method of Gobbi et al. [2019] makes use of training data points. Using a GIS, a number of points can be easily generated in the geographic extent covering the input historical map and the points are manually assigned the land use/land cover (LULC) class in which they fall. With a point-in-polygon procedure using the data points and the segmented map, the point labels can be transferred from the points to the segments in which they fall and a training map is then obtained. When the training map is ready, the classification procedure can start. Three types of classification techniques are used to classify the segments:

- Random forest: Random forest modelling is a machine learning method based on individual decision trees. Each tree predicts a class label (i.e. LULC) based on the values of the independent variables (i.e. all radiometric and geometric attributes). A vote over all trees is then performed to determine the final classification result [Breiman, 2001].

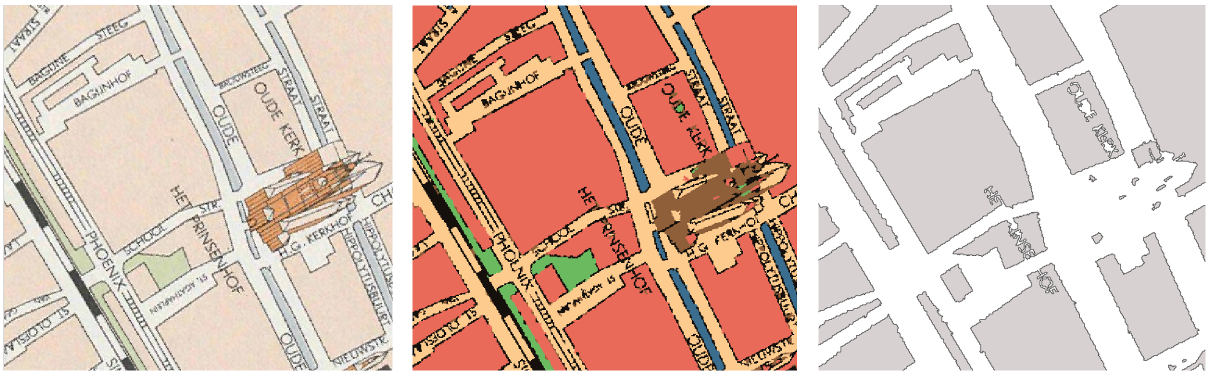
- K-nearest neighbors: K-nearest neighbors is a simple machine learning technique which simply determines the class label of a segment based on the K-closest training examples in its neighbourhood. The final classification result is voted among these K neighbours [Lee, 2019].

- Support vector machine: Support vector machine is a type of classification methods based on kernel functions. The algorithm tries to identify the optimal hyperplane in the feature space to classify unknown data. The simplest models make use of linear kernels but non-linear ones such as radial kernels can be used to model more complex interactions in the dataset [Gholami and Fakhari, 2017].

Each of this classifier assigns a certain LULC class to each segment of the segmented map. The classification results are then combined using a majority vote. From this step, a classified raster map is created.

Text removal

Historical maps usually contain various textual features and symbols which are overlaid over the different geographic features of the map (Figure 3.3a). This can make the vectorisation of historical maps even more challenging as the vectorisation of a geographic feature can be disrupted by the presence of a textual feature overlaid onto it. As a result the geographic feature will be split into different parts (Figure 3.3c). To avoid it, a cleaning procedure needs first to be applied onto the classified raster map to remove the textual features. Gobbi et al. [2019] proposes a method which consists in applying a low-pass filter over the classified raster map and replacing the value of the pixels from the textual feature (or symbol) class by the mode value in their surrounding. The process is repeated until all pixels of the textual feature class have been reclassified or when the maximum number of iterations has been reached.

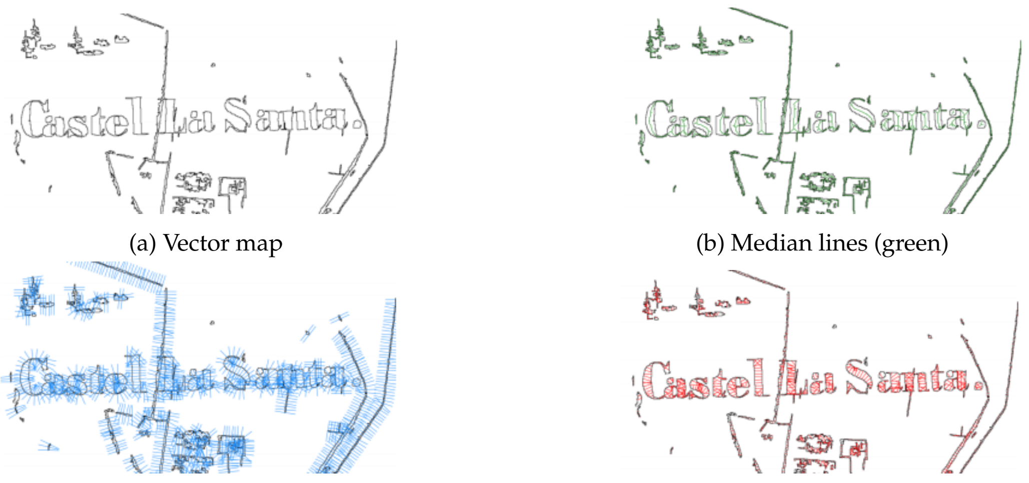


(a) Historical map (Delft 1961) (b) Classified raster map (c) vectorised building plots

Figure 3.3: Effect of textual features on the extraction of building plots from an historical map. In b, Red = buildings, blue = canals, green = vegetation and light orange = streets.

In this process, the size of the moving window of the low-pass filter is important because if the moving window is too small, the mode value could still be the value of the textual feature class. On the contrary, with large moving windows the text segment can be reassigned the value of pixels that are too far away from it. Ideally, the size of the moving window should be just larger than half the shortest width of the text segment to be replaced. In their work, Gobbi et al. [2019] shows that the minimum size of the moving window can be found by measuring the size of the text segments. In this way, the median line of each segment is drawn and perpendicular transects are created evenly along it. After intersecting the transects with the edges of the text segments, the length of the transects inside the text segments can be computed in pixels unit. The maximum transect length ($Transect_{max}$) is used in Equation 3.4 to compute the minimum size of the moving window (MV_{size}). Figure 3.4 shows the whole process to extract the transects lengths.

$$MV_{size} = \frac{Transect_{max}}{2} + 1 \tag{3.4}$$



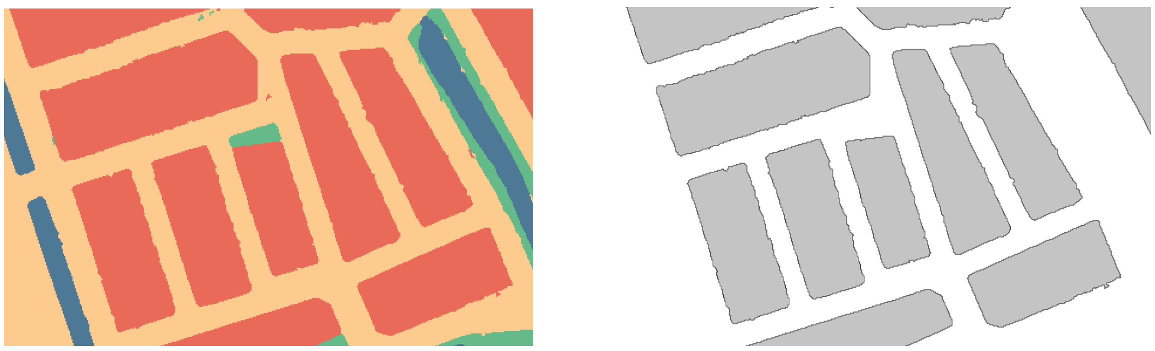
(a) Vector map (b) Median lines (green)

(c) Transects (blue) (d) Transects inside text, symbols and lines (red)

Figure 3.4: Process for computing the minimum size of the moving window necessary to remove textual features and symbols from a raster map. Figure from Gobbi et al. [2019].

3.2.2 Vectorisation and generalisation

Once the cleaned classified raster map is obtained, the next step regards the vectorisation of the patches of pixels representing the building plots. These are simply vectorised using the polygonize tool from the Geospatial Data Abstraction Library (GDAL). GDAL polygonize is based on a region-growing algorithm which creates polygons from adjacent pixels sharing the same value. As shown by Figure 3.5, the vectorisation procedure creates polygons with "stair-like" boundaries as the boundary of the polygons follows the sides of the pixels on the edges of the patches that were vectorised. Thus, vectorising the building plots is not enough, their contour must also be smoothed with some generalisation algorithms. In this thesis, several generalisation methods have been implemented and combined in different ways. A comparison of these different methods can be found in Chapter 5. The following subsections detail the generalisation workflow that provided the best results.



(a) Classified raster map. Red = buildings, blue = canals, green = vegetation and light orange = streets. (b) Vectorised building plots

Figure 3.5: Example of vectorised building plots extracted from a classified raster map

Generalisation using an α-shape operator

The first generalisation method applied on the vectorised building plots was developed by Arteaga [2013]. Their generalisation algorithm is based on an α-shape operator. From a conceptual point of view, the α-shape is a generalisation of the convex hull of a polygon [Ledoux et al., 2020]. Any edge in an α-shape has a length inferior to the α parameter. Thus by changing the α parameter, different α-shapes are obtained. When α is set at ∞, the α-shape is equal to the convex hull of the polygon, while low α values will lead to concave shapes with cavities [Ledoux et al., 2020]. Figure 3.6 illustrates the process of reducing the α parameter.

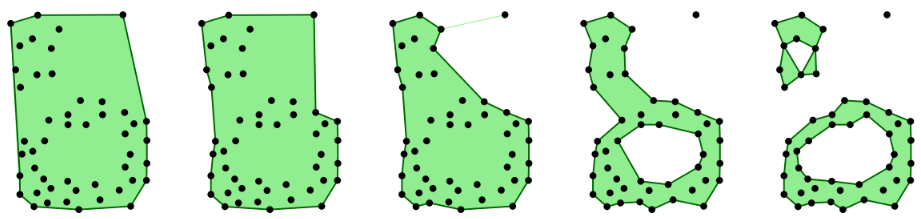


Figure 3.6: α-shape of a set of points after reducing the α parameter, from left to right. Figure from Ledoux et al. [2020].

The algorithm of Arteaga [2013] relies on the following steps:

1. It starts by generating a sample set of points inside the polygon to be generalised choosing randomly between one of these sampling methods: regular sampling, random sampling, random stratified sampling and sampling using an hexagonal grid.
2. An α-shape of the set of points is generated with a user-defined α parameter.
3. The α shape is converted to a circular graph where the vertices and the edges of the shape are respectively the nodes and the edges in the graph. If this step is not successful—for instance, the graph is not circular or not connected—the algorithm goes back to step 1 and repeats the process with a different sampling method.
4. The graph is transformed to generate a single circular graph by removing nodes of degree 1 and by keeping the largest cycle in the graph.
5. Eventually, the single circular graph is simplified by reducing the number of edges.

Generalisation based on buffered lines

The first generalisation method applied smooths the contour of the vectorised building plots but it is not enough to completely remove all irregularities in the building plots contour. The generalisation method developed by Commandeur [2012] is thus used to further smooth the contour of the building plots after the first generalisation procedure is implemented. The generalisation algorithm of Commandeur [2012] aims at merging the line segments of a polygon boundary which are nearly parallel. It starts by converting the set of line segments constituting the polygon boundary to a set of pairs made of (1) the segment line equation and (2) the segment length. Then, each pair is stored in a buffer where the segment length is used as a measure of the buffer importance. The generalisation procedure works by merging the different buffers starting from the one with the highest importance. When two buffers are merged, the averaged line of all lines present in the two buffers is computed and used to represent the new buffer resulting from the merge operation. To be merged, two buffers must fulfill these two conditions:

1. All points in the buffers to be merged are within a user-defined distance threshold ($\in_{distance}$).
2. The angle between the averaged lines of the buffers is lower than a user-defined threshold ($\in_{angle}$).

Figure 3.7 illustrates the merge operation of two buffers respecting these two conditions. When no more buffers can be merged, the generalised lines are created by averaging the lines contained in the buffers with the corresponding segment lengths used as weights. The algorithm also makes it possible to enforce perpendicularity and parallelism between the final generalised lines.

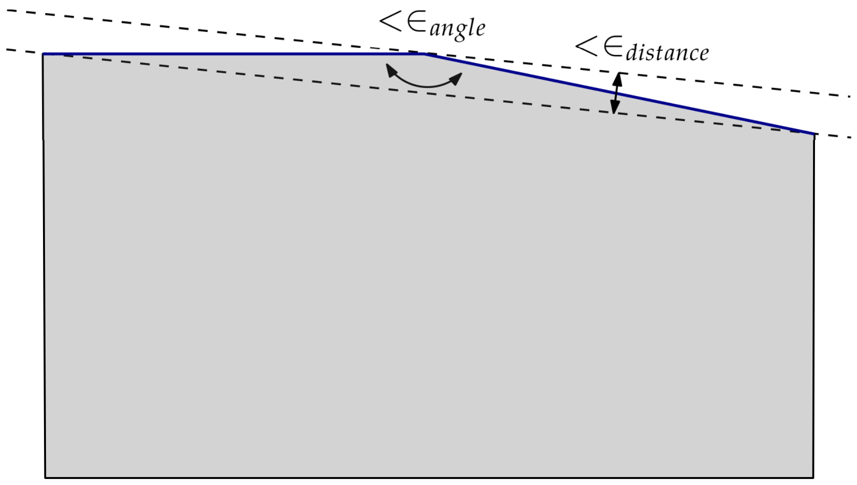


Figure 3.7: Polygon generalisation based on buffered lines. The two buffers (dashed lines) containing the blue lines are merged if the angle they form is within $\in_{angle}$ and if they are separated by a distance below $\in_{distance}$. Figure adapted from Commandeur [2012].

3.2.3 Results assessment

An important step in the building plots extraction method is the assessment of the results. In theory, any building footprints extraction method would work on any historical map, but the quality of the results will vary greatly depending on the method. The choice of the most optimal one must be driven by the quality of the results. In this thesis, different metrics are derived to assess the whole building plots extraction process, i.e. the map digitalisation and the vectorisation and generalisation steps. The following subsections detail the metrics used for assessing the quality of the results.

Performance assessment

Assessing the performance of the building plots extraction method consists in assessing how well the building plots from the input historical map were identified and classified as building plots. To do so, ground truth are needed. Using a GIS, a ground truth point is manually added in each building plot on the input historical map. A point-in-polygon procedure with the ground truth data points and the extracted building plots can then be used to count the number of buildings which were properly classified and the number of buildings which were misclassified. With this information, three metrics are computed using Equation 3.5: the precision, the recall and the F-score [Sun et al., 2020]. The precision represents the percentage of features classified as building plots which are indeed building plots in the ground truth, while the recall is the percentage of ground truth building plots which were indeed classified as building plots. High precision does not mean high recall and the reverse is also true. For instance, if all geographic features of the map are classified as building plots, then the recall will be high because all ground truth building plots have been identified but the precision will be low because there are a lot of misclassifications among the features classified as building plots. To take both the precision and the recall into consideration, the F-score is computed as a mean of the two [Sun et al., 2020].

$$
\begin{aligned}
precision &= \frac{\sum properly\ classified\ building\ plots}{\sum\ classified\ building\ plots} = \frac{\sum true\ positive}{\sum true\ positive + \sum false\ positive} \\
recall &= \frac{\sum properly\ classified\ building\ plots}{\sum\ ground\ truth\ building\ plots} = \frac{\sum true\ positive}{\sum true\ positive + \sum false\ negative} \\
F\text{-}score &= 2\,\frac{precision\ *\ recall}{precision + recall}
\end{aligned}
\tag{3.5}
$$

Shape assessment

Measuring the performance indicates how well the building plots were identified but it does not tell if the shape and contour of the building plots were properly recovered from the input historical map. To assess it, a ground truth dataset is generated by manually digitising a subset of building plots from the historical map. Then, the shape of the building plots extracted automatically is compared with the shape of the ground truth building plots using two metrics: the difference between turning functions and the shape similarity based on buffers.

The difference between turning functions is based on another representation of the polygon than the usual representations. Indeed, instead of representing a polygon with a list of vertices or a list of edges, it can also be represented with a list of pairs of angles and edge lengths obtained when turning around the polygon [Fan et al., 2014]. Its vertices can be seen as turning points, and at each turning point, it is possible to compute the counter-clockwise angle between the two adjacent edges. For instance, on Figure 3.8, ψ represents the counter-clockwise angle at vertex a. If the angle at each turning point is accumulated and combined with the normalised accumulated edge length, then the combination of the angle-edge length at each turning point can be used to uniquely describe the shape of the polygon. This is what does the shape turning function. The turning function expresses how the accumulated tangent angle changes with the normalised accumulated length [Fan et al., 2014; Arkin et al., 1991]. For instance, Figure 3.8 shows a polygon and its turning function. An increase in the accumulated angle (y-axis) translates a convex edge change while a decrease indicates a concave edge change.

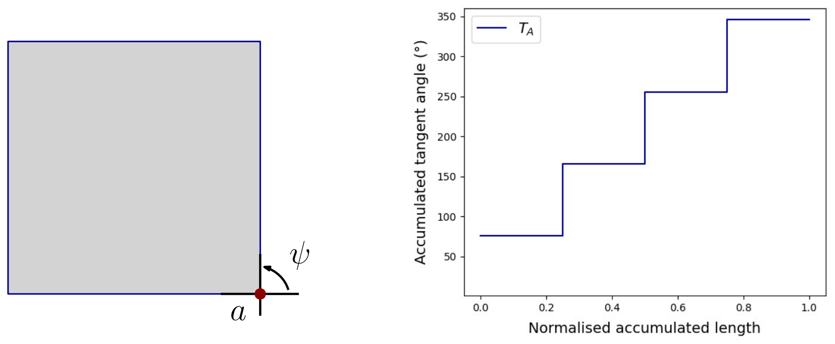


Figure 3.8: Representation of a polygon (left) using its turning function (right)

As each shape has its own turning function, two shapes can be compared by computing the difference between their turning functions. It is computed with Equation 3.6, where A and B are the two shapes to be compared, T_A, T_B their turning functions and l the normalised accumulated length [Arkin et al., 1991]. The smaller the difference, the more the shapes are similar. This metric is scale and rotation invariant so the same shapes at two different scales will be found similar using this metric [Arkin et al., 1991]. In our case, it means that if the edge pixels of a given building plot were not properly vectorised but the extracted building plot is a perfect square as its ground truth, then the shapes will be similar according to this metric, even tough they are different in size. However, the metric is really sensitive to noise; if the contour of the extracted building plot has spikes or small defects, then the difference with its ground truth will be large [Arkin et al., 1991]. Therefore, this metric is used to assess whether the building plots extraction method preserved the angles without broken corners and whether the contour of the extracted building plots is bumpy.

$$S(A,B) = d(A,B) = ||T_A - T_B||_2 = \left(\int_0^1 (T_A(l) - T_B(l))\right)^{\frac{1}{2}} \tag{3.6}$$

The shape similarity based on buffers is computed between two polygons as the average of the percentages of one polygon within the buffer area of the second polygon (Figure 3.9) [Samal et al., 2004]. The measure is computed with Equation 3.7, where $A\phi B$ is the percentage of the ground truth building plot A comprised into the buffer area of the extracted building plot B, while $B\phi A$ represents the percentage of B comprised into the buffer area of A [Samal et al., 2004]. This measure is not sensitive to spikes or small defects in the contour but it mostly checks if two shapes fit with each other. In fact, this measure is rotation invariant but not scale neither translation invariant [Samal et al., 2004]. Therefore, it is useful to check if the extracted building plots match well with the ground truth building plots on the historical map. For instance, it helps determine if the edge pixels of the building plots were also vectorised in addition to the interior of the building plots, or if the extracted building plots are smaller than their ground truth because the edge pixels were not properly identified. The measure value is included between 0—the shapes are not even intersecting—and 100—the shapes are perfectly matching.

$$\sigma(A,B) = \frac{A\phi B + B\phi A}{2} \tag{3.7}$$

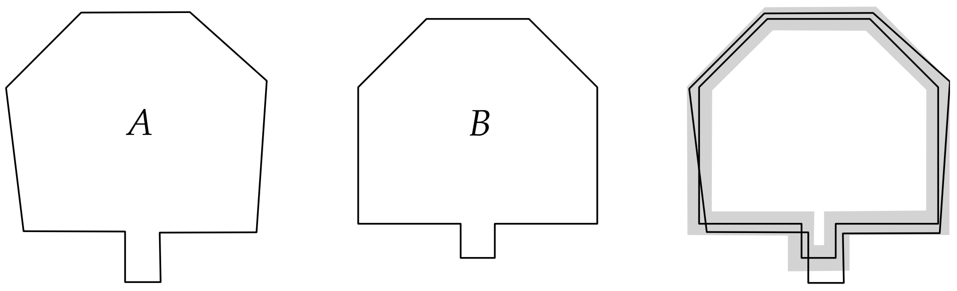


Figure 3.9: Shape similarity between polygons based on buffers. From left to right: two polygons A and B and the intersection of A within buffered B. Figure adapted from Samal et al. [2004].

These two metrics are computed for a sample of the extracted building plots. For each metric, the average is kept and used as a representative value for the whole dataset. It should be noted that the aim of this step is to assess how well the shapes were preserved by the building plots extraction method, independently from the performance of the classification, so these metrics are only computed for building plots that were properly classified.

Metrics

In addition to these measures for assessing the quality of the results, a series of metrics is computed after each step in the building plots extraction method to keep track of the effect of the different steps on the extracted building plots. In addition, they are used to analyse and compare the results of the building plots extraction method implemented with different historical maps. Table 3.1 describes those different metrics [Wirth, 2004]. Some of them might be repetitive as they provide similar information. By computing and comparing them after each step of the methodology, the most relevant ones will be identified.

Metric	**Description**	**Computation**
Area	Area of the object.	–
Perimeter	Perimeter of the object.	–
Number of vertices	Number of vertices in the object boundary.	–
Presence of holes	Boolean that indicates whether the object has holes or not (1 vs 0).	–
Compactness	Ratio between the object area and the area of a circle with the same perimeter. The metric value is comprised between 0 and 1—the higher the value, the more compact the shape. Elliptic objects and objects with an irregular contour have low compactness.	$\frac{4\pi\ area}{(perimeter)^2}$
Elongation	Ratio between the length and width of the bounding box (*bbox*) of the object. The metric value is comprised between 0 and 1. Squared and circular objects have a ratio value close to 1, while more elongated objects have low ratio values.	$\frac{length_{bbox}}{width_{bbox}}$
Roundness	Ratio between the object area and the area of a circle with the same perimeter than the convex hull of the object. The metric value varies between 0 and 1 with circular objects closer to 1. The metric is insensitive to noise, so it is not sensitive to irregularities in the object contour.	$\frac{4\pi\ area}{(convex\ perimeter)^2}$
Convexity	Ratio between the perimeter of the convex hull of the object and the object perimeter. This measure indicates whether an object is convex—ratio value equal to 1—or concave—ratio value below 1.	$\frac{convex\ perimeter}{perimeter}$
Concavity	Boolean that indicates whether the object is concave or not (1 vs 0). This measure is based on the convexity measure previously described.	–
Solidity	Ratio between the object area and the area of the convex hull of the object. This measure evaluates the density of an object. The metric value is comprised between 0 and 1. Objects with a low ratio value either have holes or irregularities in the boundary.	$\frac{area}{convex\ area}$
Rectangularity	Ratio between the object area and the area of the object minimum bounding rectangle (MBR). The metric value varies between 0 and 1. The higher the value, the more the object is rectangular.	$\frac{area}{area_{MBR}}$

Table 3.1: Metrics computed for the extracted building plots after each step of the building plots extraction method

These metrics are computed for each extracted building plot and statistics such as the sum, the mean, the median and the standard deviation are derived as representative values for the whole dataset. In addition, the total number of building plots is also computed after each step of the building plots extraction method.

3.3 Reconstruction of individual footprints from building plots

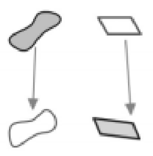
One challenging task in the reconstruction of historical 3D city models from historical maps is that historical maps do not always depict buildings as individual building footprints. Instead, building footprints might be aggregated and represented as building plots. Thus, before reconstructing the 3D buildings, the building plots must first be split into individual building footprints. In this thesis, two processes are used for this purpose: maps alignment and 2D procedural modelling. In that way, the building plots can be subdivided either by combining both processes, or by using only the 2D procedural modelling process. The choice of using one solution or the other depends on the availability of 2D and 3D building datasets with specific attributes (e.g. building year of construction) which are needed to implement the maps alignment process (Section 4.1.2). If these datasets are not available, the maps alignment step is not used and it is still possible to reconstruct historical 3D city models for the study area using only the 2D procedural modelling process. This option in the implementation makes the difference between the complete version of the methodology—using the maps alignment process—and the shortened version—skipping the maps alignment (Chapter 3). The following subsections describe the combined use of the two processes as using the 2D procedural modelling process alone only requires skipping the maps alignment step.

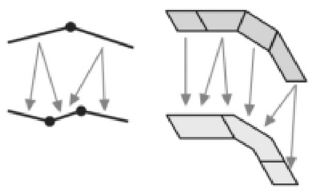
3.3.1 Maps alignment

Historic cities sometimes still have very old constructions that are more than a century old. Thus, some buildings that are still present nowadays might have already existed when the input historical map was made. If these buildings can be identified, then their current building footprints could be used to subdivide the historical building plots. This can be achieved using a maps alignment procedure.

The goal of maps alignment is to identify the geographic features that are present in two or more maps—namely the aligned features. It is usually performed by (i) selecting the properties that will be used to compare geographic features from different maps, (ii) computing similarity measures based on those properties and (iii) combining the similarity measures to determine the aligned entities [Sun et al., 2020]. There are three types of maps alignment (Figure 3.10) [Xavier et al., 2016]:

- **One-to-one alignment (1:1):** In a one-to-one alignment between map A and map B, any geographic feature of map A has at most one matching feature in map B.
- **One-to-many alignment (1:n):** In a one-to-many alignment between map A and map B, a geographic feature of map A can have several matching features in map B, but the reverse is not true.
- **Many-to-many alignment (m:n):** In a many-to-many alignment between map A and map B, a geographic feature in any of the two maps A or B can have several matching features in the other map.


(a) One-to-one alignment (1:1)


(b) One-to-many alignment (1:n)

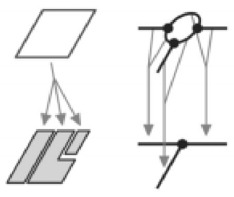

(c) Many-to-many alignment (m:n)

Figure 3.10: Types of maps alignment. Figure modified from Xavier et al. [2016].

In this study, a one-to-many alignment process is used to match the extracted building plots with the building footprints from the current dataset of the city of interest. Two types of method can be used to align the current building footprints dataset and the extracted building plots:

- **Buffer growing:** The buffer growing method was first developed by Walter and Fritsch [1999] for performing 1:n alignments. One geographic feature a is matched with a geographic feature b from another map if a lies in the buffer zone of b (Figure 3.11). The buffer distance is important as it can greatly influence the number of matches. In Walter and Fritsch [1999], it is determined by manually measuring the maximum distance between corresponding geographic features in the two original maps. Ying et al. [2011] proposed to compute the buffer distance with Equation 3.8, where SR_A and SR_B are the spatial resolutions of the input maps and k is a user-defined parameter.

- **Overlapping area:** Other research uses a similarity measure based on the overlapping area between geographic features to be aligned [Tong et al., 2009; Fan et al., 2014]. The similarity measure $\sigma(a, b)$ between a feature a and a feature b is computed with Equation 3.9. To be matched, the similarity value between two geographic features must be above a user-define threshold.

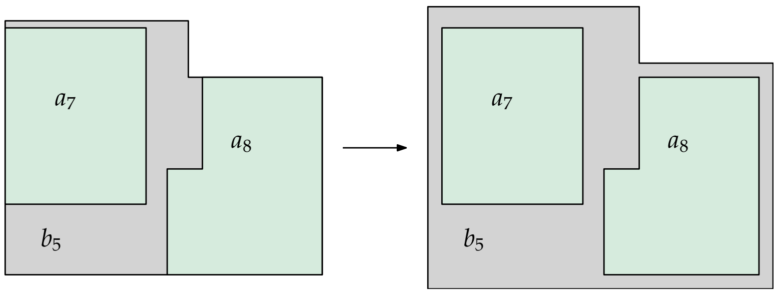


Figure 3.11: One-to-many alignment based on the buffer growing method. $a7$ and $a8$ from map A are located in the buffer zone of $b5$ from map B.

$$buffer_{distance} = k\sqrt{SR_A^2 + SR_B^2} \tag{3.8}$$

$$\sigma(a, b) = \frac{Area_{overlap}}{min(area_a, area_b)} \tag{3.9}$$

In this thesis, a combination of both the buffer growing and the overlapping area methods is used. To check if an historical building plot a can be matched with a selected building footprint b, the similarity measure $\sigma(a, b)$ is computed using the overlapping area between the building footprint and the *buffered* building plot (Equation 3.9). The similarity measure value is then compared with a user-defined threshold.

Maps alignment methods have been widely used for aligning different maps from the same time period [Walter and Fritsch, 1999; Ying et al., 2011; Tong et al., 2009; Fan et al., 2014]. As the aim here is to match datasets from different epochs, the current building footprints have first to be selected based on their year of construction before implementing the maps alignment process. Indeed, the maps alignment process should only be implemented with the building footprints having a year of construction inferior to the date at which the historical map was made. This is important to avoid the case where the buildings of an historical building plot were completely destroyed and replaced with new constructions. In this case, without the year selection, the maps alignment would still match the historical building plot with the current building footprints at the same location, even tough they are recent buildings and sometimes have a very different construction design than the one in place at the historical map date.

This whole step produces two main outputs: the current building footprints that were found to be aligned with any building plot and the ones that did not find any correspondences. The aligned building footprints are kept to subdivide their corresponding building plot into individual building footprints in combination with another process: 2D procedural modelling.

3.3.2 2D procedural modelling

Even tough some buildings dating from the historical map epoch still remain nowadays, there are also some buildings that have been demolished and replaced with new constructions by the years. The footprint of these buildings still need to be recovered. To do so, a process called procedural modelling is used. In computer graphics, procedural modelling is a process for reconstructing 3D objects in an automatic and generative way [Ullrich et al., 2010]. In this thesis, the term of procedural modelling is also extended to the reconstruction of 2D objects in a procedural way. In this way, 2D procedural modelling is used to reconstruct individual building footprints automatically from building plots.

The decomposition method applied for subdividing a given building plot depends on its size and shape. Five main cases are developed to handle the subdivision of any input building plot (Figure 3.12). The different cases tend to create building footprints with desirable characteristics: parallel edges and right angles. Several user-defined variables lead the decomposition process: the minimum and maximum facade length and the minimum and maximum building depth of the building footprints to be created. These parameters are used to generate randomness among the building footprints created, in order to make the results more realistic. The following subsections describe the implementation of the five cases.

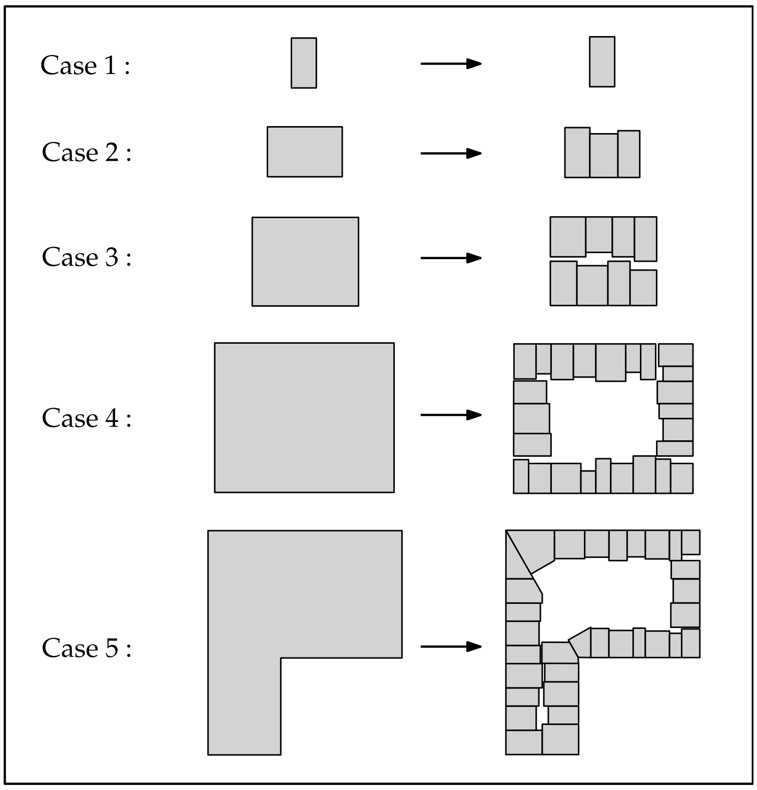


Figure 3.12: Overview of the five 2D procedural modelling cases used to subdivide a building plot into building footprints

Case 1: the building plot is made of a single building footprint

The area of the building plot is computed and compared with an area threshold. If the building plot area is below that area threshold, the building plot is considered to be already a building footprint and it is kept as is. As area threshold, the average area of the aligned building footprints is used—if the maps alignment process was implemented. Otherwise, that value can be set by the user based on real data.

Case 2: the building plot is made of one single row of building footprints

If case 1 fails, the minimum bounding rectangle (MBR) of the building plot is computed. The shortest edge length of the MBR is compared with the minimum building depth ($depth_{min}$) and if Equation 3.10 is verified, it is assumed that the building plot is constituted of one single row of building footprints. The subdivision process used in this case is shown in Figure 3.13. The longest edge of the building plot is split into segments of different lengths using the facade length range of values given as input by the user. Then, the algorithm walks along each segment and generates at its end point a perpendicular line segment. The building footprints are created by intersecting the perpendicular line segments with the opposite edge of the building plot (point P_i on Figure 3.13). Randomness is generated in the facade depth by moving P_i along the perpendicular line segment.

$$length_{shortest\ edge}\ \%\ depth_{min} \leqslant 1 \tag{3.10}$$

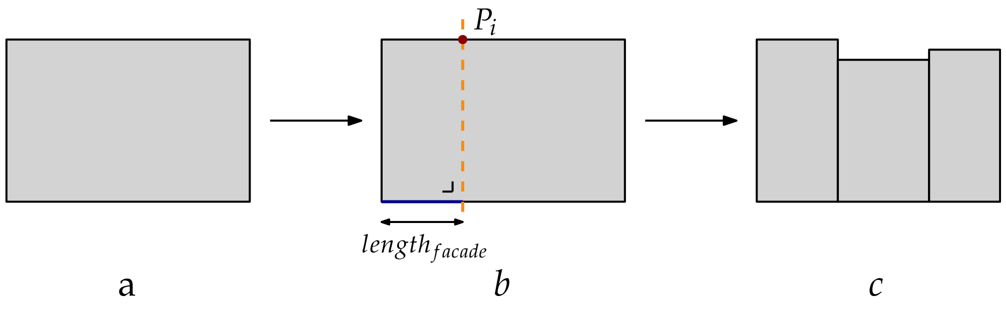


Figure 3.13: Decomposition of a building plot into building footprints using case 2—the building plot is made of one single row of building footprints. a shows the input building plot, b the decomposition process and c, the resulting building footprints.

Case 3: the building plot is made of two rows of building footprints

If Equation 3.10 is not verified, Equation 3.11 is checked and if it holds, it means that the building plot is made of two rows of building footprints. In this case, the longest median line of the building plot is used to split it into two polygons which represent the two buildings rows. Each polygon is processed individually and split into individual building footprints with the algorithm used for case 2 (Figure 3.14).

$$length_{shortest\ edge}\ \%\ depth_{min} = 2 \tag{3.11}$$

Case 4: the building plot contains an interior courtyard and/or gardens

If none of the previous cases has been verified, it means that the building plot is too big to be split entirely into building footprints. In that case, the building plot also contains an interior courtyard or a continuous space occupied by the gardens of the different buildings. The building footprints are then arranged around that interior courtyard and/or gardens. To reproduce this arrangement, an inward offset polygon is created in the input building plot.

The offset polygon at a distance t of a source polygon is a polygon which has the same orientation and all its edges at a distance t from the source polygon [Kim, 1998]. Creating an offset polygon is usually done using the straight skeleton of the source polygon. This latter is obtained by shrinking the polygon edges in a self-parallel way, towards the interior of the polygon and at a constant speed. During the

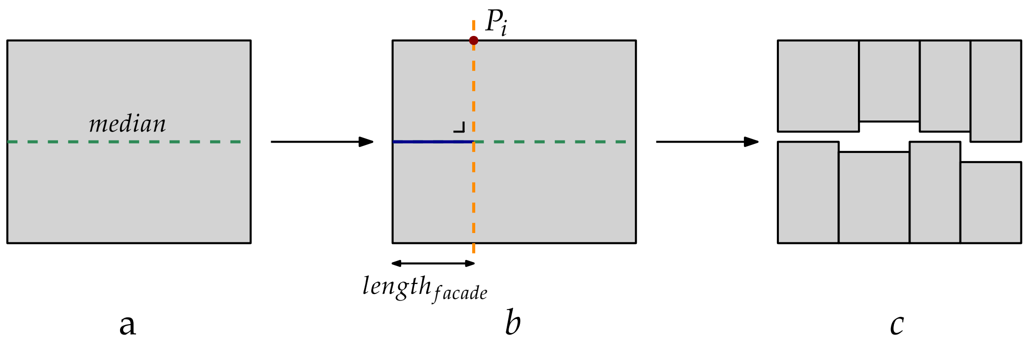


Figure 3.14: Decomposition of a building plot into building footprints using case 3—the building plot is made of two rows of building footprints. a shows the input building plot and its median, b the decomposition process and c, the resulting building footprints.

propagation process, the vertices of the source polygon are moved along the angular bisectors of the polygon edges and when they meet non-incident offset edges, the straight skeleton is created (step b in Figure 3.15) [Held and Palfrader, 2017]. From the straight skeleton it is then possible to create any offset polygon of the source polygon by sliding along the straight skeleton edges overlapping with the angular bisectors (step c in Figure 3.15) [Held and Palfrader, 2017].

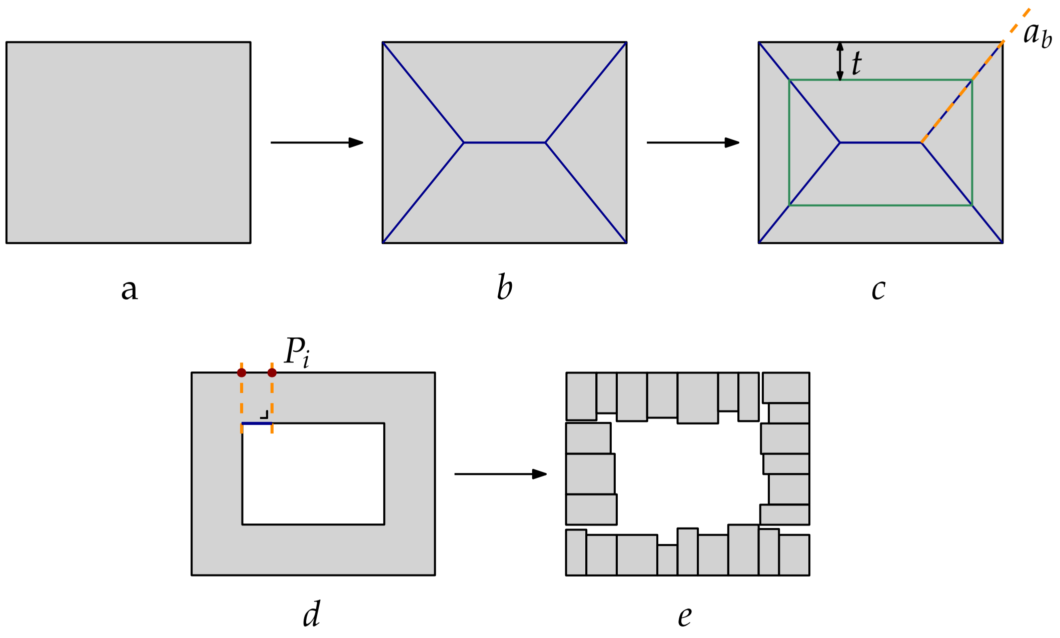


Figure 3.15: Decomposition of a building plot into building footprints using case 4—the building plot contains an interior courtyard and/or gardens. a shows the input building plot, b its straight skeleton (blue) and c, an offset polygon (green) at a distance t created by sliding along the angular bisectors a_b. d shows the decomposition process used in case 4 and e shows the resulting building footprints.

Using this technique, an inward offset polygon is created in the building plot at a distance equal to the maximum building depth. That offset polygon is used to create an interior boundary in the source polygon. The algorithm walks along that interior boundary and each of its line segments is processed using the following steps (steps d and e in Figure 3.15):

1. The line segment is split into sub-segments using the facade length range of values.
2. For each sub-segment, perpendicular lines passing through its endpoints are generated and intersected with the source polygon exterior boundary.

3. The intersection points (P_i on Figure 3.15) on the exterior boundary and the sub-segment end points define a building footprint. Randomness is added in the facade depth using the building depth range of values provided by the user.

Case 5: the building plot has a concave shape

A special case has been designed for concave building plots. These are split into convex polygons which are then processed individually using one algorithm from cases 1 to 4. The convex decomposition is made using the algorithm of Bayazit [2009]. It works by iterating over the vertices of the concave polygon until it finds a reflex vertex i. The edges incident to this vertex are then prolonged until they meet the polygon exterior boundary on another edge (step a in Figure 3.16). From this step, two intersection points are created on the polygon exterior boundary. Two different operations can then be taken:

- If there is another reflex vertex on the exterior boundary in the range defined by the two points of intersection, this reflex vertex and i are connected and used to split the concave polygon.
- If there is no other reflex vertex in that range, a Steiner point is added in the center and connected to i to split the concave polygon.

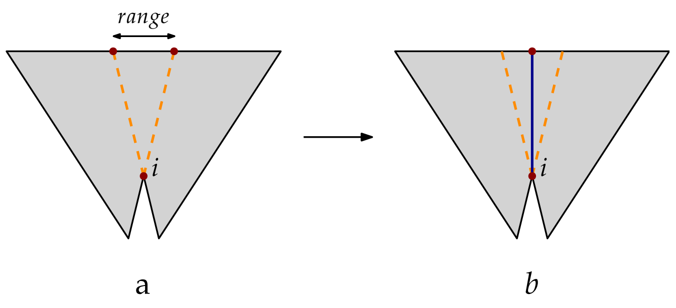


Figure 3.16: Convex decomposition of a concave polygon. In a, the reflex vertex i is found and the incident edges are extended. In b, a Steiner point is created and connected to i. Figure adapted from Bayazit [2009].

Once the concave building plot has been decomposed into convex polygons, these are subdivided into building footprints with one of the other cases. Then, a final check is made to remove all building footprints that do not have any edges on the exterior boundary of the input concave building plot. The whole process is summarised in Figure 3.17.

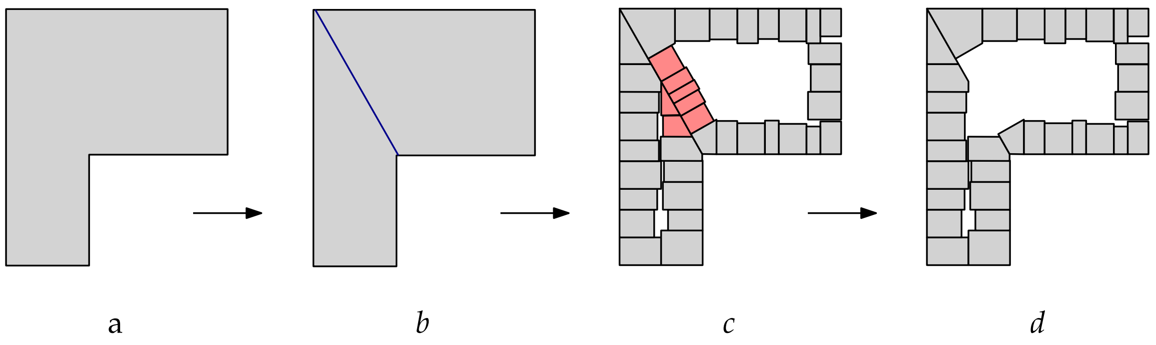


Figure 3.17: Decomposition of a building plot into building footprints using case 5—the building plot has a concave shape. a shows the concave building plot, b its decomposition into convex polygons and c, the decomposition of these convex polygons into building footprints. d shows the resulting building footprints obtained after removing the red-colored building footprints in c as they do not have any edges on the exterior boundary of the original building plot.

3.3.3 Combining the two building footprints datasets

Once the building plots have been split into building footprints using 2D procedural modelling, an overlap test is implemented. The building footprints generated with 2D procedural modelling for a given building plot are kept only if they do not overlap with any building footprint that was matched with the building plot with the maps alignment process. In this way, the final building footprints used to subdivide a given building plot are of two different types:

- The current building footprints aligned with the building plot. From this point, these are referred to as "*aligned building footprints*".
- The non-overlapping building footprints generated with 2D procedural modelling (2PM). From this point, these are referred to as "*2PM building footprints*".

The output from this step is thus a dataset containing two types of building footprints: the aligned ones and the 2PM ones. Note that in the shortened version of the methodology in which the maps alignment process is not implemented, the output dataset is only made of 2PM building footprints.

3.4 LoD2 buildings generation

The last step of the methodology deals with the processing of the building footprints dataset created in the previous steps to generate LoD2 buildings. The final historical 3D city model is reconstructed by (i) generating height attributes for all building footprints, (ii) reconstructing LoD2 buildings with 3D procedural modelling and (iii) generating a valid CityJSON file. These steps are described in the following subsections.

3.4.1 Height inference

For reconstructing 3D geometries from the building footprints dataset, the building footprints need to have two types of reference heights: a ground height and a roof height. The aligned building footprints already have height attributes. Thus, only the 2PM building footprints need to be assigned reference ground and roof heights.

Assigning a ground height

In historical 3D city modelling, studies usually make use of two different solutions to assign a ground height to the building footprints; they either use a current elevation dataset, or they reconstruct an historical Digital Elevation Model (DEM). The procedure to reconstruct an historical DEM is based on the assumption that the elevation in land use classes such as fields and pastures has not changed; it is unlikely that the landscape has been modified by large geomorphologic processes since the input historical map was made [Nobajas and Nadal, 2015]. By comparing historical and current land use maps, the unchanged pastures and fields areas can be identified. The final historical DEM can then be reconstructed by interpolating elevation points extracted in these unchanged areas. Some research argues that using a current elevation dataset instead of recreating an historic one leads to inaccurate ground heights, as human activity is likely to have generated elevation changes in densely populated areas [de Boer, 2010; Nobajas and Nadal, 2015]. However, most of these studies focus on the reconstruction of complete virtual landscapes. In this thesis, the focus is set on the urban context. Thus, the input historical maps used in this study almost only depict the city extent with some pastures and fields located around the city walls. Reproducing an historical DEM would require interpolating the elevation of the whole city center, resulting in an interpolated historical DEM of low accuracy for the urban context. An historical DEM could also be reconstructed from the input historical maps if they report heights or isolines, but this is not the case for the historical maps used in this study. For these reasons, it was chosen to derive the ground height from a current elevation dataset.

The method of Dukai et al. [2019] is used to assign a ground height to the 2PM building footprints. That methodology makes use of a current 3D point cloud made of ground points. A point-in-polygon procedure is implemented to find the ground points located in the building footprints buffered with a distance of 0.5 m. Because of the time difference between the historical maps and the current point cloud, some building footprints do not find any ground points in a buffer zone of 0.5 m. This happens when a building dating from the historical map was completely demolished and replaced with another larger building. To solve that issue, the point-in-polygon procedure is implemented in an iterative way; for each building footprint, the buffer distance is increased until ground points are found in its buffer zone. Using the ground points located in the buffered building footprints, percentile heights (0, 10, 20, 30, 40, 50) are computed and used as reference ground heights.

Assigning a roof height

In the complete version of the methodology—i.e. using the maps alignment process—the method used to assign a roof height to the 2PM building footprints relies on the fact that the aligned building footprints already have height attributes. In that way, the roof height attributes of the 2PM building footprints can be determined using a neighbourhood analysis [Kersten et al., 2012]. This analysis is made using the original building plots. Each original building plot is assigned some roof height attributes using one of these two operations:

- If the building plot was aligned with any building footprint during the maps alignment process, its roof height attributes are computed as the median of the roof height attributes of its aligned building footprints.
- If the building plot was not aligned with any building footprint, the roof height attributes are computed as the median of the roof height attributes of the aligned building footprints in the neighbouring building plots. In some cases, it can happen that the neighbouring building plots were not aligned with any building footprint either. In this case, the building plot is skipped and it is re-investigated when its neighbours have been assigned roof height attributes.

Once a building plot has been assigned roof height attributes, these attributes are transferred to the 2PM building footprints generated to subdivide it. One may wonder why the neighbourhood analysis is made at the building plot level and not at the building footprint level. It is due to the high spatial autocorrelation in the building footprints dataset. The 2PM building footprints tend to be all located close to each other such that they are each other's neighbours. For instance, in Figure 3.18, if the roof height attributes of the 2PM building footprints were defined using the aligned building footprints in their neighbourhood, they would still all have the same roof height attributes as they are each other's neighbours.

In the shortened version of the methodology, this method for assigning the roof height cannot be used as the building footprints dataset only contains 2PM building footprints and the information about the roof height attributes in the neighbours is thus not available. To cope for this issue, the roof height can be assigned in a similar way than the ground height, using a current 3D point cloud with building points, followed by a neighbourhood analysis for building footprints that do not find corresponding building points in the point cloud. Alternatively, the roof height could also be randomly generated based on the regulated roof heights that were in use at the historical map date.

When the building footprints of the dataset all have been assigned reference heights, the next steps regard the processing of those building footprints to reconstruct LoD2 buildings. Two different approaches are implemented for the aligned building footprints and for the 2PM building footprints. For the 2PM building footprints, the 3D reconstruction is made using 3D procedural modelling. As for the aligned building footprints, a current LoD2 3D city model is already available—as part of the optional datasets needed to implement the complete methodology version—and their LoD2 buildings are thus directly extracted from it. Eventually, all LoD2 buildings will be reunited in the same dataset to create the final historical 3D city model.

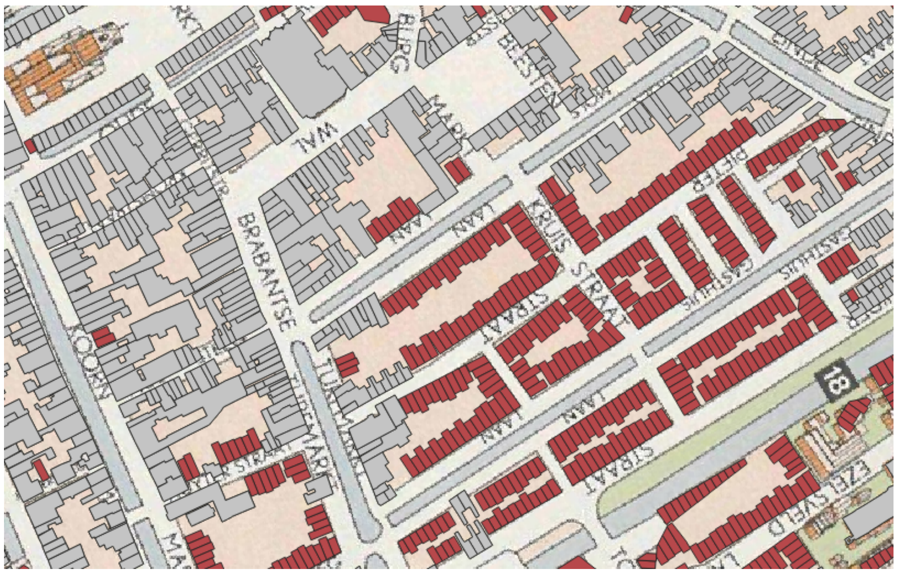


Figure 3.18: 2PM building footprints (dark red) and aligned building footprints (grey)

3.4.2 3D procedural modelling

In computer graphics, procedural modelling is a process for reconstructing 3D objects in an automatic and generative way from a user-defined set of rules [Ullrich et al., 2010]. Procedural modelling is based on shape grammars. According to Stiny [1980], a shape grammar can be defined by four components:

- S, a finite set of shapes;
- I, an initial shape made of shapes in S;
- R, a finite set of user-defined transformation rules and
- L, a finite set of symbols used to create the rules in R.

The creation of a specific shape using a procedural modelling algorithm is made from the interaction of the four components in the shape grammar. In that way, transformation rules in R are created from the set of symbols L, using the equation form in Equation 3.12 [Stiny, 1980; Müller et al., 2006]. In Equation 3.12, a rule α is applied on an input shape in S—i.e. the predecessor—to transform it and create another shape—i.e. the successor. From this, the procedural modelling process is simple to understand; it starts with the initial shape I and applies a transformation rule onto it to create another successive shape. That successive shape becomes the predecessor in Equation 3.12 and another transformation rule is applied onto it to generate another shape. The process goes on until the desired specific shape is obtained. As shown by Figure 3.19, the more rules are added, the more the final shape has details.

$$\alpha : predecessor \rightsquigarrow successor \tag{3.12}$$

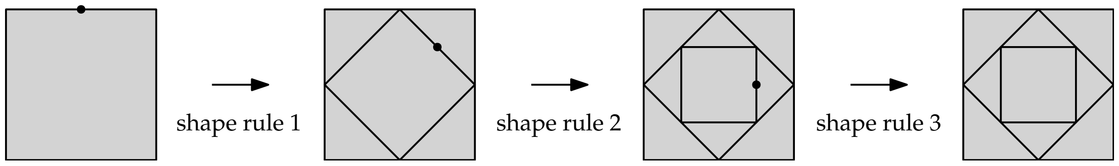


Figure 3.19: Creation of a specific shape using a set of rules. Figure adapted from Stiny [1980].

Since the 2000s, procedural modelling has been used to generate automatically textured 3D city models from vector data using the Computer Generated Architecture (CGA) grammar of Parish and Müller [2001] and Müller et al. [2006]. Their methodology works by reconstructing a mass or crude 3D city model, equivalent to a LoD1 3D city model, from a set of high-priority rules. Then, low-priority rules are used to add details in the output 3D city model and generate LoD2+ buildings. In this thesis, an implementation of the CGA grammar is used to automatically reconstruct LoD2 buildings from the 2PM building footprints. This is described in the following subsections.

Defining new transformation rules

The 3D procedural modelling step is implemented in Blender using the Computer Generated Architecture for Blender (BCGA) addon. This addon is an implementation of the CGA grammar for Blender while the original grammar created by Parish and Müller [2001] and Müller et al. [2006] was developed in the CityEngine software. The BCGA already contains the following transformation rules to reconstruct LoD1–LoD2 buildings:

- — Extrude a 2D face at a user-defined height
- — Decompose a 3D shape into different user-defined parts (top, front, sides)
- — Split a 2D face evenly into different faces in a user-defined split direction
- — Delete a 2D face
- — Copy a 2D face
- — Translate a 2D face with a user-defined translation vector
- — Generate a hip roof from user-defined pitches (Figure 3.20)
- — Generate a mansard roof from user-defined inset values (Figure 3.20)
- — Add a user-defined texture or a material to a 2D face

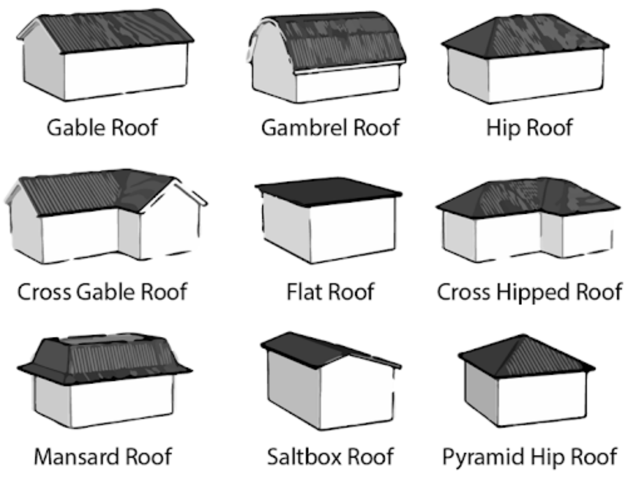


Figure 3.20: Common roof types. Figure from Bondright Roofing Services [nd].

In this thesis, the main additions made to this Blender addon regard the roof-related rules. The main issues with the existing roof rules were spotted and some solutions have been implemented to solve them. These issues and proposed improvements are depicted in Table 3.2. In the following paragraphs, these rules are described with their improvements. Table 3.3 gives a summary of the different user-defined parameters with their values for each roof-related rule.

	Issues	**Improvements**
1	The hip roof rule requires as input user-defined roof pitches for creating the roof. If the user wants the roof to have a certain roof height, this requires beforehand a complex computation of the pitches values that determine the roof height. This is especially true when the original building footprint is not a perfect rectangle, in which case the roof ridge is slanted.	Modify the hip roof rule so that it takes as input the roof height. All additional roof rules are also made that way such that the user can specify a roof height. In cases where the ridge is slanted, the roof height varies and the user-defined roof height becomes the maximum roof height. Note that this functionality fails on irregular building footprints with more than four vertices for which it is difficult to force the roof height.
2	The hip roof rule does not let the user the possibility to specify on which edge of the original building footprint the hip ends of the roof should be created. Instead, this is defined by the order in which the pitches values are passed in the rule. But this order is not always the same as it depends on the order in which the edges of the building footprints are written. Thus, this requires the user to play with each building footprint to recover the order in which to write the pitches.	Modify the hip roof rule so that it can take as input the building footprint edges on which the hip ends should be created. All additional roof rules are also made that way.
3	The hip roof rule can be extended to create gable roofs by setting the right pitches values but this requires user knowledge about gable roofs.	Add one specific rule for creating gable roofs.
4	The number of roof types that it is possible to create is limited.	Add a rule to create an additional roof type. As this thesis mainly focuses on historical maps of the Netherlands, the focus was set on reconstructing historic Dutch roofs. In this way, a rule is added to create crow-stepped gable roofs.
5	The hip roof rule does not work with concave building footprints. Thus it cannot be used to reconstruct cross hipped roofs (Figure 3.20).	This was not fixed. Note that in this study it does not have any implications on the resulting historical 3D city models as the decomposition algorithm in Section 3.3.2 does not produce concave building footprints from the building plots.

Table 3.2: Issues of the BCGA addon and proposed improvements

Roof type	Parameters	Values
Flat roof	Roof height	Based on assigned roof heights
Hip roof	Roof height	Based on assigned roof heights
	Hip ends pitch	Comprised in [$pitch_{min}$: 90°]
	Midpoint street edge	Identified street edge
Gable roof	Roof height	Based on assigned roof heights
	Midpoint street edge	Identified street edge
Mansard roof	Roof height	Based on assigned roof heights
	Inset	Comprised in [0 : $\frac{length_{shortest\ edge}}{2}$]
Crow-stepped gable roof	Roof height	Based on assigned roof heights
	One hip end pitch	Comprised in [$pitch_{min}$: 90°]
	Midpoint street edge	Identified street edge
	Step height	Comprised in [40 : 70 cm] [Mesqui, 1998]
	Facade thickness	0.5 cm

Table 3.3: Overview of the roof types with their user-defined parameters and assigned values

A hip roof (or hipped roof) is a type of roof in which all the roof faces are slanted, sloping downwards from the roof ridge to the building walls. It has two types of faces: the triangular ones—called the hip ends—and the rectangular ones. The BCGA rule relies on two main user-defined parameters: the roof height and the roof pitch of the hip ends. The roof height is used to compute the pitch of the rectangular roof faces using Equation 3.13, where the baseline is the distance measured perpendicularly between the two roof edges of the rectangular roof faces (Figure 3.21). As for the roof pitch of the hip ends, it is not directly dependent of the roof height and it is thus left to the user choice with a high degree of freedom. However, to avoid self-intersection in the output hip roof, the hip end pitch must always be comprised between $pitch_{min}$ and 90° (Figure 3.22). Once the roof pitches have been defined, the algorithm behind the hip roof rule generates a roof over a given 2D face using the face straight skeleton. As seen in Section 3.3.2, the straight skeleton of a polygon is obtained by shrinking the polygon edges in a self-parallel way, towards the interior of the polygon and at a constant speed. To generate a hip roof, the shrinking process is raised to the third dimension such that the edges slide away from the 2D polygon within a certain angle—this angle is actually the roof pitch [Held and Palfrader, 2017].

$$pitch_{rectangular\ faces} = \arctan \frac{height_{roof}}{\frac{baseline}{2}} \tag{3.13}$$

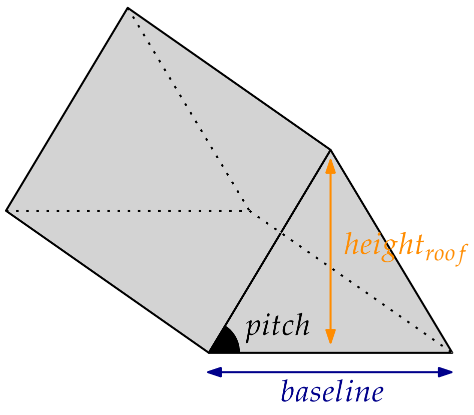


Figure 3.21: Computation of the roof pitch of the rectangular roof faces

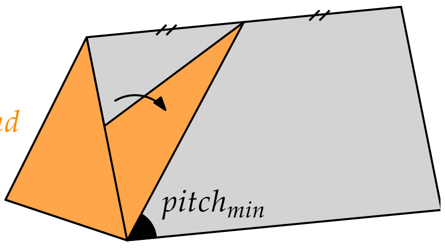


Figure 3.22: Minimum roof pitch allowed before creating self-intersection in the hip roof

A gable roof is a hip roof with vertical hip ends. Thus the algorithm to generate gable roofs is similar to the one that generates hip roofs except that the pitch of the hip ends is forced to 90°.

A crow-stepped gable roof is a type of roof characterised by a stepped vertical hip end on one side of the building (Figure 3.23). The rule to generate this type of roofs is based on the algorithm to generate hip and gable roofs. Simply put, constructing a crow-stepped gable roof consists in adding a vertical stepped face to a hip roof. The procedural modelling rule requires as input the following user-defined parameters (see Figure 3.24 for a description of what the parameters represent):

— The thickness of the stepped facade

— The roof height

— The number of ”steps” in the roof or the ”step” height. If the number of steps is given, the step height is computed by dividing the roof height by the number of steps.

— The roof pitch of the hip end opposite the stepped hip end

— The edge where to build the stepped facade

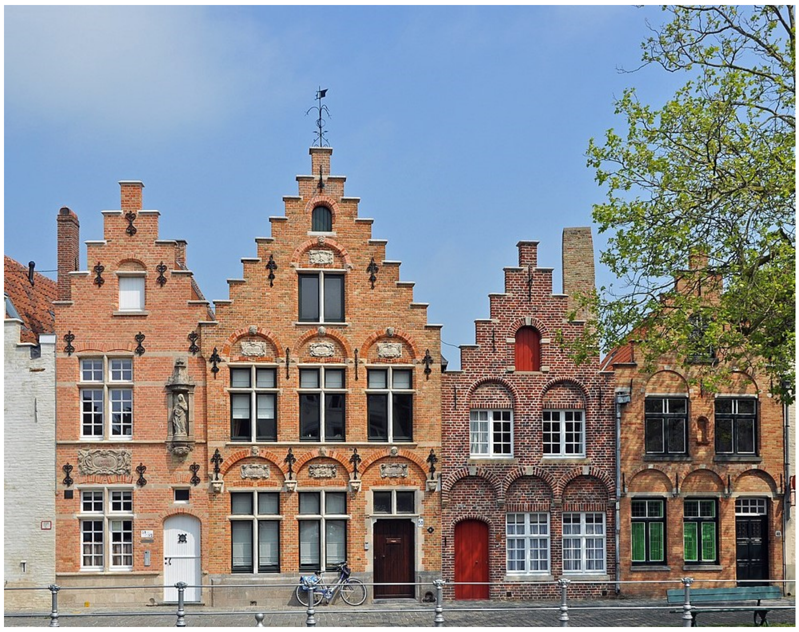


Figure 3.23: Example of crow-stepped gable roofs in Bruges (Belgium). Figure from Ryckaert [2012].

The algorithm to reconstruct a crow-stepped gable roof starts by generating a hip roof with a vertical hip end on the side of the building footprint edge specified by the user. The vertical hip end face and the building wall face connected to it are translated towards the interior of the building, from a distance

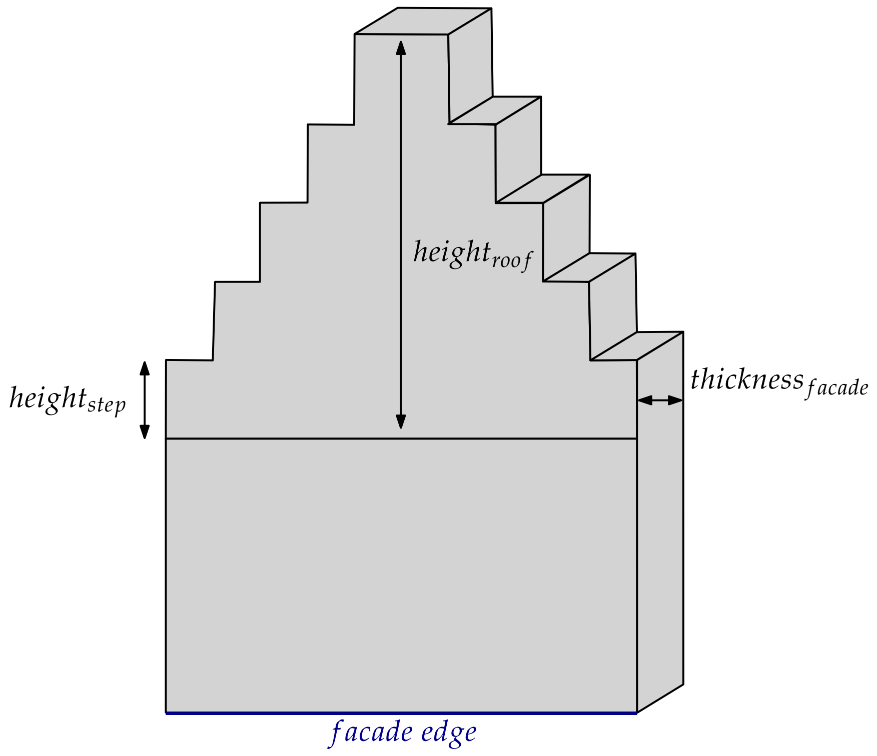


Figure 3.24: User-defined parameters needed to create the stepped facade

equal to the facade thickness parameter. This step is important to preserve the original extent of the building footprint. Not implementing it could result in overlapping buildings in the output 3D city model. Once the faces have been translated, a stepped hip end is created from the vertical hip end in an iterative way (Figure 3.25):

1. The algorithm finds the base vertices a and b and the top vertex c of the vertical hip end. A stepped face is initialised with vertices a and b.

2. a and b are offset in the z-direction to create a' and b', using an offset value equal to the step height.

3. The algorithm finds the intersection points i_1 and i_2 between the segment $[a'b']$ and the segments $[ca]$ and $[cb]$.

4. a', b' and i_1, i_2 are added to the stepped face.

5. Steps 3 to 5 are repeated until the remaining vertical distance between top vertex c and i_1, i_2 equals the step height.

6. The last two vertices d and e are created by offsetting the last i_1 and i_2 in the z-direction. They are added to the stepped face.

7. The vertical hip end is replaced by the stepped face.

Eventually, the newly created stepped face and the building wall face connected to it are duplicated and the duplicates are extruded horizontally from the original faces at a distance equal to the facade thickness. During the whole process, unused faces are removed such that only the visible faces are written to the final output 3D building. In that way, the final 3D building is a solid according the ISO 19107 definition.

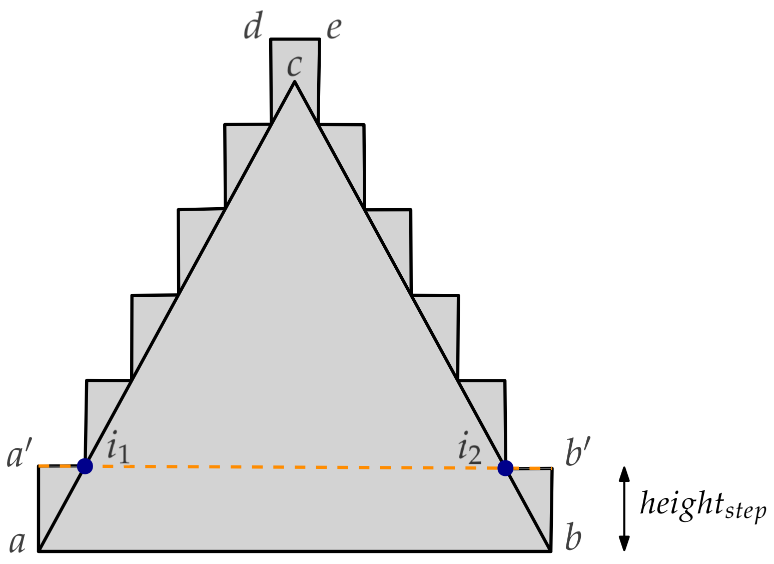


Figure 3.25: Process to generate a stepped vertical hip end

A mansard roof is a type of roof with several slanted roof faces and one flat roof face at the top. The mansard roof rule mainly relies on two user-defined parameters: the inset and the roof height. Even tough the choice of the inset value is left to the user, its value must always be smaller than half the shortest building footprint edge, in order to avoid self-intersection in the output roof (Figure 3.26). As shown in Figure 3.26, the inset value is used to generate four new corners from the original 2D face over which the mansard roof is built. Eventually, the four newly generated corners are offset in the z-direction using the user-defined roof height.

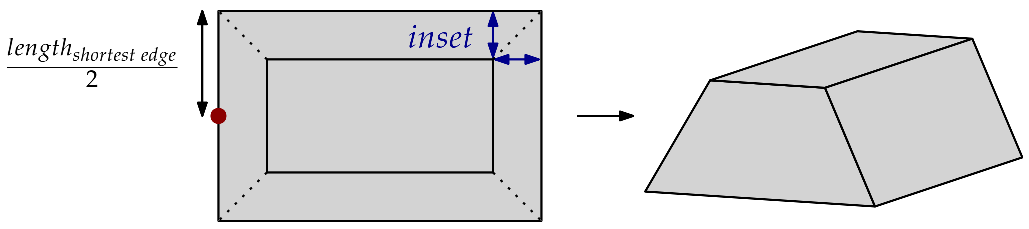


Figure 3.26: Process to generate a mansard roof

Identifying the street side of a building footprint

The roof-related rules have been improved so that the user can specify on which edge to build the hip/gable ends in the case of the hip/gable roofs and the stepped facade in the case of the crow-stepped gable roofs. In theory, any edge could be specified by the user but to achieve realistic results, two types of edges should be avoided:

- Edges shared by two building footprints
- The edge of the building footprint facing the interior courtyard or garden

Thus, the street edge of the building footprints has to be identified and provided as input to those roof rules. Historical street datasets could be used for that purpose but this means adding more difficulties due to the combination of these street data with the 2PM building footprints that come from other historical sources and that might already be affected by small discrepancies generated in the previous steps of the methodology workflow. Instead, the method used to identify the street edge is based on the following observations (Figure 3.27):

1. The building footprints tend to be aligned with each other on the street side, while on the courtyard/garden side they are not, due to variations in the building footprints depth.
2. The facade depth is always longer than the facade length.
3. Hip/gable/stepped hip ends tend to be located on the shortest edges of the building footprints and not on the long edges.

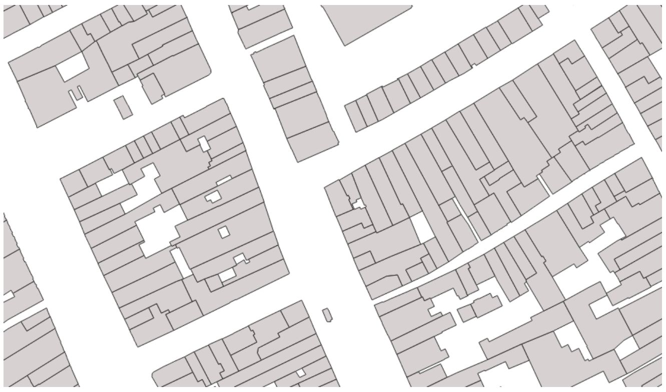
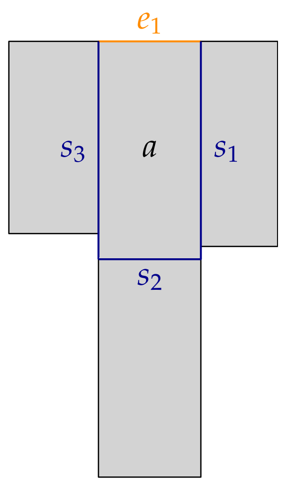


Figure 3.27: Example of building footprints from the Basisregistratie Adressen en Gebouwen (BAG) dataset of the Netherlands

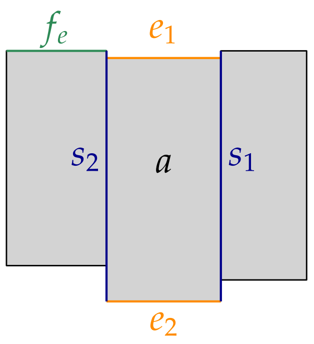
To identify the street edge of a given building footprint, the algorithm starts by finding the neighbours of the building footprint. The edges shared with those neighbours are then identified and marked so that they cannot be used as street edges. All the unshared edges are marked as potential edges where the street facade of the building could be built. From these potential edges, there are then four possible cases to identify which one is the street edge:

- **Case 1:** There is only one potential edge as all the other edges of the building footprint are shared. In this case, the street edge is simply this potential edge (Figure 3.28a).
- **Case 2:** The building footprint has a neighbour which was already reconstructed in 3D and thus already has a street facade. That information can then be used to generate the building street facade on the same side as its neighbour. The set of potential edges is reduced by keeping only the ones that are (nearly) parallel to the street edge of the neighbour. Eventually the closest one to the neighbour street edge is identified as the street edge of the building footprint (Figure 3.28b). Using the closest parallel potential edge ensures that the right street edge is still chosen even when observation 1 is not verified—i.e. neighbouring building footprints are not perfectly aligned on the street side. Note that case 2 requires keeping track of the street edges of all building footprints as they are reconstructed in 3D.
- **Case 3:** The building footprint does have neighbours but none of them was already reconstructed in 3D. In this case, the alignment with those neighbours is used to identify the street edge (observation 1). A voting procedure is applied; the unshared edges of the neighbours are iterated over and each unshared edge votes for the potential edge with which it is (nearly) aligned. In the end, the street edge is the potential edge with the highest number of votes (Figure 3.28c). If there are several potential edges with the highest number of votes, none is chosen and the algorithm goes to case 4.
- **Case 4:** If all the other cases failed, it means either that the building footprint does not have any neighbours and all its edges are thus potential edges, or that the voting procedure in case 4 failed. In any case, the street edge is chosen as the shortest edge of the building footprint (observations 2 and 3) (Figure 3.28d).

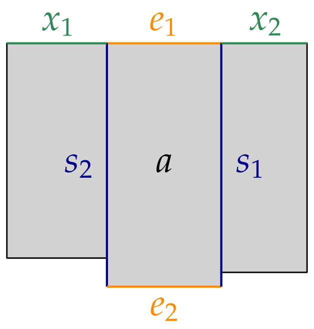
The four cases also make it possible to handle building footprints that are not touching each other but are separated by a small gap. For instance, on Figure 3.29, although edge e_1 and e_2 are better candidates for the street edge, edge e_3 would still be part of the set of potential edges. However, case 2, 3 and 4 would still tend to identify e_1 as the street edge and not e_3.



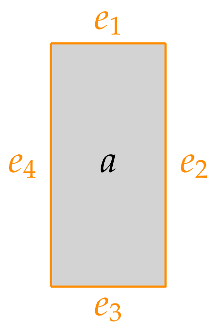
(a) Case 1: all the edges are shared except e_1, which is thus the street edge.



(b) Case 2: e_1 is the street edge as it is the closest parallel potential edge to f_e, the neighbour street edge.



(c) Case 3: e_1 is the street edge as two edges (x_1 and x_2) in the neighbours are aligned with it.



(d) Case 4: the street edge is the shortest edge, thus either e_1 or e_3.

Figure 3.28: Identifying the street edge of a given building footprint a. The potential street edges are marked in orange as e_i and the shared edges are in dark blue as s_i.

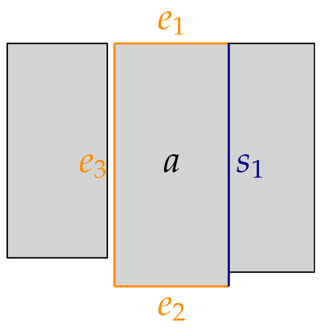


Figure 3.29: Identifying the street edge when there is a gap between two adjacent building footprints. The potential street edges are marked in orange as e_i and the shared edges are in dark blue as s_i.

Generating the final set of rules

The final set of rules used to reconstruct the LoD2 buildings is made using three main rules:

- **Rule 1**: The building footprint is extruded at a certain height (assigned roof height).
- **Rule 2**: The resulting 3D shape is decomposed into four parts (top, front and sides faces).

- **Rule 3**: One of the roof rules is used to generate a roof (hip, gable, crow-stepped gable, mansard) above the top face of the 3D shape, or the roof is kept flat. The type of roof is randomly chosen using different probabilities depending on the date of the historical map. Indeed, different roof types are characteristic of different epochs. For instance, crow-stepped gable roofs were common on traditional Dutch houses from the 12th century until the beginning of the 20th century [Mesqui, 1998], while flat roofs started to be more widely used in Northern Europe from the 19th century [Urbanik and Tomaszewicz, 2014]. By visualising the city of interest in Google Maps alongside with a building footprints dataset including the building year of construction, it is also possible to gain insights into which types of roofs were common at which epochs. The exact roof types probabilities used in this study are given in Appendix B (Table B.1).

From these three rules, the final historical 3D city model is reconstructed. The algorithm to reconstruct the LoD2 buildings processes each input 2PM building footprint at a time and works by (1) identifying the street edge of the building footprint, (2) randomly choosing a roof type, (3) choosing values for all user-defined parameters, (4) writing the rules set to a rule file and (5) executing it.

3.4.3 Valid CityJSON generation

The final steps to reconstruct the output historical 3D city model consist in (i) adding semantics to the surfaces of the LoD2 buildings generated with 3D procedural modelling, (ii) generating an output CityJSON file containing both the LoD2 buildings corresponding to the 2PM building footprints and the LoD2 buildings corresponding to the aligned building footprints, and (iii) assessing the quality of the output historical 3D city model. The following subsections describe these steps.

Adding semantics

Without semantic information, the reconstructed LoD2 buildings are only a set of geometric primitives made of vertices and triangles for a computer [Arroyo Ohori et al., 2020]. As computer-based pragmatic applications of current 3D city models require 3D geometries enriched with logical information, this also holds for historical 3D city models [Hadjiprocopis et al., 2014]. For this reason, semantics are assigned to the different faces of the LoD2 buildings based on the method and tolerance values of Biljecki et al. [2016b]. The faces of a given LoD2 building are classified between three different types of surfaces using the face normal (Figure 3.30):

- Ground surface: the face normal points towards the negative z, with an angle varying from -5° to 5° from the vertical direction.
- Wall surface: the face normal forms an angle between -5° and 5° with the horizontal vector pointing in the same direction.
- Roof surface: the face is neither a ground surface nor a wall surface and it has at least one of its vertices at a height equal or higher than the building roof height.

Generating a valid CityJSON file

Once semantics have been added, the LoD2 buildings of the 2PM building footprints are stored as Solids with semantics surfaces into a CityJSON file. A valid CityJSON Solid is a solid following the ISO19107 standards with the additional characteristics that all its 2-primitives are planar and its 1-primitives linear. To be valid solids, the LoD2 buildings must verify all the assertions described in Table 3.4.

The final historical 3D city model is obtained by merging the file containing the LoD2 buildings of the 2PM building footprints with the file containing the LoD2 buildings of the aligned building footprints. All buildings in the output historical 3D city model are LoD2 but the 3D procedural modelling process always generates LoD2.0 buildings while the aligned buildings coming from a current 3D city model—e.g. the 3D BAG of the Netherlands—could be LoD2.1 or LoD2.2 (Figure 3.31). Indeed, the aligned buildings might have more complex building footprints and additional building installations such as

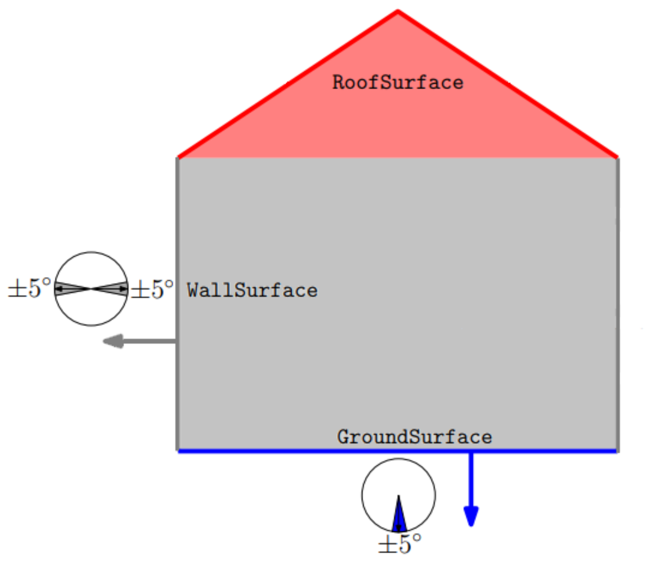


Figure 3.30: Angles and tolerances for semantics assignment. Figure modified from Biljecki et al. [2016b].

1.	The solid is closed.
2.	Each shell of the solid is simple.
3.	The boundaries of shells can intersect with each other but the intersection can only contain primitives of dimensionality 0 (vertices) and 1 (edges).
4.	Each shell of the solid is a 2-manifold and it has no dangling pieces.
5.	The interior of the solid forms a connected point-set.
6.	All the lower-dimensionality (0 and 1) primitives forming the solid are valid.
7.	The normal of the polygons in the solid points outwards in a right-hand system. Interior rings in these polygons are oriented clockwise.

Table 3.4: The assertions a solid must fulfill to be valid according to ISO19107 standards [Arroyo Ohori et al., 2020].

dormers or chimneys in comparison to the 2PM buildings, which always have convex building footprints. In this case, the final historical 3D city model is a multi-LoD model. Note that this applies when the complete version of the methodology is implemented—i.e. using the maps alignment process—as the shortened version produces an historical 3D city model only made of 2PM buildings.

Results assessment

Eventually, the quality of the output historical 3D city model is assessed. First, a visual assessment is conducted to check the physical appearance of the LoD2 buildings reconstructed with 3D procedural modelling. In this way, it can be verified if the LoD2 buildings were generated as expected. For instance,

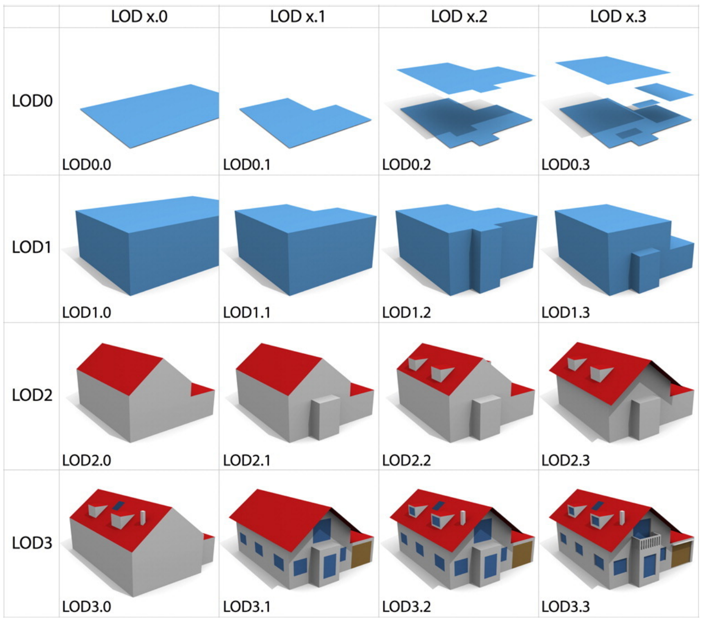


Figure 3.31: LoD specifications of 3D building models. Figure from Biljecki et al. [2016c].

it can be checked whether the street facade of the buildings was indeed created on the street side or if some street facades were generated on shared building walls (Section 3.4.2). In addition to the visual assessment, a second assessment is implemented to check the validity of the 3D geometries and of the CityJSON file against its schema. The validity of the geometries is verified using the val3dity software. In this way, it checks if the LoD2 buildings are valid CityJSON Solids. As for the CityJSON file, it is validated against its schema with the cjio validator. This validator also checks the internal consistency of the file. Among others, it checks for duplicate and orphan vertices, empty geometries and incoherent vertex indices.

4 Datasets and implementation

In this chapter, the datasets that were used to implement and test the methodology are described (Section 4.1). In addition, the programming specifics such as the used softwares and tools are detailed (Section 4.2).

4.1 Datasets

This thesis focuses on reconstructing historical 3D city models of Delft at different time periods. In addition, the methodology is also tested for the city of Brussels to assess how well the methodology performs for other study areas. The methodology relies on three types of datasets: (1) historical maps which are the primary input of the methodology, (2) 2D and 3D buildings datasets which, if available, are used along with the maps alignment process (Section 3.3.1), and (3) a 3D point cloud dataset which is needed to assign reference ground heights to the historical building footprints (Section 3.4.1). These datasets are described in the following subsections for both Delft and Brussels.

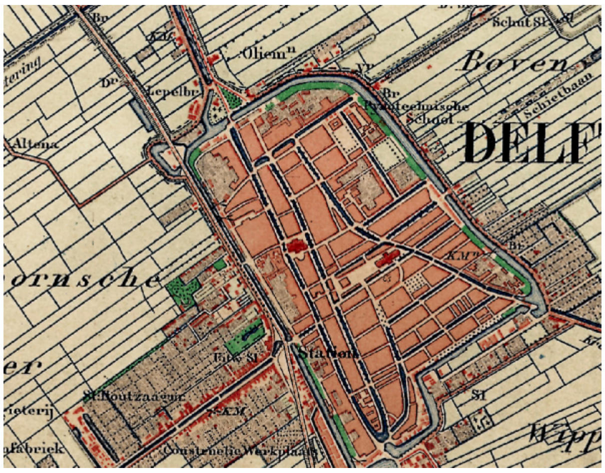
4.1.1 Historical maps

Delft

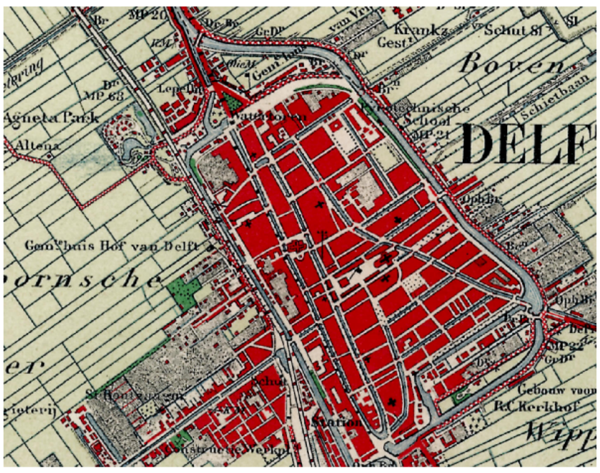
This research uses historical maps of Delft coming from the website `https://www.topotijdreis.nl/` which displays topographic maps of the Netherlands acquired by the Dutch Kadaster since 1815. These maps are already georeferenced in the national coordinate reference system (CRS) of the Netherlands (Amersfoort/RD New). Two maps are used from this map collection: a map dating from 1880 and a map dating from 1915. In addition, the methodology is also implemented with two historical maps of Delft dating from 1961 and 1982 and coming from the map collection of the TU Delft Library. They are available at `https://heritage.tudelft.nl/en/collections/lib-kaartenkamer` without any georeference information. Figure 4.1 shows the four input historical maps and Table 4.1 gives a summary of the maps properties.

Historical map	**Provider**	**Spatial resolution**	**Georeference information**	**Availability**
Delft 1880	Topotijdreis	2.58 m	EPSG: 28992	Freely available
Delft 1915	Topotijdreis	2.58 m	EPSG: 28992	Freely available
Delft 1961	TU Delft Library	1.47 m	Not georeferenced	Freely available
Delft 1982	TU Delft Library	1.58 m	Not georeferenced	Freely available
Brussels 1700	Gallica	1.80 m	Not georeferenced	Freely available
Brussels 1890	Gallica	1.38 m	Not georeferenced	Freely available
Brussels 1924	Gallica	1.80 m	Not georeferenced	Freely available

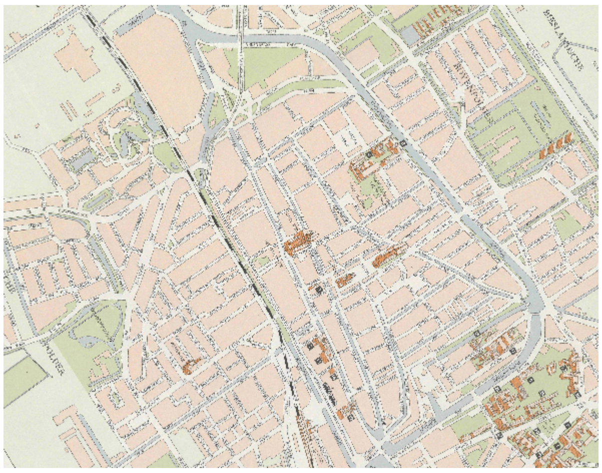
Table 4.1: Historical maps information. Note that for comparison purposes, the spatial resolution was measured in the same CRS (EPSG:3857) for all maps.



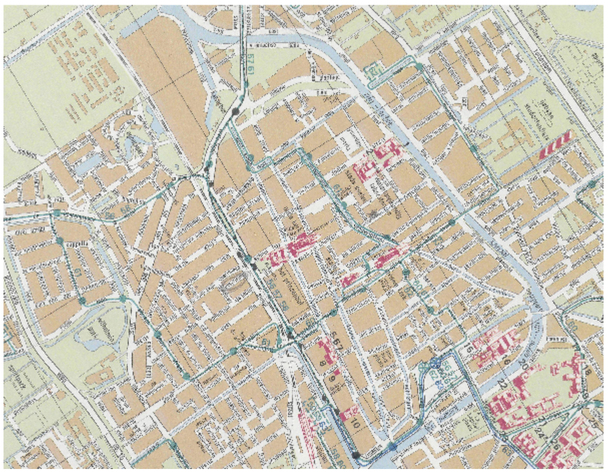
(a) Delft 1880



(b) Delft 1915

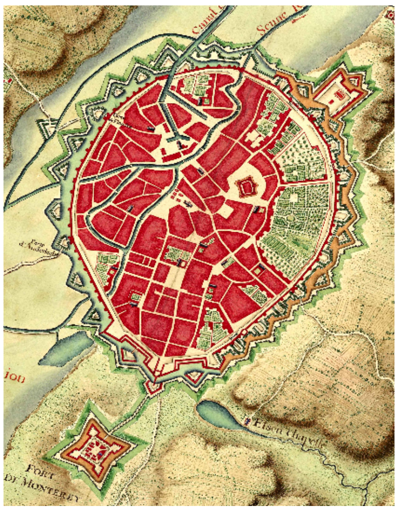


(c) Delft 1961

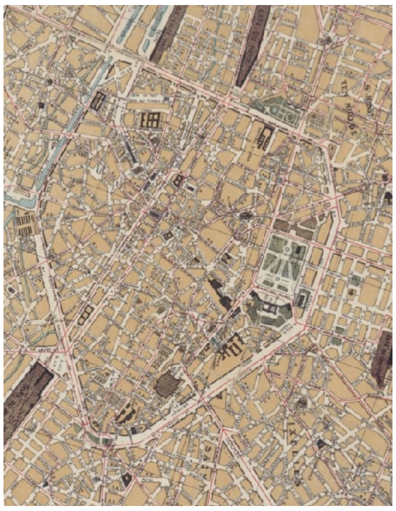


(d) Delft 1982

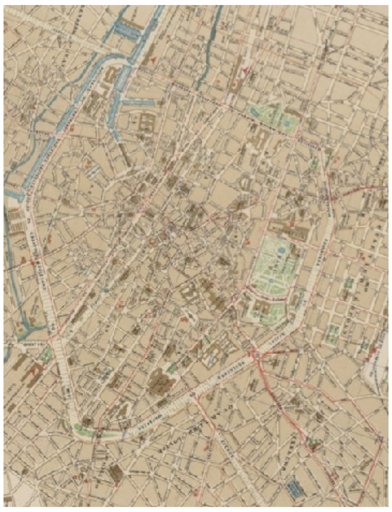
Figure 4.1: Historical maps of Delft



(a) Brussels 1700



(b) Brussels 1890



(c) Brussels 1924

Figure 4.2: Historical maps of Brussels

Brussels

The historical maps of Brussels that were used to test the methodology come from Gallica which is the digital library of the Bibliothèque nationale de France (BnF)—this latter translating to the national library of France. Gallica is accessible through `https://gallica.bnf.fr/`. Three historical maps of Brussels coming from different collections in Gallica are used to test the methodology: a map from 1700, a map from 1890 and a map from 1924. These are shown in Figure 4.2 and their properties are displayed in Table 4.1.

4.1.2 2D and 3D buildings datasets

Delft

In the case of Delft, 2D and 3D buildings datasets with the required attributes are available such that the complete version of the methodology can be implemented—i.e. including the maps alignment process (Chapter 3). These required attributes are: the building year of construction, reference roof height attributes and an attribute that links the 2D and the 3D buildings datasets. The building footprints dataset used comes from the Basisregistratie Adressen en Gebouwen (BAG). This register contains location information and other attributes about all buildings in the Netherlands. It results from the collaboration of the Dutch municipalities and it is kept up to date by the Kadaster. The dataset is available freely from the open geoportal of the Netherlands, PDOK, which is accessible from `https://www.pdok.nl/`. As for the 3D buildings dataset, the 3D BAG made by the 3D geoinformation group of TU Delft is used. The 3D BAG consists of 3D building models for the BAG building footprints, available in three level of details: LoD1.2, LoD1.3 and LoD2.2. In addition, the BAG building footprints are also available enriched with reference height attributes. All these datasets can be downloaded as open source from `https://3dbag.nl/`. The relevant BAG and 3D BAG attributes needed to implement the complete methodology are described in Table 4.2 with the section of the methodology for which they are used.

Attribute	**Description**	**Use**
Identificatie code	Unique building ID	Links BAG to 3D BAG
Bouwjaar	Authentic building year of construction	1:n alignment
$roof_{0.25}$ $roof_{0.50}$ $roof_{0.75}$ $roof_{0.90}$ $roof_{0.95}$ $roof_{0.99}$	Percentile roof heights (Figure 4.3)	Height inference

Table 4.2: Relevant BAG and 3D BAG attributes

Brussels

Although 2D and 3D buildings datasets are available for the city of Brussels, the year of construction of the buildings is not provided with their geometry. As the maps alignment process relies on that attribute, it cannot be implemented and only the shortened version of the methodology can be applied for reconstructing historical 3D city models of Brussels (Chapter 3). The current building footprints and their attributes not being used in that methodology version, the roof height attributes must be inferred from another source of information. In this case, the roof height is randomly generated based on regulated height values established for Brussels in the 19th century [de Pange, 2004]. These values are shown in Table 4.3.

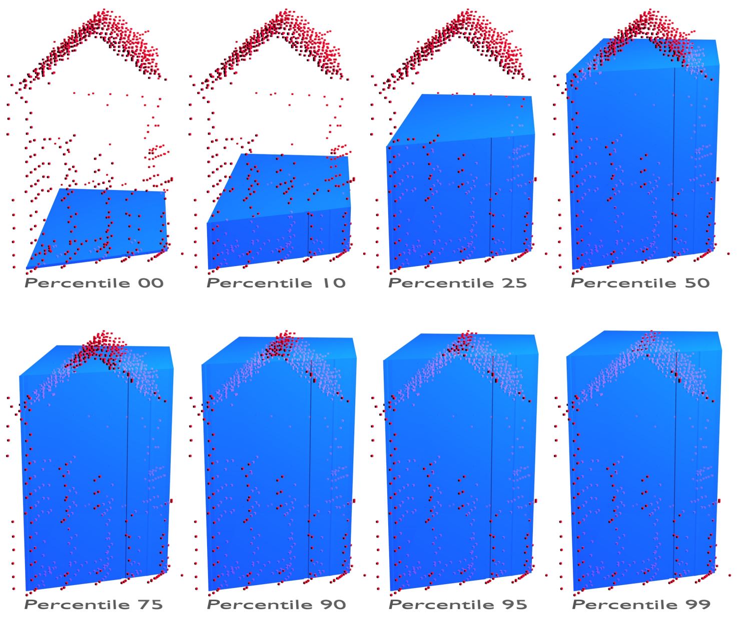


Figure 4.3: Percentile roof heights (blue) computed from a set of building roof points (red). Figure from Dukai [2018].

Building level	Registered height
Facade	Max. 21 m
Ground floor	Min. 3.5 m
First floor	Min. 3.2 m
Second+ floors	Min. 3 m
Ceiling	Min. 2.6 m

Table 4.3: Registered heights for different building levels in Brussels [de Pange, 2004]

4.1.3 3D point cloud dataset

Delft

The Actueel Hoogtebestand Nederland 3 (AHN3) is used as 3D point cloud dataset for Delft. This dataset contains accurate elevation data acquired using LiDAR technology. The points in the point cloud are classified into different categories: ground, buildings, water and others. The dataset is available as a LAZ file from the Dutch geoportal, PDOK, at `https://downloads.pdok.nl/ahn3-downloadpage/`. Before being directly used in the methodology, the point cloud is preprocessed in order, first, to thin it and second, to extract the ground points. Keeping only the ground points creates holes in the point cloud at the location of the buildings, but the method implemented for inferring the ground height was made to overcome that issue (Section 3.4.1).

Brussels

No 3D point cloud dataset is available for Brussels. Instead, a DEM is used and processed in a GIS to generate a 3D point cloud dataset. DEMs of the city of Brussels can be downloaded from the Brussels open data portal available at `https://datastore.brussels`.

4.2 Programming specifics

The main programming language used to implement the methodology is Python. The methodology also relies on different tools and programming softwares. These are described in the following subsections.

4.2.1 GRASS GIS

GRASS GIS is an open source geographic information system (GIS) software designed for geodata processing and analysis, image processing and visualisation and cartography among other various applications. The software is written in Python and it can be extended via addon modules. In this thesis, the historical maps digitalisation method of Gobbi et al. [2019] in Section 3.2.1 is implemented with GRASS GIS modules. The GRASS Python Scripting Library allows for using them outside the GRASS GIS interface in a Python script for automation purposes. Table 4.4 shows the main GRASS GIS modules used in this study and the methodology step in which they are used.

GRASS GIS module	Use
i.segment.uspo	Segmentation with hyperparameters tuning
i.segment.stats	Computation of shape and color properties for the segments
v.class.mlR	OBIA classification
r.object.thickness	Estimation of the text segments thickness
r.fill.category	Removal of textual features

Table 4.4: Main GRASS GIS modules used in the building plots extraction workflow

4.2.2 Blender

Blender is an open source 3D computer graphics software offering advanced functionalities for 3D modelling, rendering and animation among many other applications. Additional user-defined functionalities can be created via Blender addons using Python scripting. The software can be used either directly from the user interface or outside the interface in external Python scripts. In this research, Blender is used for reconstructing the historical 3D city model from the building footprints (Section 3.4.2). Several addons are used for that purpose:

- **BlenderGIS**: It allows to handle geospatial data in Blender by supporting many common GIS functionalities. It is available at `https://github.com/domlysz/BlenderGIS`. It is mainly used for importing the building footprints datasets in Blender and handling the CRS information.
- **Computer Generated Architecture for Blender (BCGA)**: It is an implementation of the CGA shape grammar for 3D procedural modelling in Blender. It is available at `https://github.com/vvoovv/bcga`. In this thesis, the reconstruction of the historical 3D city models heavily relies on it (Section 3.4.2).
- **Up3date**: This addon allows to import, export and modify 3D city models encoded in the CityJSON format. It is available at `https://github.com/cityjson/Up3date`. It is used for creating valid CityJSON files for storing the output historical 3D city models (Section 3.4.3).

4.2.3 Open source tools

In addition to these two softwares, other open source tools are also used in this thesis:

- The building footprints generalisation method of Commandeur [2012];

- The maps alignment method of Sun et al. [2020], available at https://figshare.com/articles/dataset/Aligning_geographic_entities_from_historical_maps_for_building_knowledge_graphs/13158098;
- The open source map vectoriser of Arteaga [2013], available at https://github.com/nypl-spacetime/map-vectoriser;
- cjio, the CityJSON files validator available at https://github.com/cityjson/cjio, and,
- val3dity, the CityJSON geometries validator available at https://val3dity.readthedocs.io/en/latest/.

5 Results, experiments and analysis

This chapter focuses on the description and analysis of the results obtained by implementing the methodology workflow. It is organised as the methodology chapter (Chapter 3) and presents first the results obtained with historical maps of Delft. In that way, the georeferencing of the historical maps is described (Section 5.1). Then, different building plots extraction methods are compared and the results of the final building plots extraction workflow are presented (Section 5.2). The next section details the results of the building plots decomposition into individual building footprints (Section 5.3). Lastly, the output historical 3D city models of Delft are presented and assessed (Section 5.4). In addition to the Delft study case, the methodology was also implemented with historical maps of Brussels to test its suitability in other study areas. The results of this implementation are presented in Section 5.5.

5.1 Data preparation

5.1.1 Georeferencing

Three georeferencing transformations were investigated to georeference the historical maps:

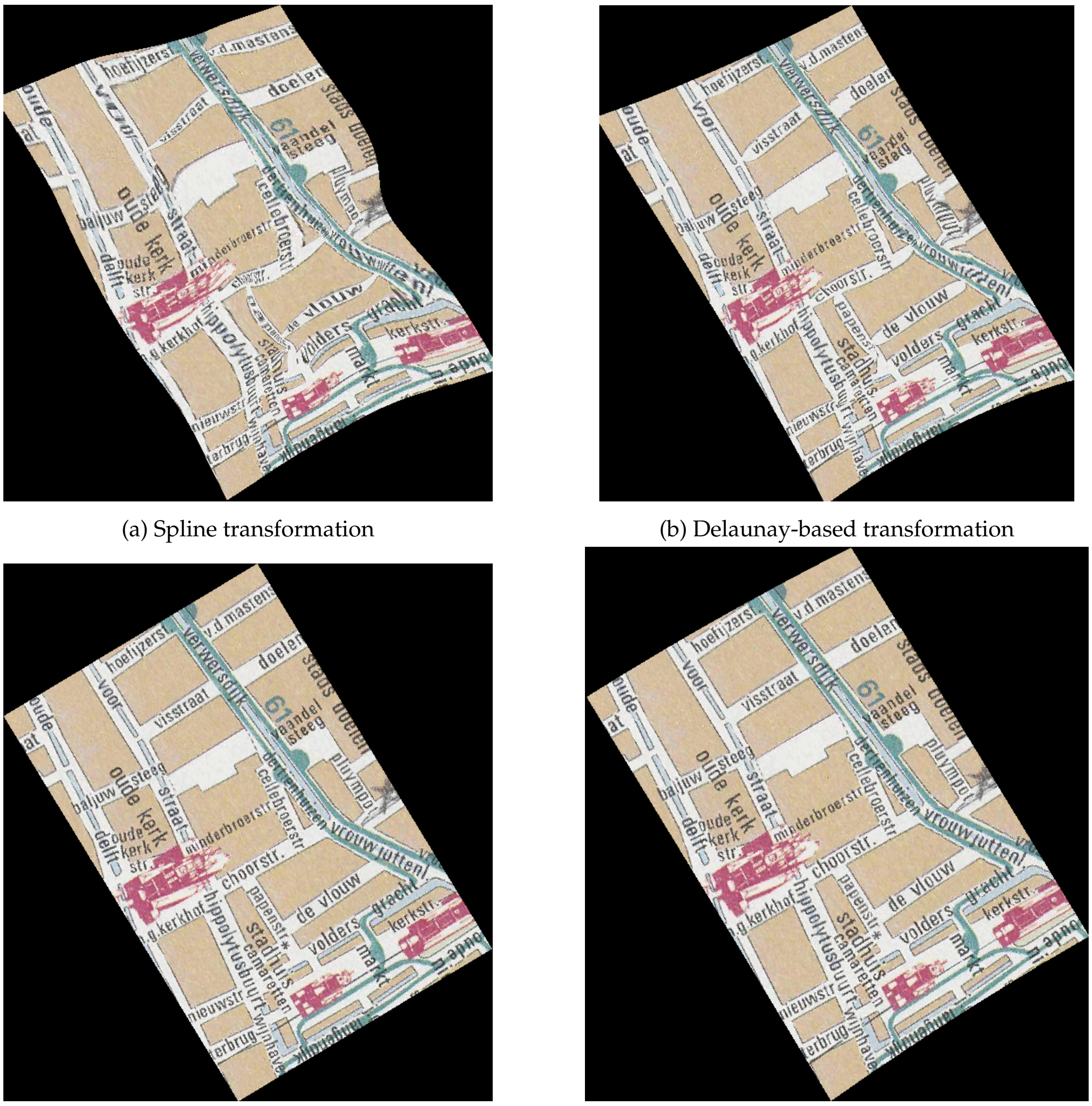
- **Spline**: The spline transformation is a local transformation method that relies on splitting the input map into regions and fitting for each region a different polynomial function, so that the control points are not shifted from their true coordinates and the local accuracy is optimised. Spline and other similar local transformations are usually used and recommended for georeferencing damaged or slightly inaccurate maps such as historical maps [Herrault et al., 2013a; Király et al., 2008].
- **Polynomial transformations**: This category of transformations works by deriving a general formula from the set of control points, using least squares fitting. These transformations preserve global accuracy at the expense of local accuracy. Depending on the order of the polynomial function that is used, straight lines are either preserved (first-order) or curvatures can be introduced (second-order).
- **Delaunay-based method**: With this method, the input map is first georeferenced with a global polynomial transformation. Then a Delaunay triangulation (DT) is built from the set of control points and they are locally adjusted to their true coordinates by using a Delaunay-based interpolation. In this way, this method optimises both the global and the local accuracy.

The RMSE values obtained with the historical map of Delft from 1982 are shown in Table 5.1. The spline and Delaunay-based methods yield low RMSE values (almost 0) as the control points are mapped at their true coordinates. Therefore, one could think that these methods are the most suitable for georeferencing this historical map. However, with these methods, the geographic features of the output georeferenced map are highly distorted and stretched, as shown in Figure 5.1a and Figure 5.1b. These distortions are due to the geometric inaccuracies that affect the historical map. Note that the results shown here were obtained with a smaller tile extracted from the input historical map, so as to highlight the effect of local geometric inaccuracies on the georeferencing results without the influence of inaccuracies affecting other areas of the input historical map. Some of these local inaccuracies can easily be noticed by comparing the historical map with a current georeferenced map of the same study area. For instance, Figure 5.2 shows that the ratio of the building plots width and the streets width is not the same on the input historical map and on the current georeferenced map.

As for the polynomial transformations, the georeferenced maps are not highly distorted and stretched but the RMSE values are larger due to the geometric inaccuracies of the historical map, which means that

Transformation	RMSE (meters)
Spline	0.00
Delaunay-based method	1.66
1st-order polynomial	8.39
2nd-order polynomial	8.27

Table 5.1: RMSE values for different georeferencing transformations



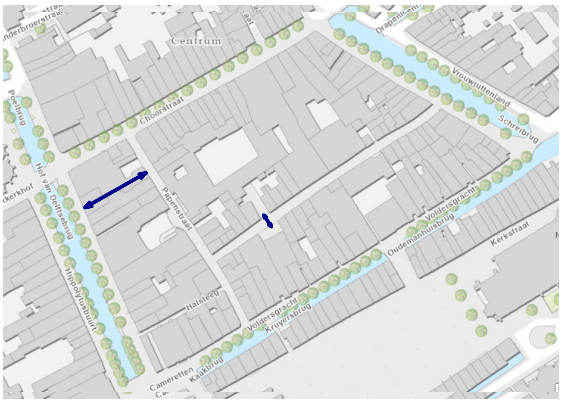
(a) Spline transformation

(b) Delaunay-based transformation

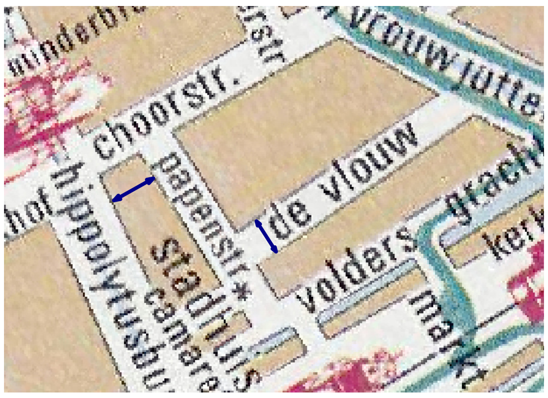
(c) 1st-order polynomial

(d) 2nd-order polynomial

Figure 5.1: Delft historical map from 1982 georeferenced using different types of transformations



(a) Current map of Delft



(b) Historical map of Delft from 1982

Figure 5.2: Comparison of the relative proportions of the width of geographic features between a current map of Delft and an historical map from 1982. The blue arrows represent the width of the geographic features. In a, the width of the building plot is way larger than the street width, while in b, the building plot and the street widths are almost equal.

the control points are slightly displaced from their true coordinates. As a result, the geographic features are sometimes slightly moved from their true location. Considering pros and cons of all methods, local transformations were not used and it was chosen to use instead a first-order polynomial transformation for georeferencing the historical maps. This is further discussed in Section 6.1.2.

5.2 Building plots extraction

In this section, the results of the building plots extraction workflow are described. First, the comparison between existing methods that led to the development of the final building plots extraction workflow is presented. Then, the results obtained with the final workflow are described and assessed.

5.2.1 Comparison between existing methods

As suggested by Liu et al. [2019] and Herrault et al. [2013b], it is difficult to find a unique building footprints extraction methodology that works for any collection of historical maps. Instead, a methodology might yield very good results for a specific type of historical maps and produce bad results when applied on any other map. Therefore, in this thesis, several experiments have been conducted to further find the most relevant methodology for the input historical maps used in this study. In this way, three types of methods have been investigated to create the final building plots extraction workflow (Section 3.2): existing building footprints extraction methods, historical maps digitalisation methods and building footprints generalisation methods. The results of these experiments are described in the following subsections and Table 5.2 contains a summary of the methodology of all the methods used and implemented.

Implementation	Type of method	Overview of the methodology
Arteaga [2013]	Building footprints extraction method	This methodology, ***used*** as is, relies on a calibration step which consists in enumerating all the colors present in the input map and the class they represent. A thresholding step is taken in GIMP where the contrasts are increased, and the brightness decreased. The resulting image is vectorised and each polygon is generalised using an α-shape operator. Each polygon is then cropped with the input historical map and it is kept if the average color of the cropped image matches with the building color.
Drolias and Tziokas [2020]	Building footprints extraction method	This method was ***implemented*** in QGIS. It starts by manually reclassifying the input map to generate a binary raster. Then, small isolated pixels are removed and the raster patches representing building footprints are vectorised. Holes are removed and the building footprints are generalised using a DP algorithm.
de Boer [2010]	Building footprints extraction method	The method was ***implemented*** in QGIS, although developed in ArcGIS by the authors. It starts by recreating a new raster by manually selecting all pixels of the building color on the historical map. All building pixels of the new raster are then vectorised into point features and a Delaunay triangulation (DT) is built. Using an area threshold, triangles with a large area are removed and adjacent polygons are merged into one connected area. Resulting building footprints are smoothed if necessary.
Gobbi et al. [2019]	Historical maps digitalisation method using a supervised OBIA classification.	This method was ***implemented*** using GRASS GIS modules. An OBIA segmentation is performed to create segments of similar shape and color. Training data are then used for classifying the segments. In order to remove textual information from the classified map, a low-pass filter is applied on the pixels representing textual features and their value is replaced by the mode value in the pixel surrounding.
Henderson et al. [2009]	Historical maps digitalisation method using unsupervised classification algorithms.	Unsupervised classification algorithms can be ***used*** for automatically segmenting raster maps, such as the expectation maximisation algorithm and the K-means clustering algorithm.
Herrault et al. [2013b]	Historical maps digitalisation method using unsupervised classification algorithms.	This method makes use of the K-means clustering algorithm to digitalise an input raster map. The main difference with the method of Henderson et al. [2009] is that some preprocessing step based on dilation and smoothing operators is ***implemented*** beforehand to remove textual features, lines and symbols from the input historical map.
scikit-learn supervised algorithms	Historical maps digitalisation method using supervised classification algorithms.	Supervised classification algorithms such as random forest can be ***used*** to digitalise an input historical map using a training dataset.
Commandeur [2012]	Building footprints generalisation method.	The generalisation algorithm, ***used*** as is, aims at merging the line segments of the boundary of a building footprint which are nearly parallel. The methodology relies on merging line partitions based on a distance and an angle threshold.
Douglas and Peucker [1973]	Building footprints generalisation method.	The DP algorithm, ***used*** as is, generalises a building footprint by keeping only a subset of its vertices. The process iteratively keeps points within a tolerance distance from the polyline being generalised.
Shapely simplification algorithm	Building footprints generalisation method.	The algorithm ***used*** is similar to the DP algorithm as it reduces the number of vertices of the input building footprint using a tolerance distance. The main difference between the two algorithms is that this one preserves the topology of the input building footprint.
CGAL simplification algorithm	Building footprints generalisation method.	Such as the DP algorithm, the algorithm ***used*** is a line simplification algorithm with the difference that it preserves topology. The number of vertices in the building footprint is reduced either by using a tolerance distance (euclidean distance mode), or keeping a defined number of vertices from the total number of vertices (count mode) or keeping a percentage of the total number of vertices (ratio mode).

Table 5.2: Overview of the different methods used, implemented and compared

Existing building footprints extraction methods

I used, implemented and compared several semi-automatic and automatic building footprints extraction methods described in Chapter 2. From these experiments, I determined the pros and cons of the different methods and the applications and types of historical maps for which they are relevant. These are described in the following paragraphs.

Arteaga [2013] designed a method which relies on open source tools and which is automatic; the user runs the code directly from the command prompt and the only user intervention needed is to fill in the average color of each class of the map in a text file. Their α-shape simplification method yields good generalisation results. Concave shapes are preserved depending on the α parameter. Depending on the irregularity of the contour of the original building footprints, it could be used as a first generalisation step. However, one downside of this method is that it relies on a thresholding step in GIMP which requires the user to play with the brightness and contrast values to find the best ones for their input map. Even when using the best values, this method generates lots of misclassified features and overlapping building footprints if the input historical map is too complex (Figure 5.3). Another con of this method is that building footprints are sometimes split into several parts because of textual labels, lines and symbols on the historical map. Considering all pros and cons of this method, it can be used for historical cadastral plans or simple historical maps without too many classes, provided that the maps are of good quality. This implies that the input maps are not affected by the aliasing effect so that the different geographic features have uniform colors on the map. This is important as the classification process labels a patch of pixels based on its average color; if there is too much color variation in the patch, it does not perform well.

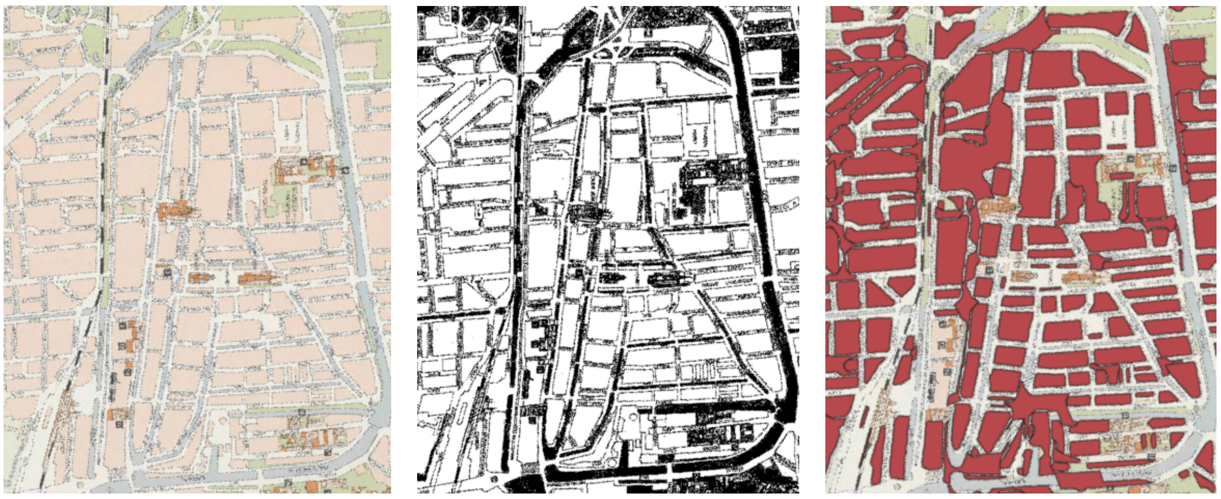
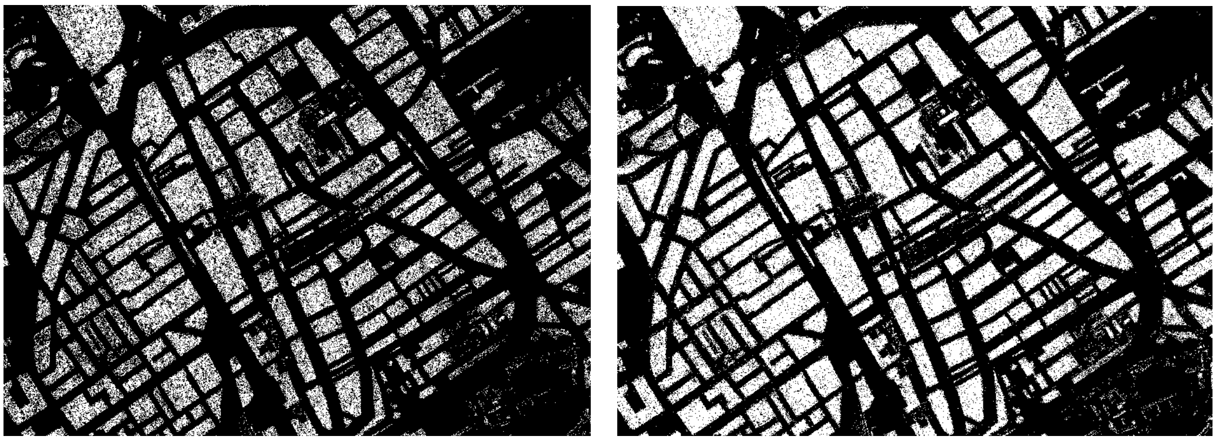

(a) Input historical map (b) Binary raster map (c) Output building plots

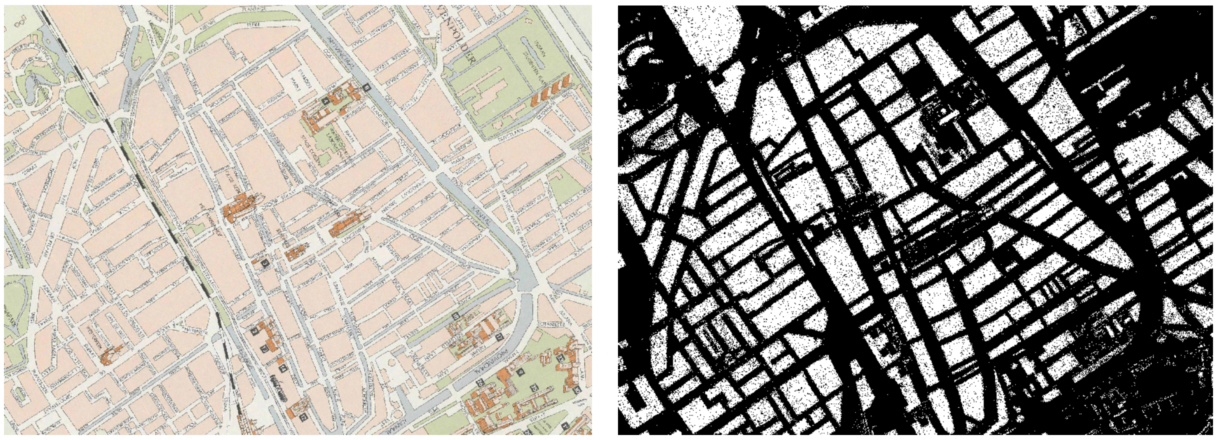
Figure 5.3: Building footprints extraction method of Arteaga [2013] applied with the historical map of Delft 1961

Drolias and Tziokas [2020] developed a methodology relying on QGIS, which is available as open source. That method is quite straightforward and easy to implement and can be made semi-automatic using QGIS Model Builder. However, the reclassification step requires lots of trial and error work. Slight color variations inside and between building footprints make it impossible to capture all building pixels without including pixels of other classes (Figure 5.4). Besides, the DP algorithm is not well suited for building footprints generalisation and it does not preserve concave shapes. Building footprints are sometimes split into several parts because of textual labels, lines and symbols on the historical map. These cons make the method usable for historical maps where buildings are depicted in a uniform white or black; this eases the thresholding step where the pixel value is the same in all RGB bands—255

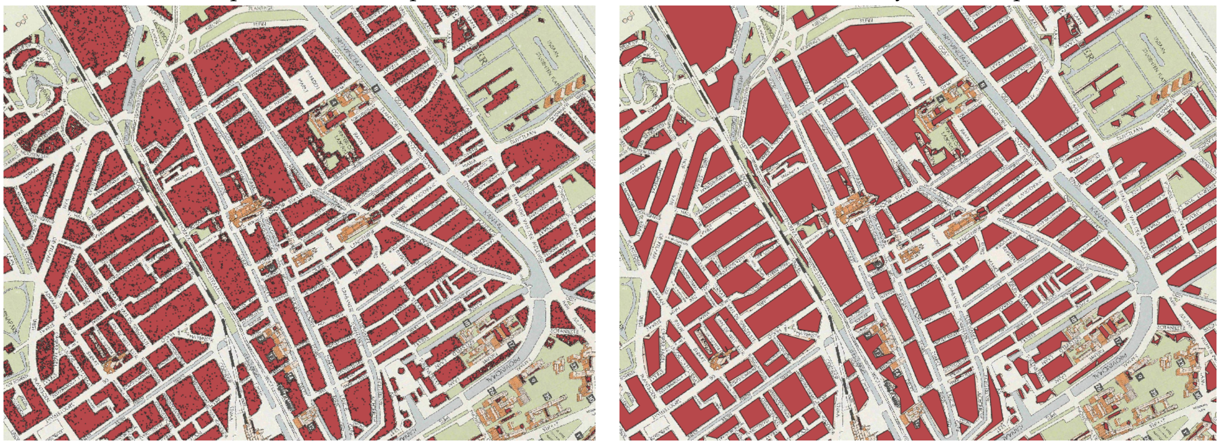


(a) (b)

Figure 5.4: Effect of enlarging the thresholds in histogram thresholding. By enlarging the thresholds from a to b, non-building pixels are included into the building category (white) while some building pixels are still classified into the non-building category (black).



(a) Input historical map (b) Binary raster map



(c) Vectorised building plots (d) Cleaned and generalised building plots

Figure 5.5: Building footprints extraction method of Drolias and Tziokas [2020] applied with the historical map of Delft 1961

or 0 depending if it is white or black. In addition, the misclassifications due to the thresholding step can be greatly reduced if the input maps are not affected by the aliasing effect. To conclude on the use cases of this method, it is actually quite handy as it is easily and quickly implementable. Thus, if the user wants a fast implementation where the geometric accuracy of the results matters less, I recommend using this method (Figure 5.5).

de Boer [2010] designed a methodology relying on ArcGIS, which is not available as open source. Nevertheless, it can be easily implemented in QGIS with equivalent tools and can thus be made semi-automatic using QGIS Model Builder. The DT step can provide good generalisation results, provided that the contour of the building footprints was properly recovered from the input historical map. Concave shapes are preserved. Despite these advantages, this method also has lots of cons. First, selecting pixels of the building color requires lots of trial and error work. Slight color variations inside and between building footprints make it impossible to capture all building pixels without including pixels of other classes (Figure 5.4). As all other methods investigated, the building footprints are sometimes split into several parts because of textual labels, lines and symbols on the historical map. Lastly, the main downside of this method is that it is computationally intensive to implement in QGIS for input maps with high spatial resolution as this requires generating a point in all building pixels and triangulating them. It is thus very slow and prone to computer bugs and it may require downsampling the input historical map at the expense of the quality of the results. Therefore, I would not recommend using this method as it can be used in similar use cases than the method of Drolias and Tziokas [2020] but provides lower quality results and is too heavy to implement in QGIS if the spatial resolution of the map is high.

Historical maps digitalisation methods

Almost all the existing building footprints extraction methods previously described and implemented make use of histogram thresholding for digitalising the input historical map. However, histogram thresholding does not yield good results if the input historical map is of low quality and affected by the aliasing and false color effects, or if the input map is too complex and contains many classes of geographic features. For these reasons, I investigated and compared other historical maps digitalisation methods.

Gobbi et al. [2019] developed an object-oriented supervised classification method. As training data are needed, the method uses training point data, which are more easily generated than training polygons. Efficient results are obtained with a reduced number of training points. An important pro of this method is that it provides a way to cope with textual labels, lines and symbols so that the labelled geographic features are not split in several parts. Moreover, this method yields better results than histogram thresholding by better coping for the aliasing and false color effects as the classification uses shape properties and not only color properties. The only downside of this method is that it is not fully automatic as Gobbi et al. [2019] developed it using GRASS GIS. Object-oriented classification methods such as this one can be used when the input historical map has a sufficient spatial resolution such that objects can be formed at the building level for instance. If the input map contains lots of mixed pixels because of a lower spatial resolution, other methods should be used instead. Provided a sufficient spatial resolution, this method is flexible and can be used for various historical maps. The higher the quality of the input map, the higher the quality of the results.

Henderson et al. [2009] and Herrault et al. [2013b] investigated pixelwise unsupervised classification algorithms which classify historical maps automatically without requiring any manual intervention from the user. Such algorithms do not require the user to produce any training data and they yield better results than histogram thresholding. Although, the preprocessing steps used in Herrault et al. [2013b] allow to remove textual labels and lines so that the geographic features are not split in several parts, implementing it reduces the quality of the results as it smooths the input historical map and details are lost. As a result, the contour of the geographic features is less properly identified by the classification algorithm. Note that it does not remove entirely symbols. Other cons of these methods is that some geographic features get wrongly classified and small features are more difficult to identify. Moreover, the user sometimes needs to set a large number of clusters for separating the different features. Manual intervention is then needed for merging the classes representing the same feature category afterwards. For these reasons, these methods perform well for input historical maps with a limited number of classes (i.e. 5 to 7) as the geographic features are properly classified without requiring the user to reassemble together the different categories of features afterwards. The preprocessing steps can be used for historical maps of low spatial resolution (e.g. 30 m) where the aim is to recover large areas such as forest areas or built-up areas where the accuracy of the contour of the geographic features matters less.

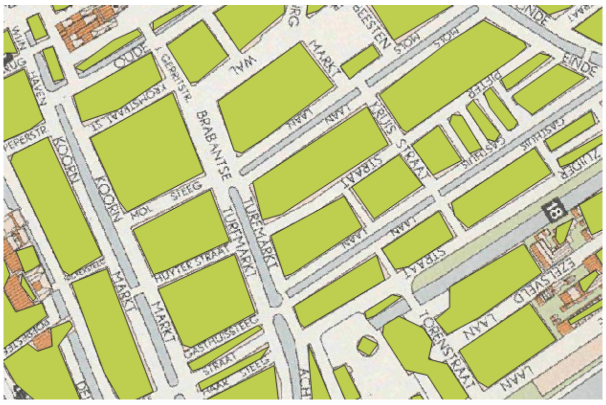
scikit-learn supervised algorithms allow to classify historical maps automatically without requiring any manual intervention from the user. These algorithms yield better results than histogram thresholding but they require lots of training data for providing better results than the unsupervised classification algorithms. Using a reduced number of training polygons, geographic features get misclassified and small size features are not identified. Besides, they do not provide a way to cope for textual labels, lines and symbols on the historical maps and this causes some geographic features to be split in several parts. Considering the use cases, these algorithms can be used if training data is already available or can be easily or quickly created. If the input historical maps contain many categories of geographic features, these methods provide better results than unsupervised classification algorithms.

Building footprints generalisation methods

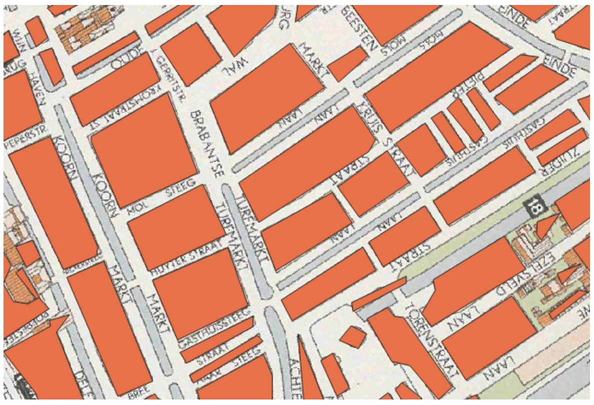
I used, implemented and compared several generalisation methods for building footprints. Figure 5.6 shows the results of the different methods. The pros, cons and use cases of these methods are described in the following paragraphs.

Douglas and Peucker [1973] (DP) developed a method easily and quickly implementable in a GIS (e.g. QGIS). It can be used directly as the only generalisation step applied on the vectorised building footprints, whether or not the contour of the vectorised building footprints is highly irregular. However, the method does not preserve topology (holes and concave shapes) and overlap can be generated between neighbouring building footprints, whether they are separated by a small gap or directly adjacent. In fact, this algorithm is a line simplification algorithm and is thus not suited for polygons generalisation. Building footprints generalisation sometimes requires adding new vertices and this is not possible with the DP algorithm as it relies only on existing vertices. Therefore, I would not recommend using this method for simplifying building footprints.

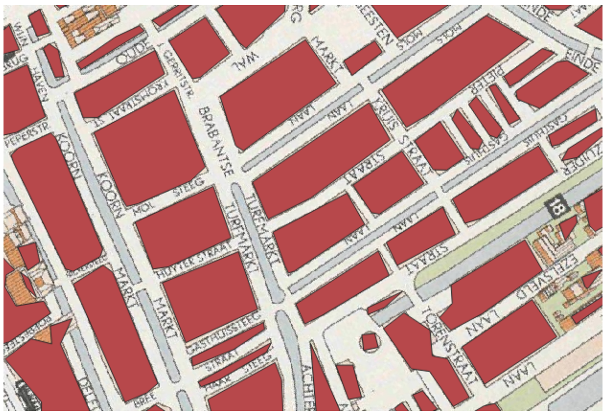
Commandeur [2012] designed a method which tends to create regular shapes and makes it possible to force the parallelism and right angles in the generalised building footprints. Topology is preserved (holes and concave shapes) and direct adjacency (i.e. sharing an edge) between building footprints is considered in the generalisation process to avoid overlap. However, the method cannot be used directly as a first generalisation step on the vectorised building footprints which have lots of irregularities in their contour. The building footprints need first to be slightly generalised with another operator. As other downside, the algorithm does not always work with small building footprints and tends to create peaky shapes for elongated building footprints. Also, overlap can be generated between neighbouring building footprints separated by a small gap. This is even reinforced because of the peaky shapes effect. Considering both pros and cons, this method can largely be used for generalising building footprints extracted from historical maps, provided that it is not the first generalisation step applied. Besides, this method provides better results than the other methods investigated when the input building footprints dataset contains directly adjacent building footprints as the generalisation algorithm tries to avoid overlap in the generalised building footprints.



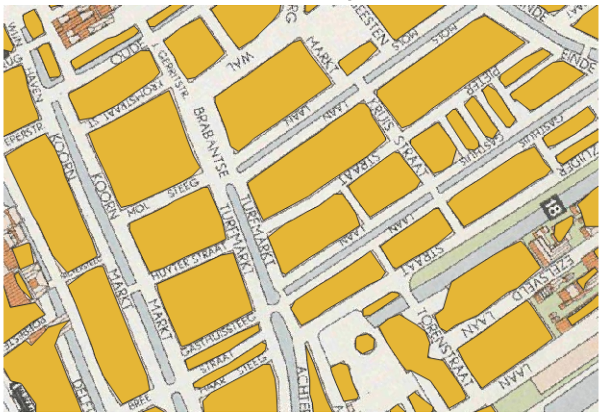
(a) DP generalisation (α = 10, tolerance = 5)



(b) Generalisation of Commandeur [2012] (α = 12, distance threshold = 5, angle threshold = 120)



(c) Shapely generalisation (α = 10, tolerance = 5)



(d) CGAL generalisation (α = 10, ratio = 0.03)

Figure 5.6: Results from different building footprints generalisation methods implemented as second generalisation steps after simplifying the original building plots with an α-shape operator

Shapely simplification algorithm can be used directly as the only generalisation step applied on the vectorised building footprints, whether or not their contour is highly irregular. It makes it possible to preserve the topology (holes and concave shapes). Another pro of this method is that it does not generate overlapping building footprints as the generalised building footprints are always smaller than the originals. However, this also has a downside as the contour of the generalised building footprints matches less with the edges of the building footprints on the historical maps, especially if the tolerance used is high. This is even worst when the edges of the building footprints were not properly recovered by the maps digitalisation method. For this reason, this method can be used for generalising building footprints that were *properly* extracted from input historical maps. It will not yield good results if the map digitalisation process did not perform well. The option to preserve the topology should always be used, otherwise the algorithm implemented is a DP simplification algorithm.

CGAL simplification algorithm allows to preserve the topology (holes and concave shapes) but overlap is easily generated between neighbouring building footprints, whether they are separated by a small gap or directly adjacent. Even when applying this method as a second generalisation step, the contour of the building footprints is still irregular and still contains many vertices. Besides, extreme tolerance values must be used for obtaining the best results obtainable with this method, and at a certain point, changing the tolerance values does not change the generalised building footprints anymore. For these reasons, I would not recommend using this method in the context of extracting building footprints from (historical) raster maps. It does not generalise enough the vectorised contour of the building footprints. In case the user still wants to implement it, only the ratio and euclidean distance modes should be used. Indeed, with the count mode a fixed number of vertices is kept in the contour of the generalised building footprints, and this number is the same for all of them. Thus, it makes it difficult to consider variations in the shape complexity of the building footprints.

5.2.2 Final building plots extraction workflow

Based on these experiments and comparisons, the final building plots extraction method described in Section 3.2 was developed. I implemented the methodology with four Delft historical maps (Section 4.1.1). Figure 5.7 shows the intermediary results of the different steps of the workflow for the Delft map of 1961. As different methods in this workflow rely on user-defined parameters, their influence on the results is described in the following subsections and guidance is provided regarding the choice of the parameters values. For the specific values for all Delft historical maps, see Appendix B (Table B.4).

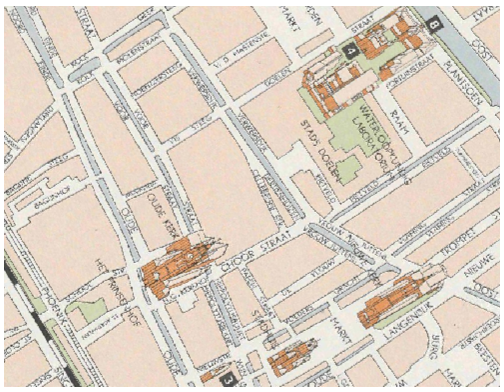
Effect of parameters values on the maps digitalisation results

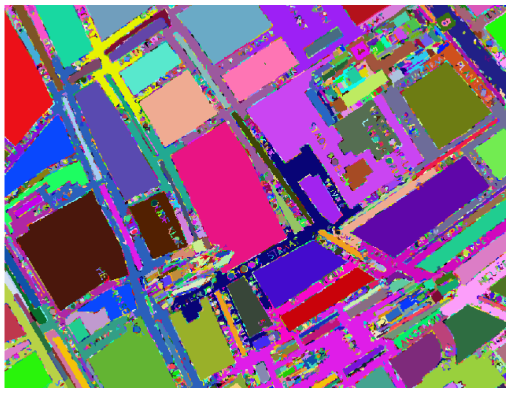
Hyperparameters tuning is used to determine the best values for the segmentation parameters that are, as described in Section 3.2.1, a threshold representing the maximum dissimilarity between adjacent segments and the minimum number of pixels required to form a single segment. Although the best parameters values are automatically found by the hyperparameters tuning, it still requires the user to provide ranges of values to be tested during the tuning process. As all possible combinations of values for the two parameters are tried by the algorithm, the longer the ranges, the longer the computational time and it can takes up to several hours for long ranges of values. Experiments with the four Delft historical maps allowed to determine relevant range values for the two parameters from a visual assessment. In that way, it was found that all the maps yield better segmentation results with threshold values lying between 0.07 and 0.1. With lower values, many small segments are created and it results in many small patches of pixels that are further wrongly classified by the OBIA classification. On the contrary, with higher values, bigger segments are created and it causes the building patches of pixels to be merged with other features in their neighbourhood such as other buildings or even features from other categories. For the minimum segment size, all maps find their best parameter values between 5 and 10 pixels. Lowering or increasing these values tends to create respectively smaller or bigger segments and it thus has the same effect as lowering or increasing the threshold parameter values. It should be noted that the minimum segment size depends on the spatial resolution of the input map; lower resolution maps would require higher values for the minimum segment size.

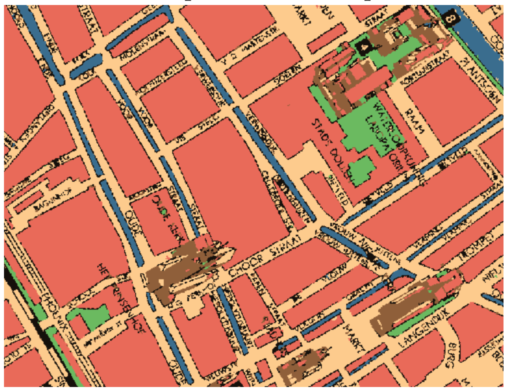
Effect of parameters values on the generalisation results

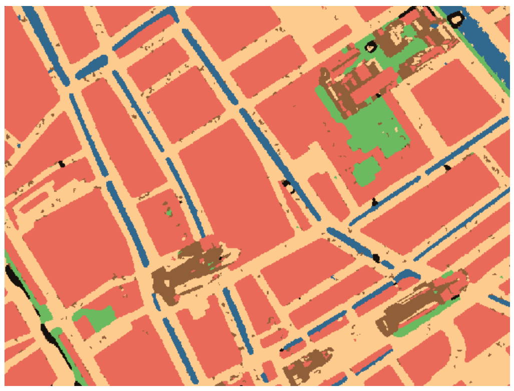
The first generalisation method used—the α-shape operator of Arteaga [2013]—requires a user-defined α parameter value (Section 3.2.2). This value strongly depends on the results of the previous OBIA segmentation and classification steps. Indeed, if those steps provided good results such that the contour of the extracted building plots was properly recovered, then the building plots need to be less generalised and lower α values can be used. On the contrary, higher α values must be used for generalising the building plots when their contour is noisier. In that way, the building plots extracted from the historical maps of 1961 and 1982 have neater vectorised contours and they are thus generalised with a low α value (i.e. 10–12). This allows to preserve all concave shapes. For the two older maps, dating from 1880 and 1915, good generalisation results are obtained with higher α values (i.e. 30).

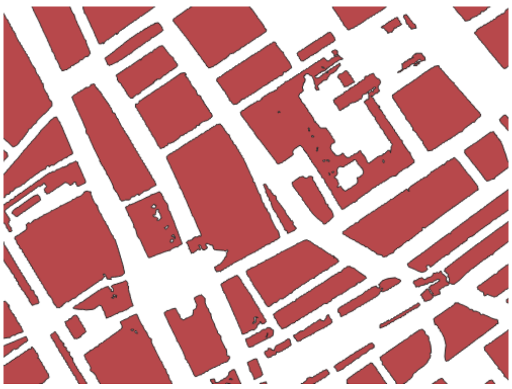
The second generalisation method used—the method of Commandeur [2012]—makes use of two user-defined parameters: an angle and a distance threshold. After experimenting different values, the same observations were made for all Delft maps. The best generalisation results are obtained with an angle threshold above 70° so as to generalise enough the building plots. As for the distance threshold, it must always be lower than 5 m. With higher distance thresholds, peaky shapes tend to be created in the generalised building plots, especially for thin and elongated building plots (Figure 5.8). In addition to these parameters, the method of Commandeur [2012] allows to ensure parallelism and right angles. In this thesis, this functionality was not used as it tends to move the building plots from their true location such that they do not exactly match with the original building plots on the historical map (Figure 5.9).

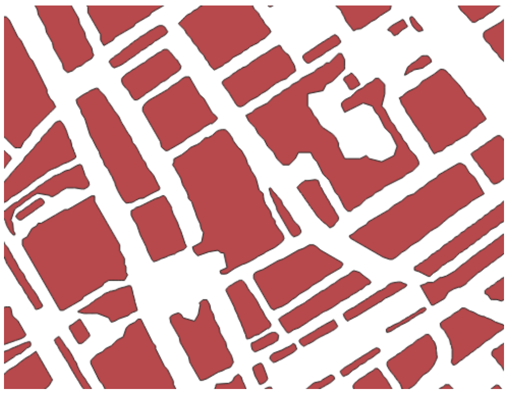

(a) Input historical map


(b) Segmented raster map


(c) Classified raster map. Red = buildings, blue = canals, green = vegetation, light orange = streets, black = text and symbols, brown = other.


(d) Classified raster map after text removal. Red = buildings, blue = canals, green = vegetation, light orange = streets, black = text and symbols, brown = other.


(e) Vectorised building plots


(f) Building plots after the α-shape generalisation

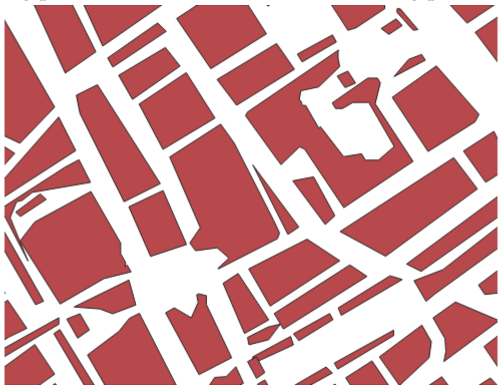
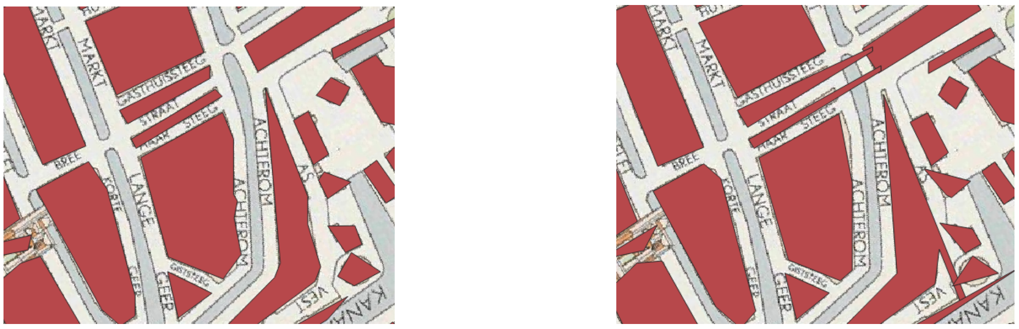

(g) Building plots after the generalisation of Commandeur [2012]

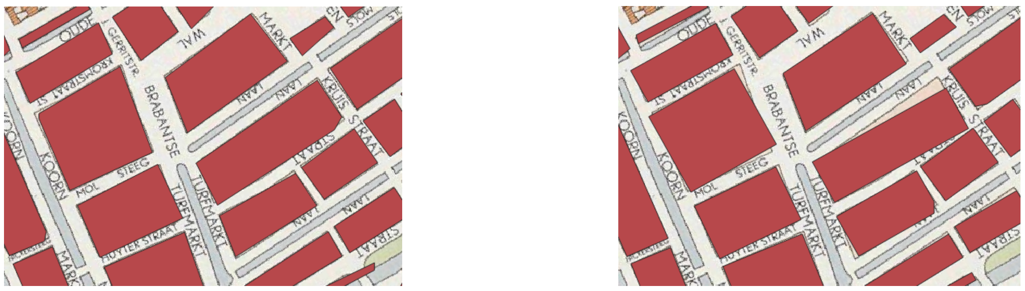
Figure 5.7: Intermediary results of the building plots extraction method applied with the Delft historical map of 1961



(a) Distance threshold equal to 5

(b) Distance threshold equal to 6

Figure 5.8: Effect of increasing the distance threshold in the method of Commandeur [2012]. Peaky shapes start to appear in b due to a higher distance threshold.



(a) Generalised building plots without enforcing parallelism and right angles

(b) Generalised building plots with enforced parallelism and right angles

Figure 5.9: Effect of forcing the parallelism and right angles in the generalised building plots. When parallelism and right angles are enforced, building plots are shifted from their true location (b).

5.2.3 Results assessment

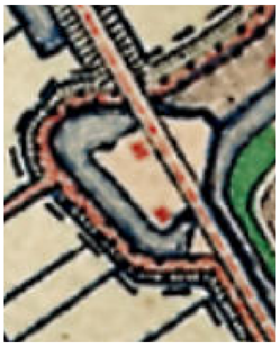
In this section, the results of the different quality assessments conducted are described and analysed. The results of the building plots extraction method implemented with the four historical maps of Delft (1880, 1915, 1961 and 1982) are compared using different quality metrics in order to evaluate the efficiency of the method with historical maps that date back from different time periods, come from different providers and were made using different cartography rules. Appendix B contains information about the ground truth datasets needed to conduct these quality assessments and about the training datasets associated with the metric values described in this section (Table B.2).

Performance assessment

The performance of the building plots extraction method is assessed by computing the precision, the recall and the F-score from a point-in-polygon procedure with a ground truth data points dataset and the extracted building plots (Section 3.2.3). The values obtained for the four Delft historical maps are shown in Table 5.3. The Delft map of 1880 yields low precision and recall values respectively equal to 60.73% and 67.44%. These low values are mainly explained by a high confusion between the building class and the road class which are both represented in red on the historical map. The road features are not represented as straight lines but instead they are depicted using spotted lines (Figure 5.10a). As a result, the spots constituting the roads are easily confounded with small and thin rectangular building plots, leading to a high rate of false positives (roads classified as buildings) and a high rate of false negatives (buildings classified as roads). It should be noted that it mainly regards the building plots located outside the walls of the city as all building plots inside the city walls are large and thus properly identified. The low recall and precision values obtained for Delft 1880 are also due to the fact that the historical map is affected by the false color effect (Section 2.1). Because of that effect, the color inside the building plots varies between red and white, which complicates the building plots extraction process. For the Delft map of 1915, the precision is high (92.67%) which means that there are only a few false positives—other features classified as buildings. However, the recall is low (56.05%) because of the confusion between the road and the building class. Such as the map of 1880, both categories are represented in red and roads are depicted with both straight lines and twisted pair lines (Figure 5.10b). As a result, lots of small building plots outside the city walls get wrongly classified as road features. Finally, both the maps from 1961 and 1982 have high precision (93.63% and 95.41%) and recall (88.92% and 90.51%) values. This means that there are only a few false positives and false negatives. The main classification errors regard some small building plots that were classified in other categories of features. In addition, the historical map from 1961 contains some 3D drawings for representing important historic buildings, as shown in Figure 5.11. The roofs of these buildings are also detected as building plots.

Historical map	**Precision (%)**	**Recall (%)**	**F-score (%)**
Delft 1880	60.73	67.44	63.91
Delft 1915	92.67	56.05	69.85
Delft 1961	93.63	88.92	91.21
Delft 1982	95.41	90.51	92.90

Table 5.3: Performance metrics computed for the four Delft historical maps


(a) Delft 1880

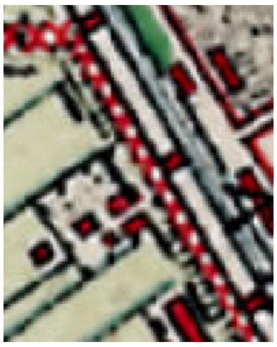

(b) Delft 1915

Figure 5.10: Similar symbology and colors (red) used to represent the roads and the buildings on the Delft maps of 1880 and 1915

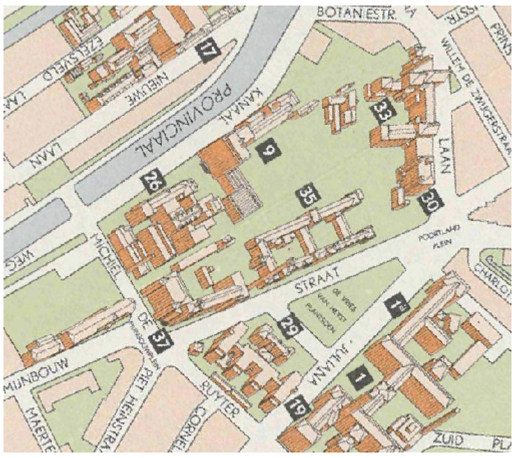


Figure 5.11: 3D drawings on the Delft historical map from 1961

Shape assessment

To assess whether the shape and the contour of the building plots were properly recovered from the input historical maps, the generalised buildings plots are compared with ground truth using two metrics: the averaged difference between turning functions and the averaged shape similarity based on buffers (Section 3.2.3). Table 5.4 shows the metric values for the four Delft historical maps. All maps yield low values for the difference between turning functions. This indicates that the generalisation step produced smoothed building plots contour without spikes or other irregularities and that the angles of the original building plots were preserved. As for the averaged shape similarity, the maps from 1915, 1961 and 1982 yield high metric values. This means that the edges of the building plots were properly identified on the historical maps and that the generalised building plots thus perfectly match with their ground truth. This metric value is lower for the map from 1880 because the edges of the building plots were not properly recovered from the historical map and the generalised building plots are thus smaller than their ground truth (Figure 5.12).

Historical map	Shape similarity (%)	Turning functions difference
Delft 1880	90.56	1.11
Delft 1915	97.02	1.20
Delft 1961	98.29	1.26
Delft 1982	98.40	1.18

Table 5.4: Shape metrics computed for the four Delft historical maps

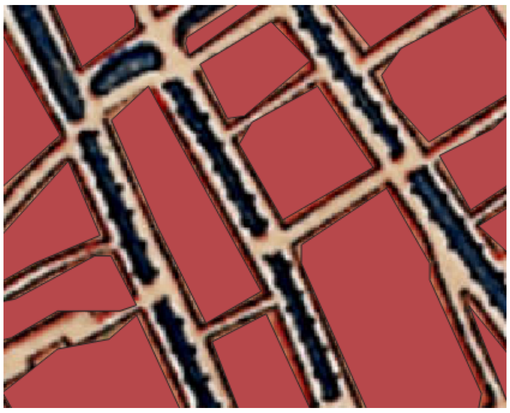


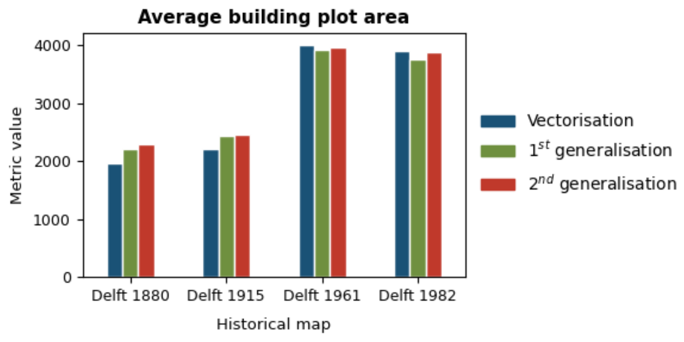
Figure 5.12: Generalised building plots overlaid over the Delft map of 1880. The edges of the original building plots are not properly vectorised.

Metrics

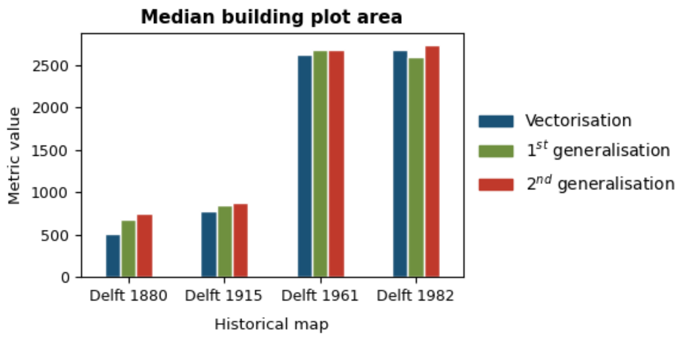
Different metrics are computed after each step of the building plots extraction method to keep track of the effect of the different steps on the extracted building plots (Section 3.2.3). The most relevant ones to understand or analyse the results are presented in this section. Figure 5.13 displays bar plots showing the evolution of the metrics computed after the vectorisation and generalisation of the building plots. See Appendix B for the specific metric values (Table B.5). Several observations can be made from the bar charts:

- The average and median area of the vectorised building plots is clearly lower for the maps from 1880 and 1915 (Figure 5.13a and Figure 5.13b). This is also the case for the median perimeter (Figure 5.13c). This supports the explanation given about the low precision and recall values for Delft 1880 and 1915; the two maps contain smaller building plots which are more easily confounded with other types of features.

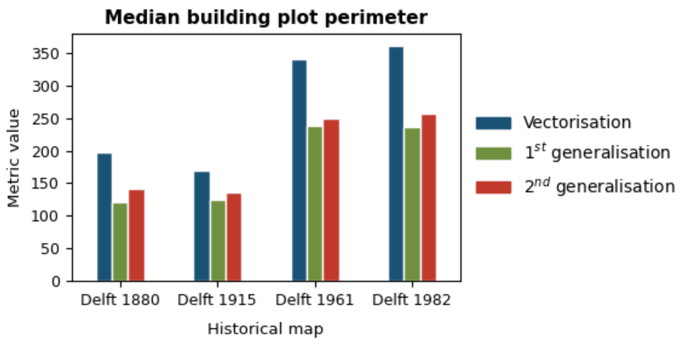
- The median number of vertices of the vectorised building plots is higher for the maps from 1961 and 1982 (Figure 5.13d). This can be explained by the higher spatial resolution of these maps. As a result, when the building plots are vectorised, their stair-like boundary is more detailed as the pixels on the historical maps are smaller. However, as the generalisation steps are implemented, the median number of vertices drops, for all maps, and the generalised building plots datasets all have a median number of vertices equal to 5 (Figure 5.13d). This shows that the generalised building plots extracted from the four Delft historical maps all have the same level of generalisation in the end.
- The average convexity of the building plots from all maps increases as the generalisation steps are implemented (Figure 5.13e). The same information is found by looking at the number of concave building plots which decreases with the generalisation steps (Figure 5.13f). One may think that the generalisation workflow implemented does not preserve concave shapes but by comparing the number of concave vectorised building plots with the total number of vectorised building plots it is found that all building plots are actually concave. This is because of the irregularities in their contour which make them all concave whereas the global shape of the building plot is actually convex.
- The median rectangularity increases with the generalisation steps for all maps (Figure 5.13g). This shows that the generalisation workflow implemented tends to create regular shapes, which is a desirable characteristic for the output building plots.
- The number of building plots is kept constant between the vectorisation and the first generalisation step (i.e. α-shape generalisation), for all maps (Figure 5.13h). However, some building plots are lost between the first and the second generalisation step (i.e. generalisation of Commandeur [2012]). This is because the method of Commandeur [2012] sometimes fails on small building plots.



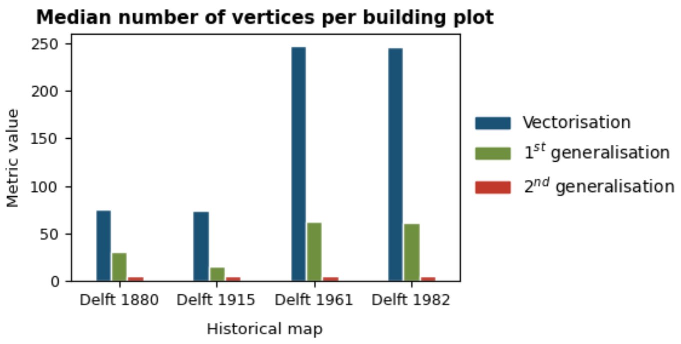
(a)



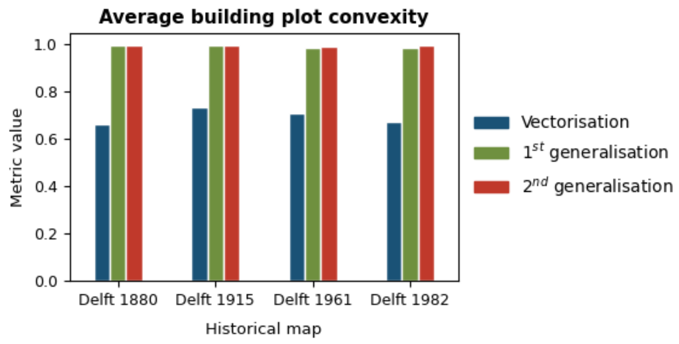
(b)



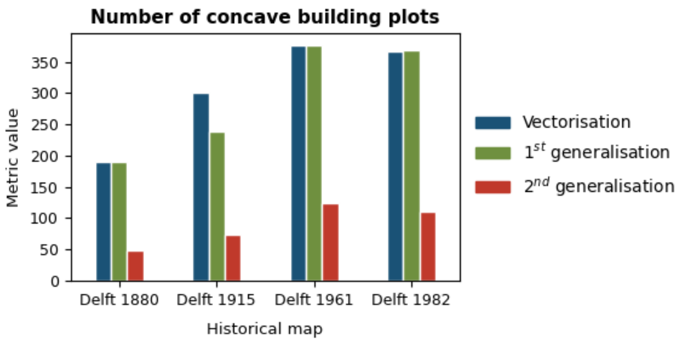
(c)



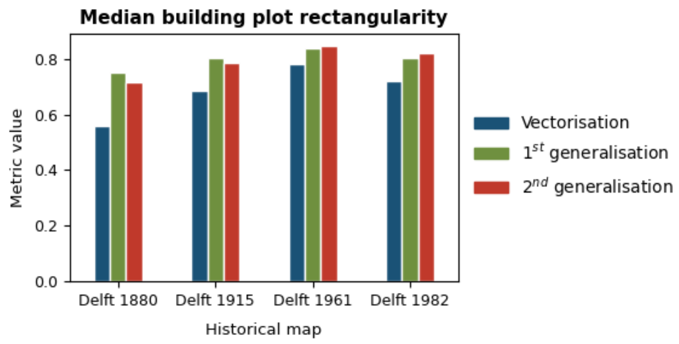
(d)



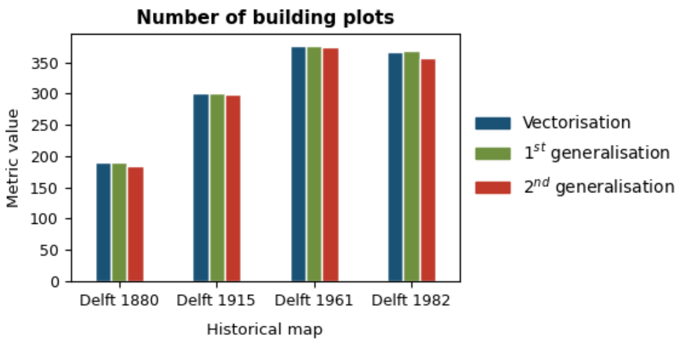
(e)



(f)



(g)



(h)

Figure 5.13: Bar charts showing the different metrics computed for the historical building plots after (1) the vectorisation step, (2) the first generalisation step (α-shape generalisation) and (3) the second generalisation step (generalisation of Commandeur [2012])

5.3 Reconstruction of individual footprints from building plots

In this section, the results of the building plots decomposition into building footprints using the one-to-many alignment (1:n) and the 2D procedural modelling processes are presented. Three points of analysis are addressed: (1) the limits of the 1:n alignment, (2) the specific cases of the 2D procedural modelling and, (3) the combination of the two types of building footprints datasets.

5.3.1 One-to-many alignment limits

The results of the 1:n alignment process are variable depending on the input historical map. In that way, it was found after a visual assessment that the 1:n alignment produced good results for the two more recent Delft maps (1961 and 1982), although there is a small offset between the building plots on the historical maps and their aligned BAG building footprints (Figure 5.14). Different elements could explain this offset; it can be due to chronometric or geometric inaccuracies in the input historical maps or to the quality of the georeferencing (Section 3.1.1). However, as described in Section 5.1, different georeferencing transformations have been tested and there was still an offset with all of them. Therefore, that offset is really likely to be due to geometric inaccuracies affecting the historical maps, as shown in Section 5.1. For the two older maps (1880 and 1915) coming from the Topotijdreis website[1] and thus georeferenced by the Kadaster, high inaccuracies are found by comparing them with more recent maps of the Topotijdreis collection (Figure 5.15). The Kadaster itself mentions these inaccuracies on their website. As a result, the historical building plots and their corresponding building footprints are sometimes located far from each other and they are not matched by the maps alignment process. This point is further discussed in Section 6.1.2.

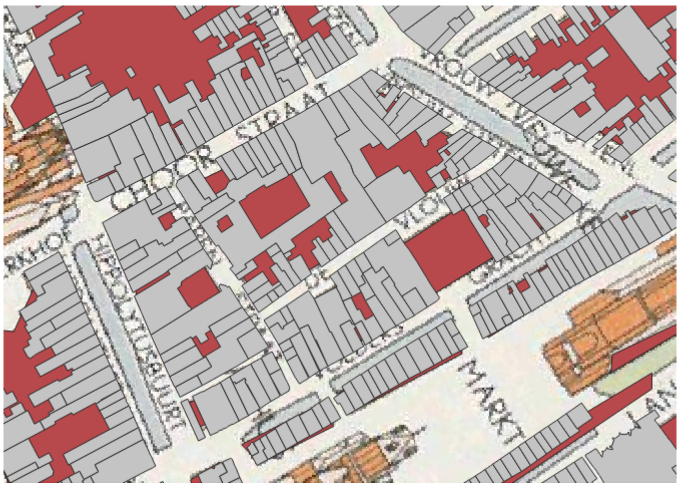


Figure 5.14: Offset between the BAG building footprints (grey) and the corresponding historical building plots (dark red) extracted from the Delft map of 1961

5.3.2 2D procedural modelling specific cases

Among the five cases described to subdivide automatically the building plots into building footprints (Section 3.3.2), case 5 was developed to handle the subdivision of concave building plots by first subdividing them into convex polygons before processing them individually to generate building footprints. Experiments have been conducted to determine the added value of using case 5 instead of treating the concave building plots as convex building plots directly. It was found that for some concave building plots decomposing them into convex polygons creates more realistic results than processing the concave building plots directly (c vs. d in Figure 5.16). However, this is not always the case and sometimes

[1]Available at `https://www.topotijdreis.nl/`.

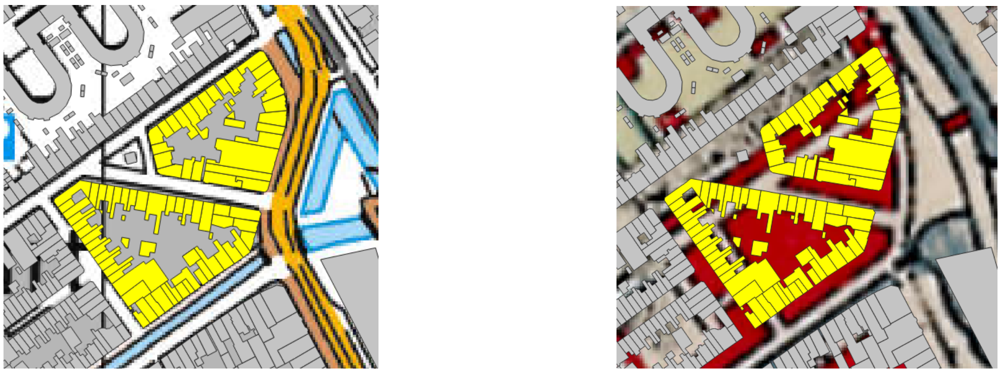

(a) BAG footprints overlaid onto the Delft map of 2019 (b) BAG footprints overlaid onto the Delft map of 1915

Figure 5.15: Overlay of the BAG building footprints over two maps of the Topotijdreis collection. In a, the yellow-colored BAG footprints perfectly match with the corresponding building plots (in grey) on the map. In b, these yellow-colored BAG footprints are offset with respect to their corresponding building plots (in red).

not decomposing the concave building plots into convex ones leads to more realistic results (a vs. b in Figure 5.16). More importantly, the specific case designed for concave building plots is more flexible as it allows to handle the subdivision of any concave building plot without generating invalid geometries or overlap, which is not true for all the other cases (i.e. 1 to 4). As a result, the 2PM building footprints—created using 2D procedural modelling—have been generated using all the cases including case 5.

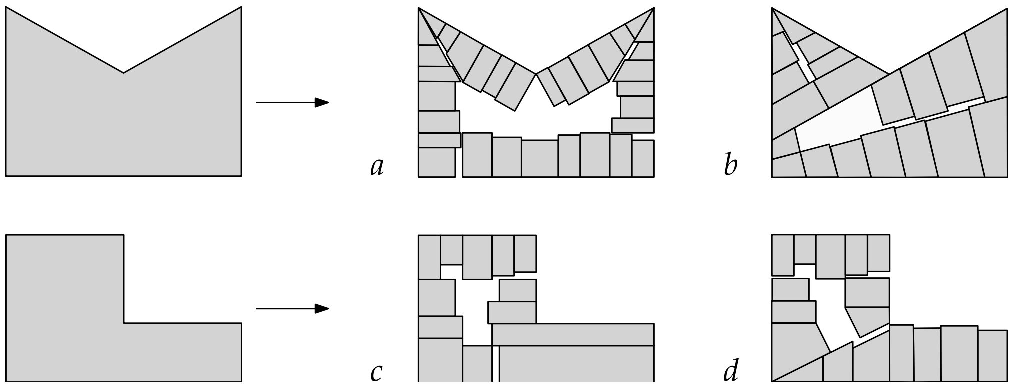


Figure 5.16: Subdivision of concave building plots into building footprints. Left-sided building footprints (a and c) have been created without using case 5 while the right-sided ones (b and d) have been generated using it. In a, the resulting building footprints look more realistic than in b and the opposite is observed in c and d.

5.3.3 Combining the two building footprints datasets

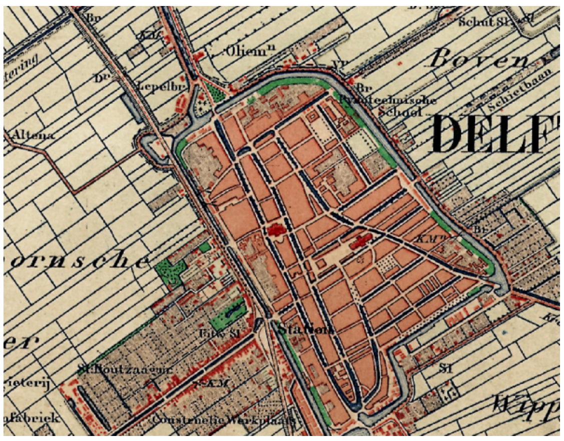
The output building footprints dataset is created by merging the aligned building footprints with the 2PM building footprints. Table 5.5 shows the number of aligned building footprints and 2PM building footprints for each Delft historical map for the same study area (4.01 km^2). The different numbers displayed follow logical trends; the ratio of aligned building footprints to 2PM building footprints decreases with the age of the map as very old buildings (i.e. from 1880) are less likely to remain than more recent ones (i.e. from 1982). However, still a lot of building footprints are created with 2D procedural modelling after 1982 (2958), which seems too high if we think of these buildings as new constructions. In fact, these 2PM building footprints also include buildings that have been refurbished after 1982. In the

BAG dataset used for the maps alignment process, the year of construction of the buildings is updated when the buildings are refurbished. As a result, these buildings are not used in the maps alignment process as they are removed beforehand by the selection step based on the year of construction of the buildings (Section 3.3.1).

Another trend that is interesting to comment is the variation in the total number of building footprints which highly increases from Delft 1880 (3783) to Delft 1982 (13074), as a consequence of the urbanisation process. As shown in Figure 5.17, the proportion of built-up areas has largely increased in Delft between 1880 and 1982 especially in the outskirts of the city. The same information is found in Table 5.5 by looking at the number of building plots which also increases from 1880 (184) to 1982 (358). Note that the ratio between the total number of building footprints and the number of building plots should not be compared between the four periods as it was shown in Section 5.2.3 that the average building plot area varies for each map and this influences the number of buildings. Lastly, the large increase in the total number of building footprints could also be due to large differences in the precision and recall values between the four historical maps (F-score of 63% for 1880 vs. 92% for 1982) (Section 5.2.3).

Historical map	Building plots	Aligned building footprints	2PM building footprints	Total building footprints
Delft 1880	184	674	3109	3783
Delft 1915	298	2129	4052	6181
Delft 1961	366	9271	4493	13764
Delft 1982	358	10116	2958	13074

Table 5.5: Number of building plots, aligned building footprints and 2PM building footprints for each Delft historical map. Note that these values refer to the same study area such that they can be directly compared.


(a) Delft 1880

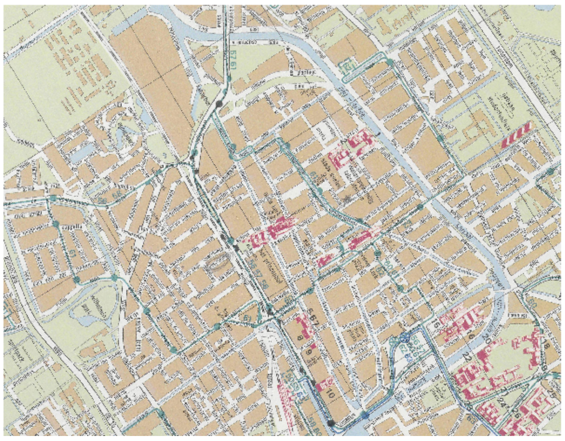

(b) Delft 1982

Figure 5.17: Urbanisation of Delft between 1880 and 1982. The proportion of built-up areas has largely increased in Delft especially in the outskirts of the city.

Figure 5.18 shows the output building footprints dataset for the map from 1961, where aligned building footprints are represented in grey and 2PM building footprints in red. Several observations can be made from this figure:

- The 2PM building footprints and the aligned ones are very similar. The main difference between the two types of building footprints is that some aligned ones are concave when an extension was added to the building footprint while all 2PM building footprints are convex and thus do not have any extensions.

- At some points it can happen that some aligned building footprints are overlapping with 2PM building footprints but this case is limited as an overlap check was implemented for aligned and 2PM building footprints belonging to the same original building plot. The only overlap cases thus happen between building footprints coming from different building plots.
- In some cases, there is a slight misalignment on the street side between the aligned building footprints and the 2PM building footprints in the same historical building plot. Sometimes that offset is larger because of inaccuracies in the input map and as a consequence, 2PM building footprints are created inside the interior courtyard made by aligned building footprints (Figure 5.19). This is explained by the limits of the one-to-many alignment (Section 5.3.1) as there is sometimes an offset between the aligned building footprints and their corresponding building plot.

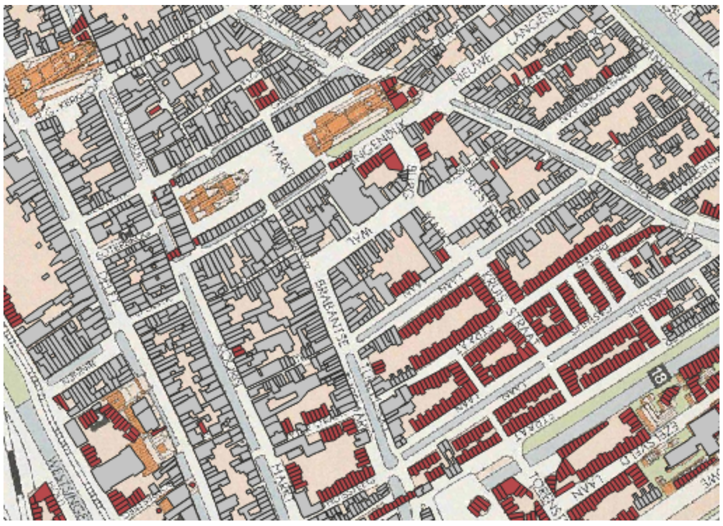


Figure 5.18: Final building footprints dataset for Delft 1961 with aligned building footprints in grey and 2PM building footprints in dark red.

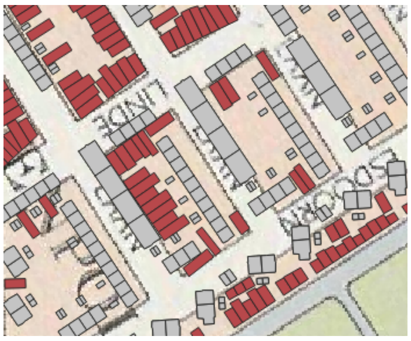
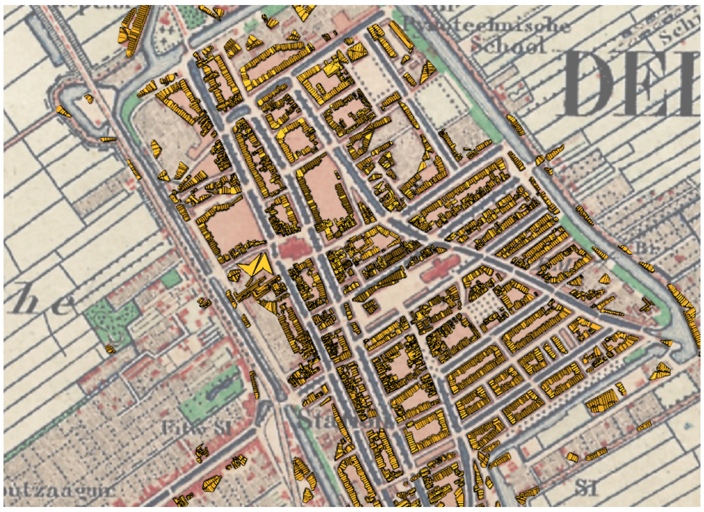
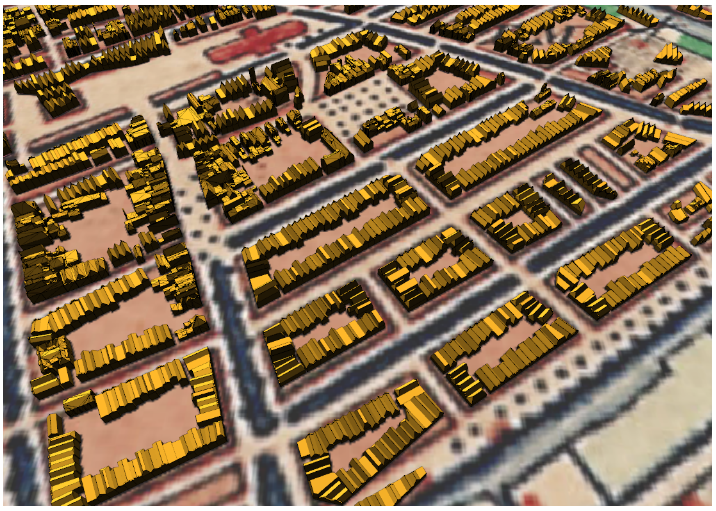


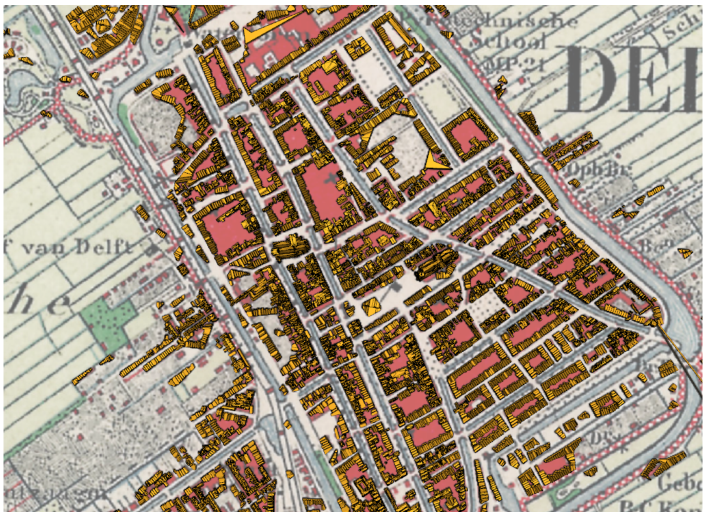
Figure 5.19: 2PM building footprints (dark red) generated inside the courtyard made by aligned building footprints (grey)

5.4 LoD2 buildings generation

Once the building footprints dataset has been created, the last step of the methodology deals with generating LoD2 buildings from the building footprints. Figure 5.20 and Figure 5.21 show the output historical 3D city models reconstructed for each Delft map. The results of the assessment of these historical 3D city models are presented in the following subsections.


(a) Historical 3D city model of Delft in 1880


(b) Zoom over the historical 3D city model of Delft in 1880


(c) Historical 3D city model of Delft in 1915

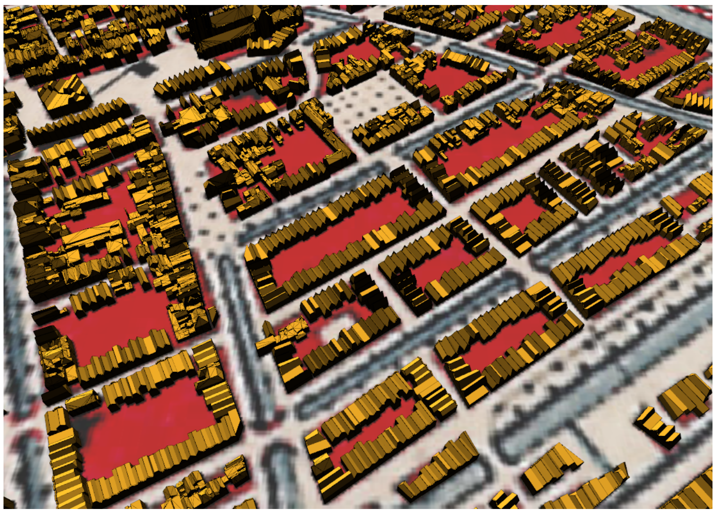
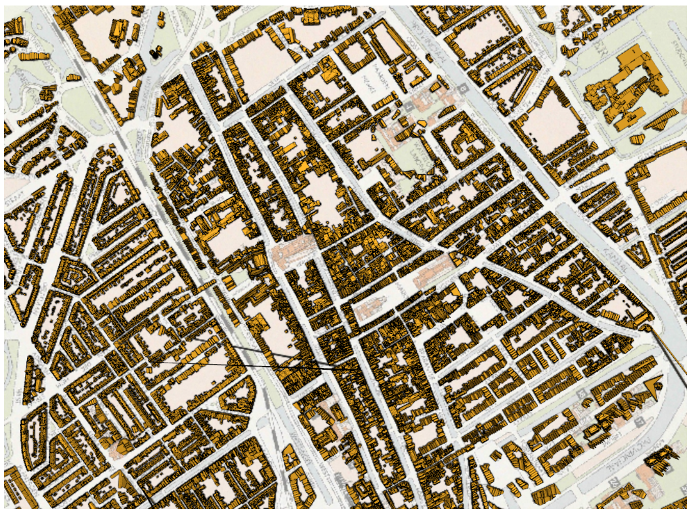
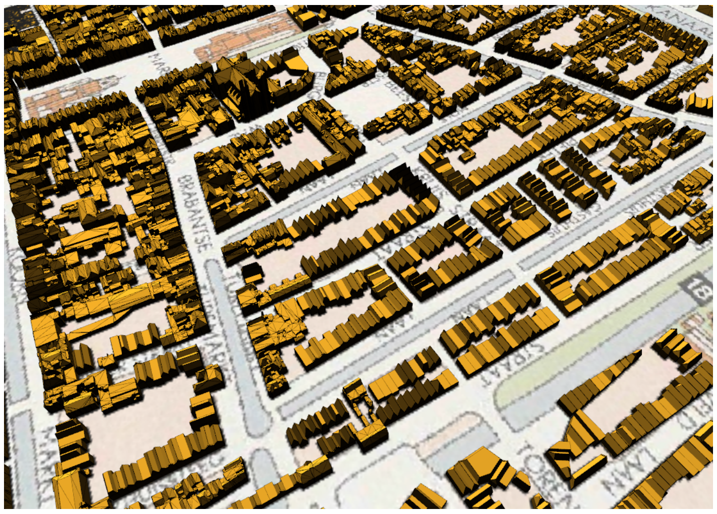

(d) Zoom over the historical 3D city model of Delft in 1915

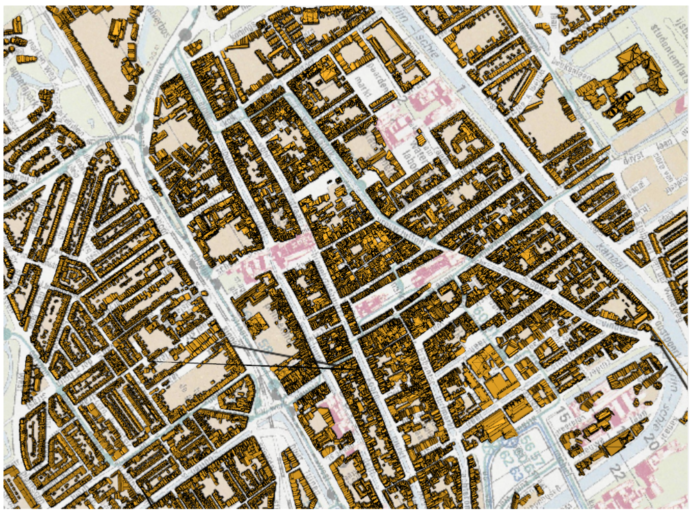
Figure 5.20: Historical 3D city models of Delft in 1880 and in 1915. The models are overlaid over the historical map from which they were reconstructed.



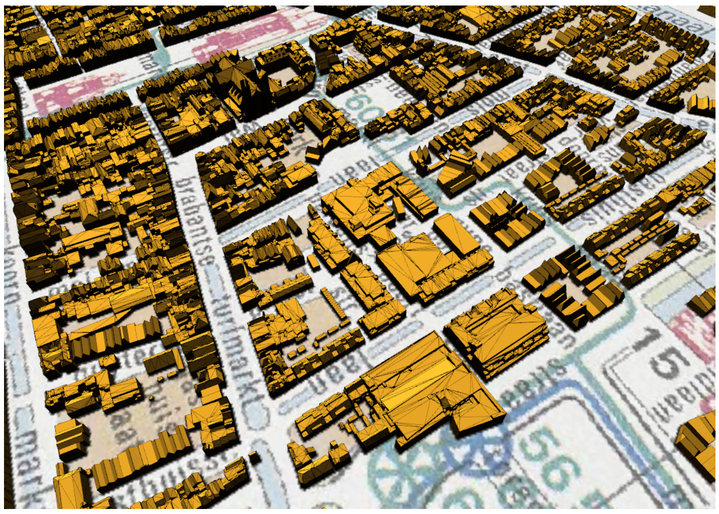
(a) Historical 3D city model of Delft in 1961



(b) Zoom over the historical 3D city model of Delft in 1961



(c) Historical 3D city model of Delft in 1982



(d) Zoom over the historical 3D city model of Delft in 1982

Figure 5.21: Historical 3D city models of Delft in 1961 and in 1982. The models are overlaid over the historical map from which they were reconstructed.

5.4.1 Visual assessment

As noticed in Section 3.4.3, the output historical 3D city models of Delft are multi-LoD as the BAG buildings are LoD2.2 while the 2PM buildings—associated with 2PM building footprints—are LoD2.0. The visual assessment showed that the 2PM buildings have been generated as expected. The semantics were properly assigned as shown in Figure 5.22. Besides, the street facade of the buildings was almost all the time built on the street side. The only limitation spotted is that the assigned roof height is sometimes too high in comparison to the building height below the roof, as shown in Figure 5.23. These irregularities come from the way the roof height attributes are assigned, i.e. using a neighbourhood analysis (Section 3.4.1). Such errors happen in some areas of the historical maps where there are less aligned building footprints with height attributes. As a result, in these areas, the roof height attributes of the 2PM building footprints might only be based on one or two neighbouring aligned building footprints. If these few aligned building footprints have a very high roof height—maybe their height is not accurate or one of these buildings is a chapel—then these values are still assigned to the surrounding 2PM building footprints which end up with abnormally high roof height attributes.

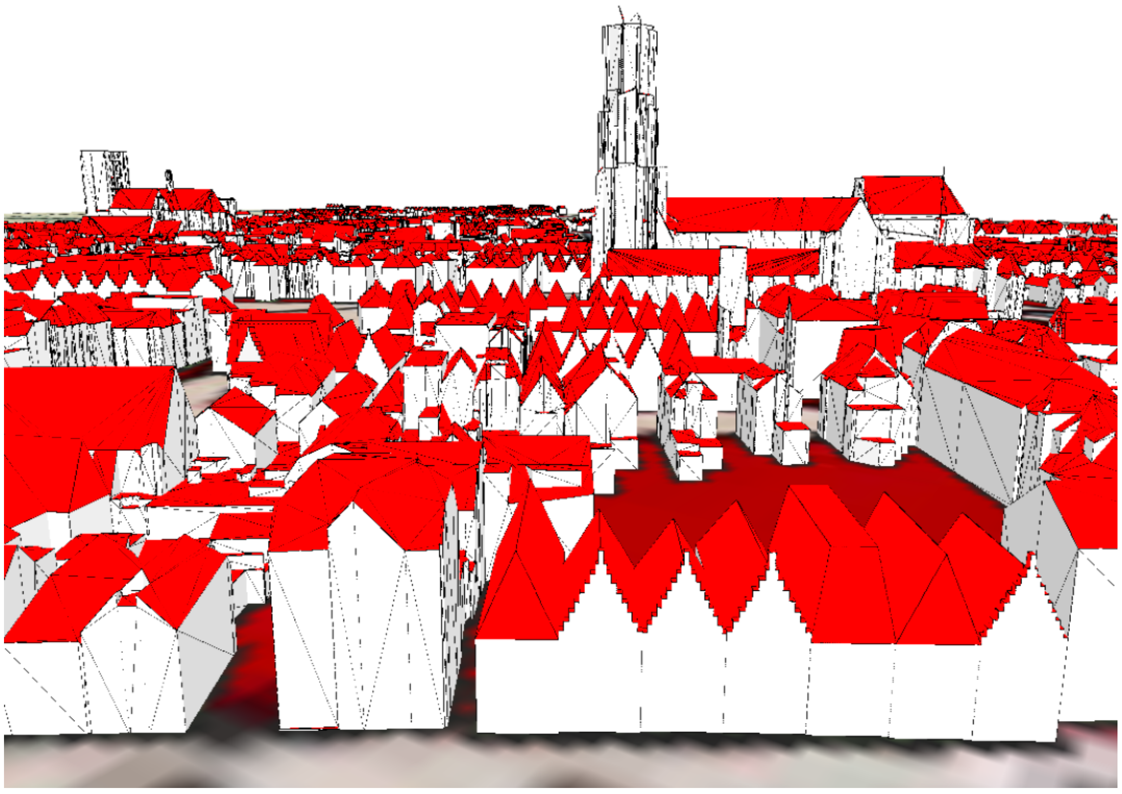


Figure 5.22: Historical 3D city model of Delft in 1915 with semantics. The roof surfaces are depicted in red and the wall surfaces are in white.

5.4.2 Validity assessment

The validity of the CityJSON files generated was first assessed with cjio which showed that all files are valid against their schema. The validity of the geometries was then assessed using val3dity. The results of this validation are shown in Table 5.6 for all Delft maps for the buildings generated with 3D procedural modelling, as the geometry of the aligned buildings from the 3D BAG is not reassessed here. The four historical 3D city models are all valid at more than 99%. The few validity errors that occur in these models are presented in Table 5.7. Note that invalid geometries tend to combine several types of errors. Most of the errors that occur (104–203–204) appear with abnormally high crow-stepped gable roofs (Figure 5.23). The high roof height combined with the facade length values that were used causes

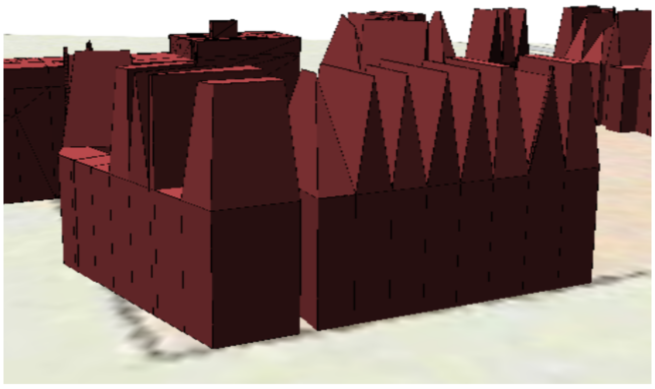


Figure 5.23: Example of 3D buildings with abnormally high roof height

the pitch roof to be very steep. As a result, at the summit of the roof, the last crow steps that are created on opposite sides of the gable end are very close to each other, especially with low step height values. Therefore, there is at the top of the roof in the stepped gable end lots of vertices created very close to each other and almost at the same location in certain cases. This could be an explanation for the 104, 203 and 204 errors as increasing the step height in these cases prevents the creation of these almost overlapping vertices and does not generate any validity errors anymore. Anyway, the errors that are created in the historical 3D city models do not come from the input datasets but are related to specific building configurations that are not well handled in the 3D procedural modelling implementation.

Historical map	Validity (%)
Delft 1880	99.87
Delft 1915	99.97
Delft 1961	99.38
Delft 1982	99.22

Table 5.6: Validity of the historical 3D city models of Delft

Error type	Context	Relative occurrence (%)
102 – CONSECUTIVE_POINTS_SAME	Specific isolated cases.	4.8
104 – RING_SELF_INTERSECTION	Occurs with crow-stepped gable roofs with abnormally high roof height such as in Figure 5.23.	11.3
203 – NON_PLANAR_POLYGON_DISTANCE_PLANE	Occurs with crow-stepped gable roofs created for very irregular building footprints.	8.1
204 – NON_PLANAR_POLYGON_NORMALS_DEVIATION	Occurs with crow-stepped gable roofs with abnormally high roof height such as in Figure 5.23.	74.2
301 – TOO_FEW_POLYGONS	Specific isolated cases probably due to a wrong choice of parameter values.	1.6

Table 5.7: Validity errors in the historical 3D city models reconstructed with their context and their relative occurrence

5.5 Results for historical maps of Brussels

In order to assess whether the methodology could be easily implemented with other study areas and whether it performs well for those other study areas, it was tested with historical maps of the city of Brussels. This section presents the results of that implementation. First, the results of the building plots extraction method are presented. Then, the historical 3D city models reconstructed for Brussels are described and assessed.

5.5.1 Building plots extraction

Choice of user-defined parameters values

To implement the methodology with the historical maps of Brussels, it was first needed to define some values for the user-defined parameters that are used in the map digitalisation and in the generalisation steps. Regarding the map digitalisation user-defined parameters—i.e. the threshold of dissimilarity between segments and the minimum size of a segment in pixel units—I used the same ranges of values that were used for the Delft study case and that were provided as guidance in Section 5.2.2.

As for the generalisation parameters—i.e. the α parameter and the angle and distance thresholds between buffered lines—using the same values than the ones used for the historical maps of Delft was not sufficient for generalising enough the building plots. For the Brussels maps from 1890 and 1924, higher values were used for the α parameter (i.e. 25–30) and for the distance threshold (i.e. 10–12 m). Note that in Section 5.2.2, it was not recommended using a distance threshold above 5 m as it was creating peaky shapes in the generalised building plots, especially for thin and elongated building plots. It is not the case here as the building plots on Brussels historical maps tend to be bigger and thicker. As for the Brussels map from 1700, the α-shape generalisation failed regardless of the α parameter value. As explained in Section 3.2.2, the algorithm behind the α-shape operator works by converting the α-shape of the set of vertices in the building plots boundary to a circular graph. On the map from 1700, the building plots have specific shapes as they are made of a large interior courtyard containing gardens and other geographic features (Figure 5.24). As a result, the algorithm of the α-shape generalisation fails to create a circular graph and the graph ends up either not circular or disconnected. Therefore, in order to generalise the building plots extracted from that map, I checked other generalisation methods described in Section 5.2.1 that had been tested with Delft maps and that were shown to provide good generalisation results. In that way, the Brussels building plots from 1700 have been generalised with the Shapely simplification algorithm. This is further discussed in Section 6.1.1. The specific values for the map digitalisation and generalisation parameters for all Brussels historical maps are detailed in Appendix B (Table B.4).

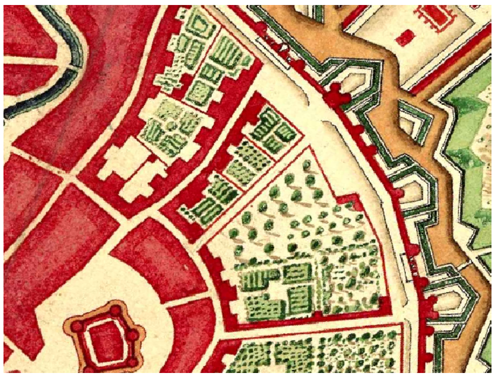


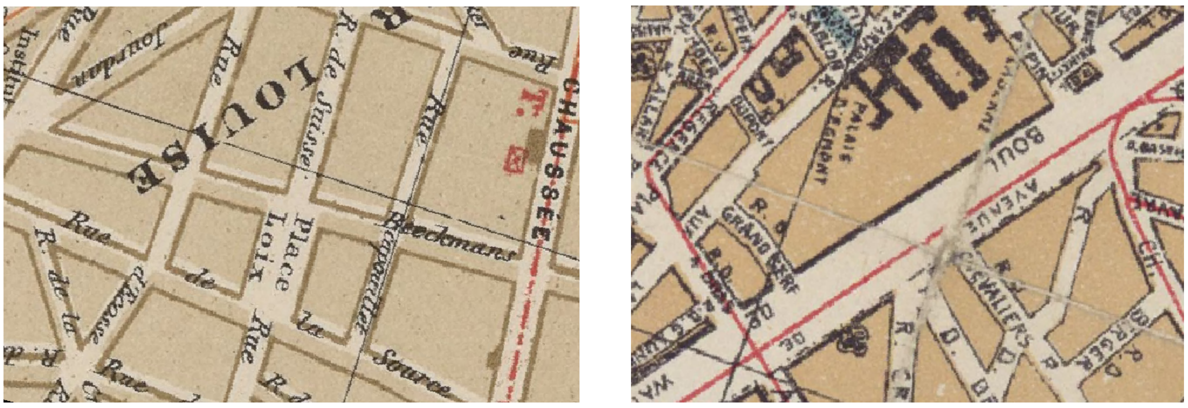
Figure 5.24: Example of building plots on the historical map of Brussels from 1700

Results assessment

As for Delft historical maps, the results of the building plots extraction method have been assessed with a series of metrics. The results of the performance and shape assessments are shown in Table 5.8. Overall, the values obtained for the different metrics show that the building plots have been properly identified on the historical maps and that their shape was properly recovered. Only the Brussels map from 1700 shows a lower precision value (78%). This is due to a confusion between the building category and the city walls which are both depicted in red. As for the maps from 1890 and 1924, some building plots are sometimes merged by the building plots extraction algorithm, explaining the few misclassifications errors that occurred. This happens because the building color sometimes goes over the edges of the building plots on these historical maps, such that the adjacent street between two building plots is depicted in the building color (Figure 5.25a). This phenomenon creates some artefacts when recovering the contour of the building plots. In addition, on the map from 1924, there are some fold marks along which the building plots were not properly identified (Figure 5.25b). See Appendix B for more information about the ground truth datasets used to compute these metrics and about the training data associated with these values (Table B.2).

Historical map	Precision (%)	Recall (%)	F-score (%)	Shape similarity (%)	Turning functions difference
Brussels 1700	78.70	89.47	83.74	98.62	0.99
Brussels 1890	89.70	84.62	87.09	96.10	1.37
Brussels 1924	92.73	85.32	88.87	95.17	0.90

Table 5.8: Performance and shape metrics computed for the three Brussels maps



(a) Brussels 1890

(b) Brussels 1924

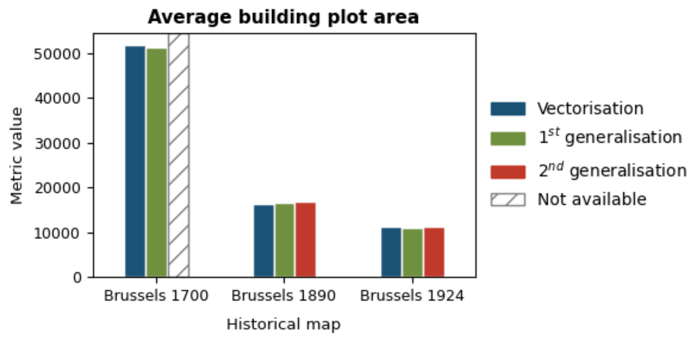
Figure 5.25: Artefacts in Brussels historical maps which complicate the building plots extraction step. In a, the building color goes over the building plots boundary. In b, some building plots are cut into different parts because of fold marks.

In addition to these performance and shape metrics, some additional metrics have also been computed. Figure 5.26 shows with the help of bar charts the evolution of the metrics values after each step of the building plots extraction workflow. The detailed values are given in Appendix B (Table B.6). The bar charts show the following trends:

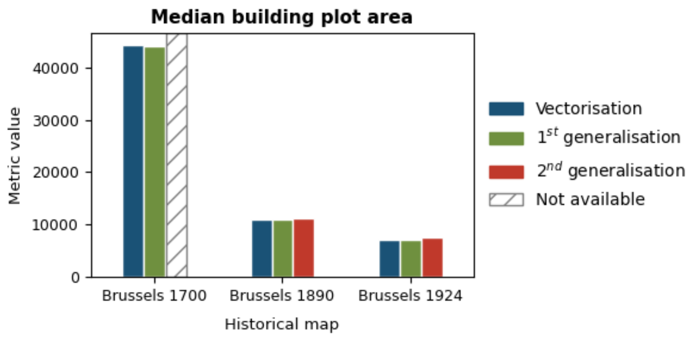
- The average and median building plot area of the map from 1700 is more than four times bigger than the building plot area in the maps from 1890 and 1924 (Figure 5.26a and Figure 5.26b). Similarly, the median perimeter is also larger for the map from 1700 (Figure 5.26c).
- The median number of vertices of the vectorised building plots widely varies for the three maps (Figure 5.26d). However, as the generalisation steps are implemented, the median number of

vertices drops and the generalised building plots of the three maps have a comparable medium number of vertices in the end (i.e. 5 to 8).

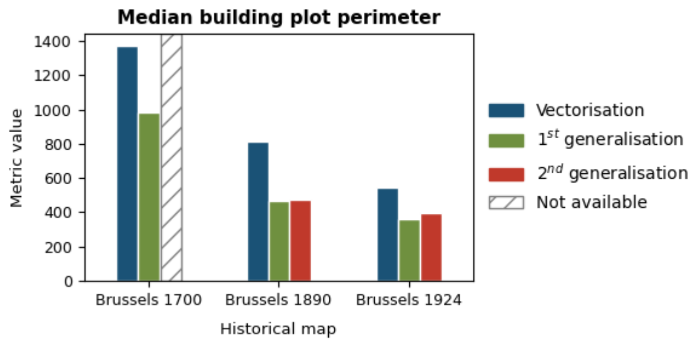
- The average convexity increases with the generalisation steps as the irregularities in the contour of the building plots are removed (Figure 5.26e). This also explains why the number of concave building plots decreases with the generalisation steps; the irregularities in their contour make them all concave while their global shape might actually be convex (Figure 5.26f).
- The median rectangularity increases with the generalisation steps as the generalisation workflow tends to create regular building plots (Figure 5.26g).
- The number of building plots decreases slightly as the generalisation steps are taken for the maps from 1890 and 1924. This is due to the method of Commandeur [2012] which sometimes fails on small building plots (Figure 5.26h). It is however not the case for the building plots from 1700 which have been generalised with the Shapely simplification algorithm and for which the number of building plots stays constant. This shows an advantage of the Shapely simplification algorithm over the generalisation method of Commandeur [2012] as it can handle the generalisation of any polygon regardless of its size and/or complexity.



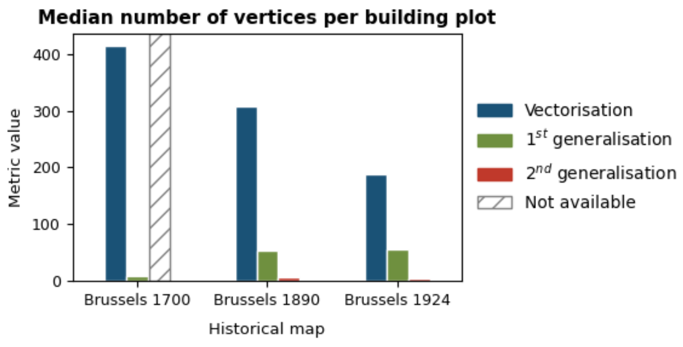
(a)



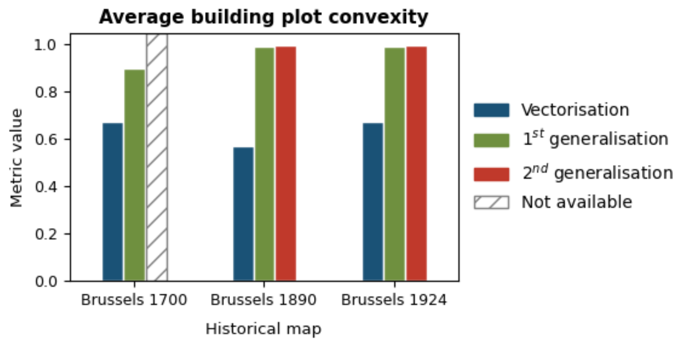
(b)



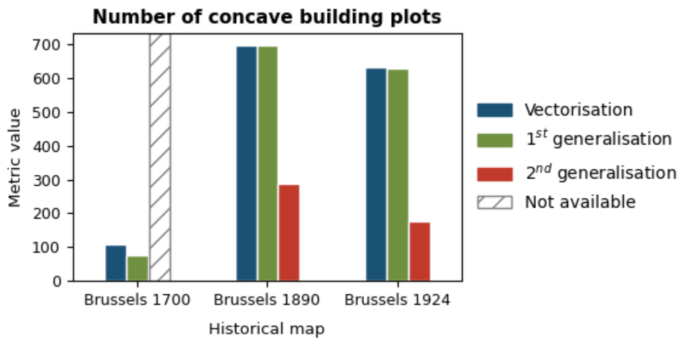
(c)



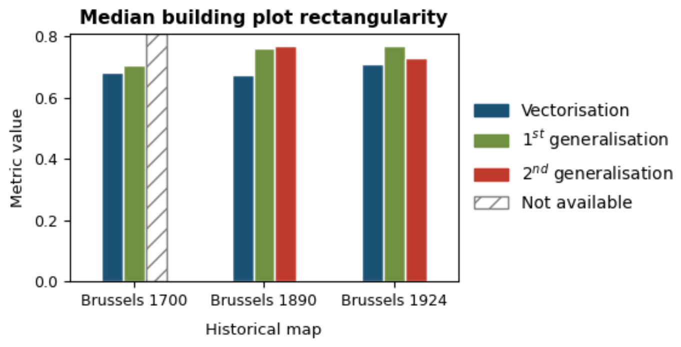
(d)



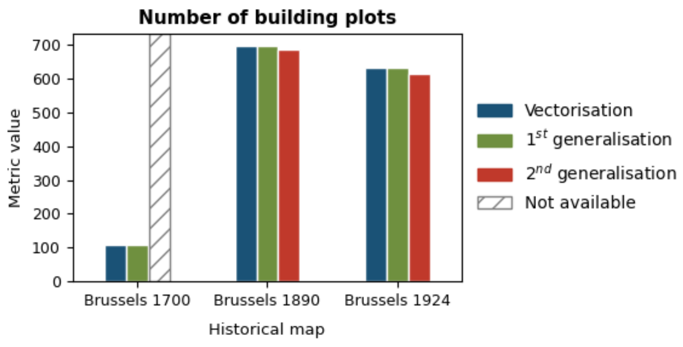
(e)



(f)



(g)



(h)

Figure 5.26: Bar charts showing the different metrics computed for the historical building plots after (1) the vectorisation step, (2) the first generalisation step (α-shape generalisation or Shapely simplification algorithm) and (3) the second generalisation step (generalisation of Commandeur [2012]) if implemented

5.5.2 Reconstruction of individual footprints from building plots

As the optional datasets (2D and 3D buildings datasets with specific attributes) needed to the implement the complete version of the methodology were not available for Brussels, only the shortened version was implemented. Thus the reconstruction of individual building footprints from the building plots was made only using the 2D procedural modelling process. As this significantly influences the running time of the LoD2 buildings generation step, only a sample of the building footprints was kept and used to implement the rest of the methodology. Table 5.9 shows the total and sampled number of individual building footprints for each Brussels historical map for the same study area (8.02 km^2).

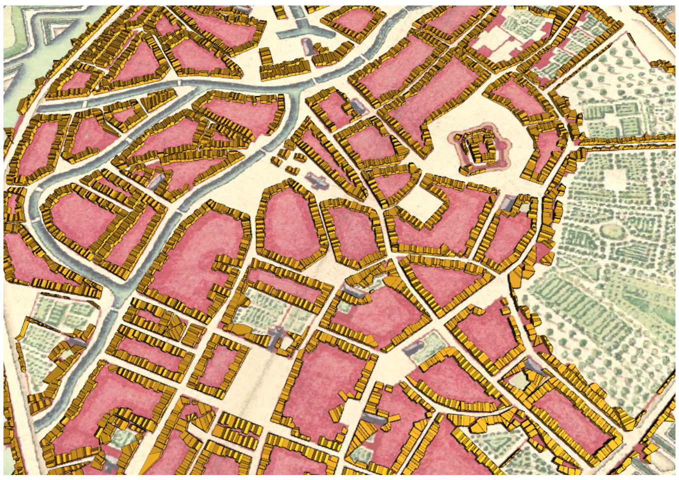
Historical map	Building plots	Total building footprints	Sampled building footprints
Brussels 1700	108	4481	4481
Brussels 1890	687	25202	4976
Brussels 1924	616	15015	3706

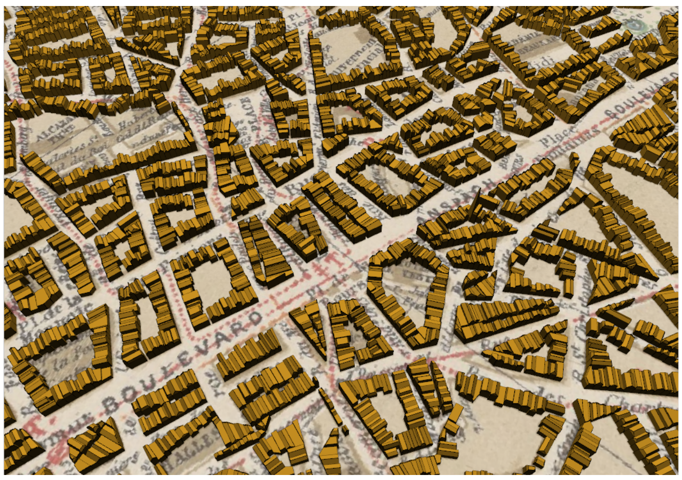
Table 5.9: Number of building plots and building footprints for each Brussels historical map. Note that these values refer to the same study area such that they can be directly compared.

The observations and comments that were made for the Delft study case still holds here (Section 5.3). However, there is one value that stands out from the others; 4481 building footprints in Brussels in 1700 seems to be too little in comparison to the number of building footprints in 1890 (23202) and 1924 (15015). As mentioned in Section 5.5.1, the Brussels building plots in 1700 are four times bigger than the building plots in 1890 and 1924. Besides, when observing the maps in a GIS, it is not possible to identify which building plots are present both in the map from 1700 and in one of two other maps. Therefore, it seems that one building plot from 1700 actually corresponds to several building plots in the other maps. It is thus likely that the building plots from 1700 also contain lanes and alleys and maybe several interior courtyards. This case was not considered in the 2D procedural modelling algorithm and this is why the total number of building footprints in 1700 might be underestimated. Note that the difference in the total number of buildings between 1890 (25202) and 1924 (15015) can also be explained by a difference in the median building plot area as the building plots from 1890 are on average slightly bigger than the building plots from 1924 (Section 5.5.1).

5.5.3 LoD2 buildings generation

The historical 3D city models of Brussels in 1700, 1890 and 1924 are shown in Figure 5.27. Unlike Delft historical 3D city models, the roof height of the 3D buildings in Brussels is here consistent with the building height below the roof. As explained in Section 4.1.2, the roof height was assigned differently for the two case studies; in Delft, the roof height was defined using a neighbourhood analysis while in Brussels the roof height values were based on regulated values coming from the literature. Therefore, there might be less irregularities or errors due to the neighbourhood analysis in Brussels models. This is further discussed in Section 6.1.1.


(a) Historical 3D city model of Brussels in 1700


(b) Historical 3D city model of Brussels in 1890

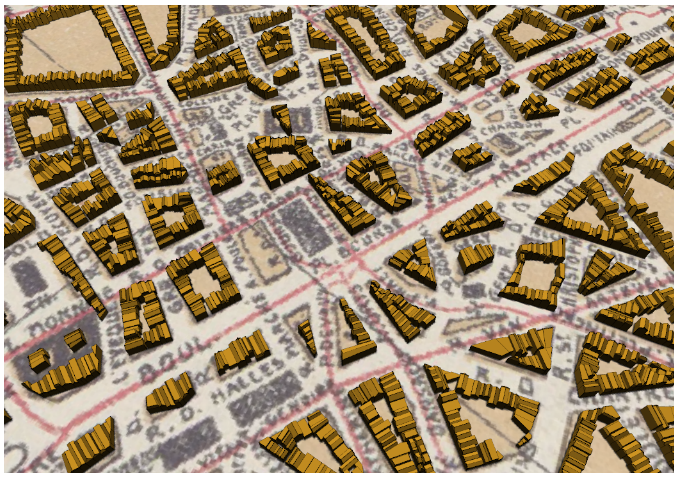

(c) Historical 3D city model of Brussels in 1924

Figure 5.27: Historical 3D city models of Brussels in 1700, 1890 and 1924. The models are overlaid over the historical map from which they were reconstructed.

The validity assessment conducted showed that all historical 3D city models of Brussels are valid against their schema and their geometries are valid at more than 99% (Table 5.10). Note that the few validity errors that occur are the same than the ones affecting Delft historical 3D city models (Section 5.4.2).

Historical map	**Validity (%)**
Brussels 1700	99.73
Brussels 1890	99.96
Brussels 1924	100.00

Table 5.10: Validity of the historical 3D city models of Brussels

6 Conclusions

This chapter is dedicated to the conclusions of this thesis. First, the results of the implementations are discussed (Section 6.1). Hereafter, the research questions of this thesis are answered (Section 6.2) and the contributions of this work are provided (Section 6.3). Finally, the findings and limitations of this study opened a window for future work (Section 6.4).

6.1 Discussion

6.1.1 Application of the methodology in different study areas

The whole methodology was intentionally developed to recreate historical 3D city models of Delft at different time periods. Then it was also tested with historical maps of Brussels to assess the suitability of the methodology for reconstructing historical 3D city models of other study areas. A major difference between the two study areas is the data availability, which strongly influenced the implementation of the methodology. While the optional datasets needed to implement the complete methodology were available for Delft (2D and 3D buildings datasets with specific attributes), this was not the case for Brussels. Therefore it was only possible to use the shortened version of the methodology without the maps alignment process for reconstructing historical 3D city models of Brussels (Chapter 3). This difference in data availability creates a high difference in the methodology running time. Indeed, if remaining historical buildings can be recovered from an existing 3D city model using the maps alignment process, this significantly reduces the processing time. In that way, it took about four hours to reconstruct the city of Delft in 1982 (10116 aligned buildings and 2958 buildings reconstructed with 2D procedural modelling (2PM)) but it took up to fifteen hours to reconstruct the city of Brussels in 1700 entirely with the 2D procedural modelling process (4481 2PM buildings) (Section 5.3.3 and Section 5.5.2). Moreover, for the other Brussels historical maps (1890 and 1924), it was only possible to reconstruct a reduced part of the city due to the too large processing time needed to reconstruct it entirely. Using the complete or the shortened version of the methodology also influences the way the reference roof height is assigned. In that way, in Delft the roof height was defined using a neighbourhood analysis based on the aligned building footprints (Section 3.4.1), while in Brussels the roof height values were randomly generated based on regulated values coming from the literature (Section 4.1.2). As a result, the roof height in Delft might be more accurate at the individual building level as it was defined using a spatial process (i.e. the neighbourhood analysis), but it might also be more prone to interpolation errors or irregularities.

In addition to these implementation changes due to the data availability, some adaptations had to be made regarding the choice of the user-defined parameters values. Although the user-defined parameters of the map digitalisation step required similar values for both Delft and Brussels, this was not the case for the generalisation and 2D procedural modelling steps. Brussels building plots from 1890 and 1924 needed more generalisation than Delft building plots. By comparing the quality metric values, it was found that Brussels building plots are more than four time bigger than Delft building plots, which also leads to larger perimeter and more vertices in the contour of the non-generalised building plots (Section 5.2.3 and Section 5.5.1). These metrics differences explain the need for generalising more Brussels building plots as they have more irregularities in their boundaries. Nevertheless, the average number of vertices after generalisation was the same for all Delft and Brussels building plots at the different time periods, which means that the same level of generalisation was achieved in the end. My advice for implementing the methodology with other study areas would be to check the metrics previously mentioned and choose based on their values the proper generalisation parameters values, based on what was used with Delft and Brussels (Section 5.2.2 and Section 5.5.1). As for the 2D procedural modelling user-defined parameters, the facade length and depth values vary for the two cities as Brussels building footprints tend to be bigger than Delft building footprints. This is in fact highly related to the architectural rules of the cities. For implementing the methodology with other study areas, I recommend either looking in the literature if these architectural rules can be found for the study area. Otherwise, one can also measure in a GIS or in Google Maps the facade length and depth of the current building footprints of the city of interest.

Finally, a last change in the implementation was specific to one of the historical maps of Brussels: the generalisation workflow of the methodology failed with the building plots extracted from the historical map from 1700 (Section 5.5.1). Instead, another generalisation algorithm that had been investigated when comparing different generalisation methods was used (Section 5.2.1). This shows the relevance of the methods comparisons that have been conducted in this thesis; if the implementation still fails with a given historical map, the user can easily adjust the methodology by checking the pros and cons of the methods investigated to find the most suitable one for their needs. Although implementing the methodology for another study area required some changes in the implementation, the results obtained were similar for both case studies. All historical 3D city models were valid at more than 99% and visually the major difference between the historical 3D city models of Delft and Brussels is that Delft models are multi-LoD due to the combination of two 3D buildings datasets, which is not the case for Brussels historical 3D city models (Section 5.4 and Section 5.5.3). In fact, the methodology can be extended to any study area ***as long as the input historical maps are multicolored maps displaying building plots***. I do not guarantee that the methodology will work for historical maps with individual building footprints. Also, although the 2D procedural modelling algorithm works for any given building plot, certain cities might not be characterised by an arrangement of the buildings around an interior courtyard, such that the 2D procedural modelling will not produce realistic results for these cities. From the moment an historical map fulfills this criteria—the historical map is multicolored and displays building plots—there are two additional factors that will influence the performance of the methodology: the quality of the scanning process and the cartography rules of the input historical map.

The quality of the scanning process influences whether the scanned map will be affected by the aliasing and false color effects described in Section 2.1. These two phenomena can greatly influence the building plots extraction step, leading to lower precision and recall values. The scanning process also deals with the choice of a spatial resolution for scanning the historical map and this parameter highly influences the results obtained with this methodology. Indeed, higher spatial resolution means more details in the contour of the geographic features and smoother transitions between pixels of different categories of features (mixed pixels). Besides, the spatial resolution is an important factor to consider when using an object-based classification as such classifications require the geographic features to be big enough to be identified as objects. If the spatial resolution is too low, some geographic features might only be constituted of a few pixels and are then not identified as objects. In this case, a pixelwise classification might be more suitable for identifying the geographic features. The results obtained with different historical maps support this discussion point. In that way, on all historical maps used in this study, the Delft maps from 1880 and 1915 yielded low precision and/or recall values (F-score below 70%) (Section 5.2.3) as they are more affected by the aliasing and false color effects and have a low spatial resolution (2.58 m) (Section 4.1.1). Besides, these low precision and recall values were also explained by the fact that these two maps contain some tiny building plots made only of a few pixels, which were thus not properly identified. In comparison, all other historical maps are less affected by the scanning process defects and have a higher spatial resolution (below 1.80 m) (Section 4.1.1). This thus led to higher precision and recall values (F-score superior at 83 %) (Section 5.2.3 and Section 5.5.1).

The second factor that strongly influences the results is the cartography rules that were used to create the input historical map. For maps with strict cartography rules, the building plots extraction method performs better because all categories of geographic features are depicted in different colors. Contrarily, when the maps have geographic features of different categories represented using the same symbology, it creates some confusion for the building plots extraction algorithm to distinguish between these categories of features. In that way, the lower precision and recall values for Delft 1880 and 1915 were also explained by the symbology of the maps; the building plots and the roads are depicted in the same color, leading to confusion between the two classes.

6.1.2 Challenges of reconstructing individual building footprints

The most challenging part of the methodology implemented was the processing of the building plots extracted from the historical maps to recreate individual building footprints (Section 3.3). One main challenge of this step is that it heavily relies on the results of the previous building plots extraction step. Indeed, if the building plots extraction method yielded bad results regarding the identification and generalisation of the building plots, it will also lead to bad results in the subdivision of the building plots into building footprints. In addition, assessing the results other than visually is difficult as it is rare to find ground truth datasets to compare with. This challenging task of reconstructing individual building footprints was made possible using two processes: 2D procedural modelling and one-to-many alignment (1:n).

The 2D procedural modelling process produces very good results for subdividing regular building plots. However, it is difficult to handle the subdivision of any building plot as these can greatly vary in shape complexity. As a result, the output building footprints from the 2D procedural modelling are not always realistic (Section 5.3.2). More challenges are introduced by the 1:n alignment. As shown in Section 5.3.1, historical building plots and their corresponding current building footprints are sometimes offset with respect to each other as a consequence of geometric inaccuracies in the input historical maps [Uhl et al., 2017; Nobajas and Nadal, 2015]. Section 5.1 and Section 5.3.1 showed that these inaccuracies affect the georeferencing process and that the offset between the historical building plots and their corresponding aligned building footprints remains, with all the georeferencing transformations tested. However, the parameters used in this process (i.e. buffer size and threshold value) can still be adjusted so that the building plots are still matched with their corresponding building footprints despite the offset, but the question arises as to whether or not they should be matched with each other. On the one hand, matching them is a way to cope for the inaccuracies of the input historical maps as the remaining historic buildings will be placed at their true location, even tough they were not properly located on the historical maps [Laycock et al., 2011]. On the other hand, when overlaying these historic buildings at their true location with the input historical maps, they can end up lying over other geographic features (e.g. roads or canals) than building plots. For this reason, it makes it difficult to determine what a good match is. This limit of the 1:n alignment also challenges the reunification of the aligned building footprints and the 2PM building footprints to create the final building footprints dataset. In some cases, the aligned building footprints are slightly misaligned with their adjacent 2PM building footprints as these latter perfectly match with the building plots on the historical maps, which is not always true for the aligned building footprints.

Another challenge of the 1:n alignment is that it requires a specific step to make sure that the entities in the datasets to be aligned date back from the same time period. If this is not ensured, it is very likely that historical building plots that were completely demolished would still be matched with recent buildings built a long time afterwards at the same location. For this reason, the complete version of the methodology requires a current building footprints dataset enriched with the construction year of the buildings. This can also poses a challenge as it is the case with the BAG dataset for instance. Indeed, the year of construction used in the BAG is updated when the buildings are refurbished or partly rebuilt and it is thus not always the actual year of construction of the buildings. Despite all these challenges, the 1:n alignment also leads to great advantages as, in addition to recovering the actual footprints of remaining historic buildings, it also highly speeds up the methodology runtime [Laycock et al., 2011]. For instance, with the Delft map of 1982, the 1:n alignment allowed to recover 10116 historic building footprints so that only 2958 building footprints were generated with 2D procedural modelling and further processed to reconstruct LoD2 buildings (Section 5.3.3). The maps alignment also makes it possible to use the height attributes of the aligned building footprints for inferring the roof height of the adjacent 2PM building footprints. However, it is worth noting that if no matches are found between historical building plots and current building footprints, roof height inference cannot be implemented.

6.1.3 Use cases

The methodology implemented aims at automatically reconstructing simple historical 3D city models from historical maps. In this thesis, simple historical 3D city models were defined as 3D city models made of LoD2 buildings without any other type of features. A literature survey was conducted to define the use cases for such historical 3D city models. Three different axes are identified.

First, simple historical 3D city models greatly help with the **preservation and communication of the historical heritage**. They allow for the widespread diffusion of historical sites over the Internet or in museums for cultural and educational purposes [Balletti and Guerra, 2016; Kersten et al., 2012]. It makes it then possible for a wider and more diversified public to access historical sites that sometimes no longer exist [Maiwald et al., 2019; Balletti and Guerra, 2016]. 3D-enabled visualisation provides for these users a more intuitive way to access the history of a city by navigating in space and time, and thus gaining a better understanding of how it looked like [Maiwald et al., 2019]. Furthermore, simple historical 3D city models also come as a good mean to give a context to more specific reconstructions. For instance they could be used as a basis for reconstructing more detailed historical 3D city models as they already provide the location and geometries of the building footprints recovered from the historical maps. These simple 3D city models could then simply be refined by adding windows and other details in the building facade, using 3D procedural modelling, in order to create more detailed historical 3D city models. By facilitating the reconstruction of highly detailed historical 3D city models, simple historical 3D city models can also find application in the gaming industry [de Boer, 2010].

Alongside these visualisation-based applications, historical 3D city models can also support more **pragmatic applications**. For instance, current 3D city models have already been used to estimate the population of a city [Biljecki et al., 2016a]. This application could be extended to historical 3D city models to estimate the population of a city at times where demographic data were not available. Reversely, combining historical census data with historical 3D buildings could be used to compute the past population density of a city [Kersten et al., 2012]. Other pragmatic applications of historical 3D city models could be found in computational fluid dynamics (CFD), for performing past micro-climate analysis [Biljecki et al., 2015]. In that way, information such as the past ground temperature, wind flows or air quality information could be recovered with the help of historical 3D city models. Henceforth these applications in the demographic domain and in CFD can lead to another major use case: the study of the spatial distribution of historic diseases. For instance, Galanaud et al. [2015] have mapped plague deaths in Dijon at different time intervals in the 15th century to further study the correlation between plague deaths and the socioeconomic status of households. One could imagine extending this research by studying the influence of other factors such as climatic variables and population density derived from a dynamic historical 3D city model covering different time periods. Similar studies could be conducted with another historic disease, tuberculosis, which was studied in relation with air quality and climatic variables over long time periods by Fernandes et al. [2017]. Historical maps of the disease distribution, such as the one shown in Figure 6.1, could be used to reconstruct historical 3D city models and derive tuberculosis driving factors to better understand the propagation of the disease. Although these applications seem to be directed at the comprehension of the past, this is actually still related to present-day challenges as an improved understanding of the propagation of past pandemics can help better prevent future ones [Reyes et al., 2018].

Lastly, if simple historical 3D city models can be more easily reconstructed, it also opens the door for a series of **new applications and use cases that have not been investigated yet**. Indeed, the utilisation of historical 3D city models has been limited because of the difficulty to reconstruct them [Herrault et al., 2013b]. Facilitating their reconstruction is thus likely to lead to new use cases. This is even more true as historical maps and other historical data are already used in many different domains, such as in urban studies, ecology or economy, in which researchers do not always have the GIS and programming skills to reconstruct historical 3D city models [Bshouty et al., 2019]. Therefore, if the availability of these models is increased, they could find new applications in these domains. For instance, one could imagine the appearance of use cases in historical ecology where historical 3D city models could help studying the evolution of the distribution of urban species in a cityscape, over different time periods. Moreover, in economy, the roof type of the 3D buildings could also be used for inferring historical taxes and other information about the socioeconomic status of households.

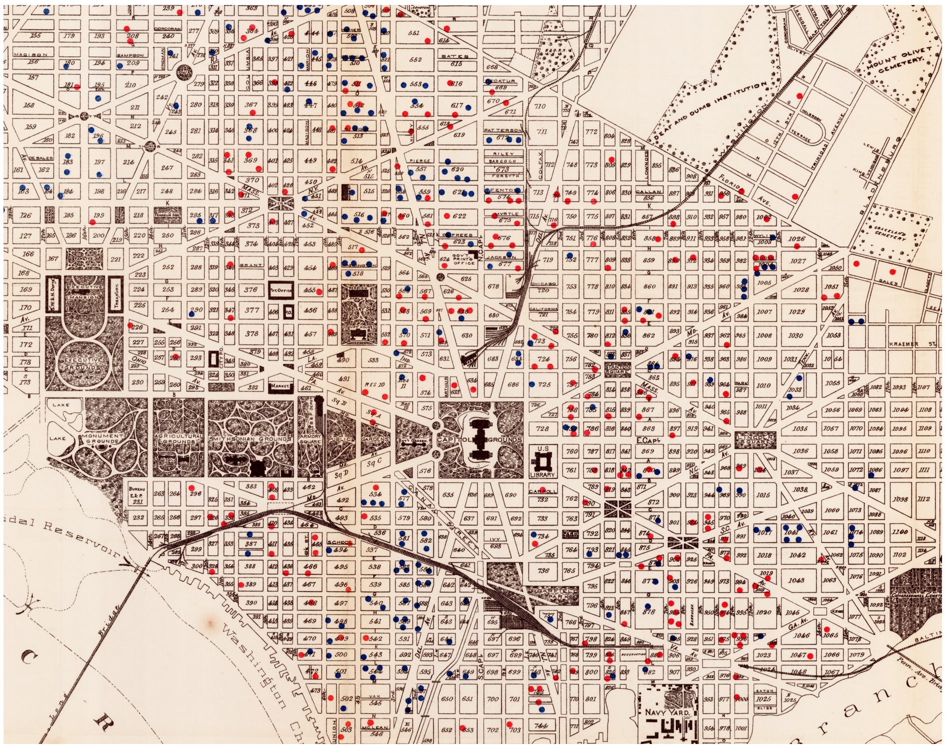


Figure 6.1: Historical map showing the tuberculosis deaths in Washington, D.C. in 1900–1901. Tuberculosis deaths are represented by blue and red spots. Available at https://lib.msu.edu/.

6.2 Research questions

In this section, the main research question of this thesis is answered; *"To what extent can be automated the process of reconstructing simple 3D city models from historical maps?"*. This is done by addressing the sub-questions specified in Section 1.1, based on evidence developed in the other chapters.

- **What are the difficulties in the process of automatically reconstructing 3D city models from historical maps?**

Different difficulties related to historical maps have arisen throughout the implementation of the methodology. Three types of difficulties have been identified: the difficulties due to the intrinsic characteristics of historical maps, the difficulties introduced by their state of conservation and the difficulties driven by the scanning process.

The difficulties due to the intrinsic characteristics of historical maps concern the way the maps were produced. These include the symbology chosen to create the map; is each class of the map depicted in a different color or is the same color used to represent different classes? This strongly influences the map digitalisation step, as discussed in Section 6.1.1, as using the same color for different classes complicates the process. Another characteristic of historical maps leading to challenges in their processing is the presence of hand-written textual features and symbols overlaid over other geographic features that are in consequence more difficult to extract. The presence of 3D drawings sometimes also challenges the processing of historical maps. Furthermore, historical maps often contain geometric and chronometric inaccuracies introduced both by cartographers' subjectivity and the less accurate techniques used back in the day to produce them (e.g. triangulation networks) [de Boer, 2010; Heitzler and Hurni, 2020]. As a consequence, as discussed in Section 6.1.2, these inaccuracies hinder the easy linkage between historical maps and maps dating from other epochs, be they current maps or other historical maps.

Difficulties related to the state of conservation of historical maps depend whether the maps were damaged with time or not. The older the historical maps, the more they are likely to be affected by this category of difficulties. A common phenomenon introduced with time is bleaching, which can affect differently different tiles of an historical map, and thus complicates the map digitalisation process if this latter relies on color information. Other defects can be introduced if the maps were preserved folded, which let large bleached marks on the historical map at the folding.

Lastly, the difficulties caused by the scanning process mainly concern the aliasing and false color effects [Liu et al., 2019]. These two phenomena described in Section 2.1 respectively lead to mixed pixels and high color variations inside geographic features, which both influence the quality of the map digitisation results. The scanning process also deals with the choice of a spatial resolution for the scanned maps. As discussed in Section 6.1.1, this choice can also greatly influence the map digitalisation process as high resolution maps contain more details about the contour of geographic features, and it thus leads to higher quality results. While the intrinsic characteristics and the state of conservation of the historical maps are as is and thus cannot be modified, special care should be taken when scanning historical maps. It should be noted that for some historical maps it remains difficult to properly scan them because of their size or their conservation status [Nobajas and Nadal, 2015].

- **What results do we achieve with existing methods for extracting building footprints from historical maps? What are the pros and cons of these methods?**

The different methods implemented produce good or bad quality results depending on the input historical maps. A clear commonality between all these methods is that they all produce better results with better quality historical maps for which the defects due to the scanning process are limited. However, even with the historical maps for which the methods perform well, no method is perfect and they all have pros and cons which make them suitable only for specific applications. In that way, their pros, cons and applications were described in Section 5.2.1 so that users can find the method that will answer the most their needs in terms of building footprints extraction from historical maps.

- **Which degree of automation can be achieved in the process of reconstructing 3D city models from historical maps? Are some manual interventions necessary?**

The methodology implemented in this study is fully automated such that the only user interventions needed are (1) to generate a training dataset for the building plots extraction step, (2) to provide the file paths to the datasets needed for running the methodology and (3) to pass some user-defined parameters values. Training data are sometimes seen as an obstacle towards automating the process as it usually requires time-consuming work to generate them. However, the comparison between methods in Section 5.2.1 showed that the best methods are the ones relying on training data. To cope for the tedious work of creating training data, the methodology was developed such that training points are used instead of training polygons, as training points are more easily and quickly generated. Another element that is sometimes considered as a roadblock towards automation are user-defined parameters, as it sometimes requires lots of trial and error work for finding the most suitable values. However, as presented in Section 5.2.2, different collections of historical maps of the same study area required similar values for most of the user-defined parameters, even tough the maps had different providers, spatial resolution, symbology and other characteristics. Thus these values (Table B.3 and Table B.4 in Appendix B) and the reflection about it (Section 5.2.2) provide guidance for implementing the methodology with other historical maps. As a result, the methodology can be easily implemented as it does not require lots of input from the user.

Therefore, except for the creation or preparation of the required datasets, the methodology implemented does not require any manual interventions for reconstructing historical 3D city models from historical maps. One great advantage of this methodology is its flexibility as it can be implemented with various collections of historical maps of variable quality, while lots of the existing methods that were investigated only work for a specific map collection (Section 5.2.1). However, a consequence of that flexibility is that the results are not perfect, in the sense that some errors or discrepancies are sometimes introduced. For instance, churches and other special buildings are not always recovered with this methodology as they are often represented in different ways on historical maps (e.g. symbols, 3D drawings, building plots), and it thus difficult to implement a methodology that would recover them automatically regardless of the way they are represented. Thus, in certain cases, these buildings should be manually added. One which want very accurate and perfect results could implement a simple and short manual cleaning step after the building plots extraction and the reconstruction of individual building footprints steps. Note that the amount of manual processing that could be implemented to have the most accurate results depends on the input historical maps. As discussed in Section 6.1.1, if the map was not properly scanned, more manual work will be needed for compensating the low recall and precision values obtained (Section 3.2.3). On the contrary, high recall and precision values would require almost no manual processing. Note that this thesis was essentially an experimental work to investigate whether automation was possible or not. I believe that each step of the methodology could benefit from additional focus to solve these discrepancies. This is further developed in Section 6.4.

Although the methodology is automatic, one downside might be its running time, which can be large depending on the size of the study area. This is essentially due to the LoD2 buildings generation step which is more time-consuming than the other parts of the methodology. Therefore the running time strongly varies depending if the maps alignment step is implemented or not. As discussed in Section 6.1.1, if remaining historical buildings can be recovered from an existing 3D city model, this significantly reduces the processing time. In that way, it took about four hours to reconstruct the city of Delft in 1982 (10116 aligned buildings and 2958 2PM buildings) (Section 5.3.3). However, reconstructing the city of Brussels in 1700 entirely with the 2D procedural modelling process took up to fifteen hours (4481 2PM buildings) (Section 5.5.2). As the aim of this thesis was the automation, I do not think that the processing time is really an issue as it does not require any interventions from the users and they just have to wait for the results.

- **How much accurate are the historical 3D city models reconstructed?**

The methodology implemented in this research aims at reconstructing historical 3D city models that represent as much accurately as possible the past state of a city. The term accuracy is here a global term which encompasses:

— The positional accuracy: are the historic buildings modelled at their true location?

— The geometric accuracy: is the geometry of the historic building footprints properly recovered?

— The accuracy about the number of buildings: is there a one-to-one correspondence between the historic buildings and the 3D buildings in the historical 3D city model?

— The architectural accuracy : are the historic buildings modelled as they looked like in reality?

In this thesis, the building plots extraction method was developed in order to optimise the positional accuracy and the geometric accuracy of the building plots, as well as the accuracy about their number. To further ensure these types of accuracy at the individual building level, the one-to-many alignment process (Section 3.3.1) was used as it allows to recover the exact building footprints of historic buildings that still remain nowadays. As for the historic buildings that did not persist, they were recovered using the 2D procedural modelling process, based on assumptions and observations made from other datasets (Section 3.3.2). Lastly, for the architectural accuracy, educated guesses had to be made regarding the appearance of the buildings, as it is not possible to recover that information for all the buildings of a city. For this reason, a literature review supported the reconstruction of the LoD2 buildings (Section 3.4). In that way, the historical 3D city models reconstructed in this study were made to represent as much accurately as possible the past. Maximising the accuracy is not necessary for the visualisation-based use cases these historical 3D city models support, but it is of a great importance for the pragmatic applications they enable as these applications usually require geometric accuracy (Section 6.1.3).

An important thing to consider about representing accurately the past is that certain historical maps have high inaccuracies in geometry and chronometry, which leave us uncertain about the historical situation of a city [de Boer, 2010]. As a result one historical 3D city model might stick perfectly to the input historical map but could still be inaccurate when representing the reality because the historical map does not depict the exact location of the buildings or other geographic features. Therefore the older the historical maps, the less the reconstructed historical 3D city models are likely to be accurate as the historical maps are more likely to contain inaccuracies. Nevertheless, historical maps are often the only source of information about the pre-satellites and digital maps era and there is thus no other choice than accepting these potential inaccuracies [Liu et al., 2019].

- **In what ways and for which applications are the simple historical 3D city models obtained relevant in comparison to highly detailed manually derived historical 3D city models?**

Simple historical 3D city models actually support more diversified applications than highly detailed historical 3D city models. Highly detailed historical 3D city models are mainly used for visualisation purposes in the historical or in the gaming domains as they often contain too much details to be used in more practical applications. Besides, as highly detailed historical 3D city models are essentially built for these purposes, the focus is usually set on making it look realistic and less on being accurate in the reconstruction process. In contrast, as discussed in Section 6.1.3, simple historical 3D city models enable a large series of use cases from visualisation-based applications in the history and gaming domains to pragmatic applications in various other domains (e.g. demography, health or urban studies). Moreover, simple historical 3D city models automatically reconstructed can be used as a basis for reconstructing highly detailed ones.

Furthermore, one may wonder whether it is not better to have an error-proof simple 3D city model with low level geometries than a highly detailed 3D city model which is more likely to contain geometric errors due to the high level geometries or due to the numerous assumptions on which it relies. In this study, a robust methodology was developed such that the simple historical 3D city models reconstructed have their geometries valid at more than 99% (Section 5.4.2), which make them almost error-proof models. These simple historical 3D city models support a range of use cases which require low level geometries (Section 6.1.3), and they also fasten the computational time for conducting analyses in comparison to highly detailed historical 3D city models.

6.3 Contributions

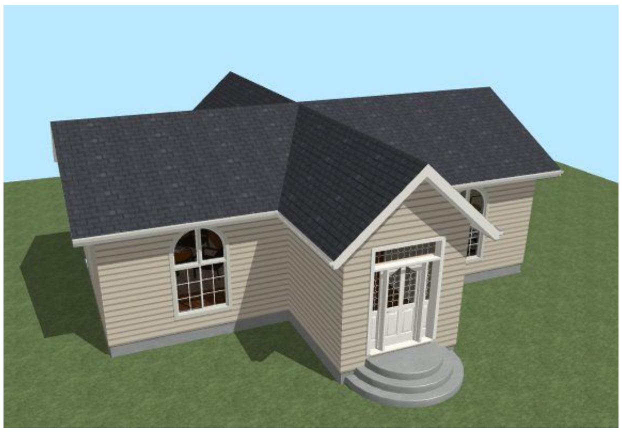
The contributions of this thesis are as follows:

- This study establishes one of the rare—if not the only one —literature review about historical 3D city modelling available in the state of the art.
- Different methods have been implemented and compared to derive the pros and cons of each method and the applications for which it is relevant. In that way, hints are provided about how to process a given input historical map to digitalise it or to reconstruct an historical 3D city model.
- This thesis provides a methodology for subdividing building plots into individual building footprints in a procedural way, which to my knowledge has not been investigated yet in the literature.
- The BCGA addon, which allows to procedurally generate LoD2 buildings, has been enriched with additional roof rules. This addon could be used in other contexts than historical 3D city modelling so as to reconstruct current 3D city models. This is even more interesting for cities of the Netherlands as crow-stepped gable roofs can be reconstructed automatically.
- This study provides a methodology to reconstruct automatically historical 3D city models from historical maps. Even if it does not work directly for all input historical maps, the methodology can be easily adapted for other historical maps as the digitalisation method implemented can be easily replaced with another one from Section 5.2.1

6.4 Future work

This thesis was mainly an experimental work to determine what can be done with historical maps in the domain of historical 3D city modelling, and what cannot. The main steps of the methodology implemented—i.e. building plots extraction, reconstruction of individual building footprints and LoD2 buildings generation—could be, in themselves, a thesis topic. Therefore each of this step could be further improved to derive a better methodology and produce better results. Several improvements are proposed:

- **Improved building plots subdivision into building footprints**: The decomposition of building plots into building footprints relies on two main processes—maps alignment and 2D procedural modelling—which could be both improved. As discussed in section Section 6.1.2, the maps alignment process has limits when applied with historical maps with high geometric inaccuracies, as the same buildings do not always have the same location in the current and in the historical maps. To further solve this issue, other georeferencing methods for historical maps should be investigated. For instance, some research suggests that kernel-based methods produce better results than global and local transformations with historical maps [Follin et al., 2016; Herrault et al., 2013a]. Besides, it is very likely that different collections of historical maps actually require different georeferencing methods, as the same georeferencing transformations applied on different map collections yielded different results in different studies [Follin et al., 2016; Herrault et al., 2013a; Király et al., 2008]. As for the 2D procedural modelling process for subdividing building plots, the method implemented tends to create regular building footprints but building extensions are not created. This was spotted as the main difference between the building footprints generated with procedural modelling and the ones from the current building footprints dataset. This could further be improved in the future. Additionally, a better clean-up could be implemented to avoid any overlap between building footprints.
- **Addition of procedural modelling rules**: The BCGA addon could be indefinitely extended with new BCGA rules. In particular, it would be interesting to add a roof rule for reconstructing cross hipped/gable roofs as most of the current roof rules cannot be applied on concave building footprints (Figure 6.2a). In addition, this thesis focused on reconstructing a certain type of Dutch roofs—i.e. crow-stepped gable roofs—but Dutch gable roofs are also widely spread in the Netherlands and it could be interesting to create a procedural modelling rule to reconstruct them automatically (Figure 6.2b). Especially, as this roof type contains curved surfaces it could be a first step towards the reconstruction of other types of curved roofs.


(a) Example of cross gable roof. Figure from Home Designer® Software [2019].

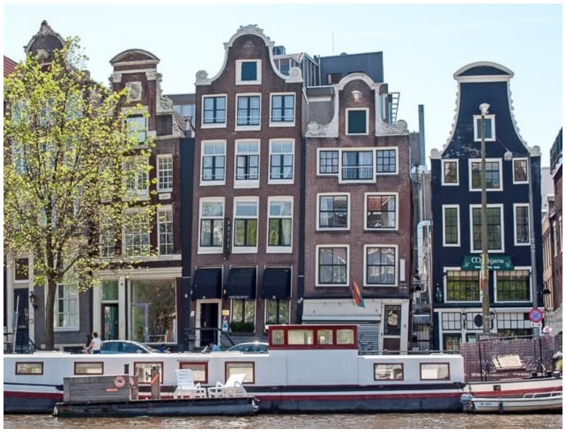
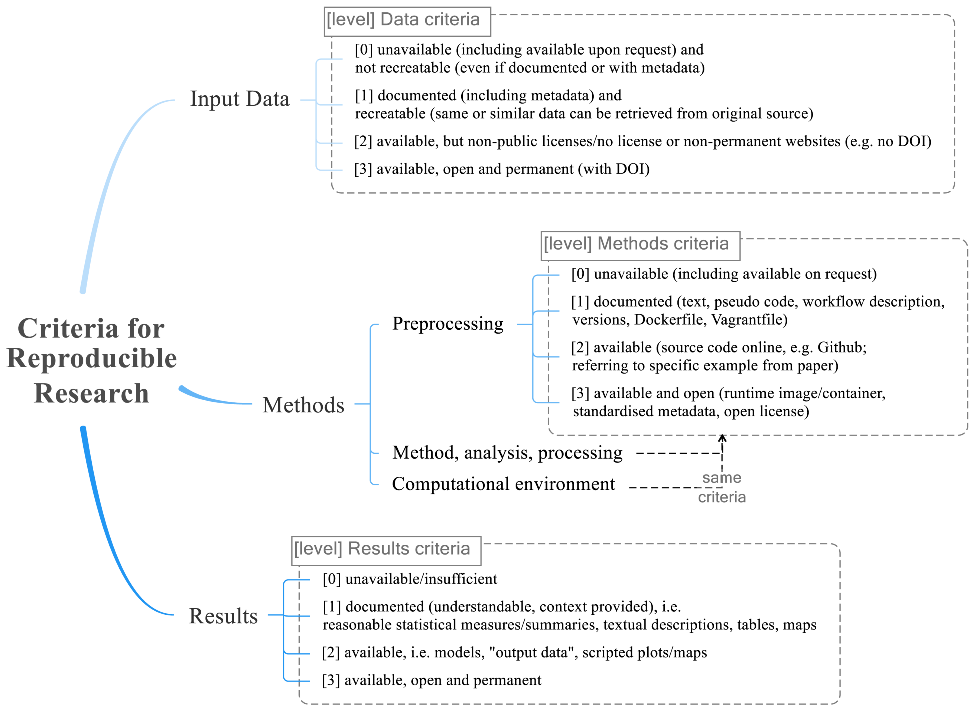

(b) Example of Dutch gable roofs in Amsterdam. Figure from Jones [nd].

Figure 6.2: Additional types of roofs that could be reconstructed with 3D procedural modelling

- **Use of additional historical sources**: This research focused on reconstructing historical 3D city models from historical maps as only historical source. Using other types of historical sources in addition to historical maps could have added value. In particular, as discussed in Section 6.1.2, one downside of the complete methodology version is that it requires aligned building footprints enriched with height attributes as input for inferring the roof height. This step of the methodology could be replaced by extracting the roof height from other historical sources. For instance, machine learning techniques could be used to extract building height from historical aerial images [Lánský, 2020]. Some research has also used historical photographs or postcards to recover the building height at the building level [Hadjiprocopis et al., 2014; Kersten et al., 2012]. Furthermore, using additional historical sources could also help recover more accurately the true appearance of the historic buildings in the historical 3D city models. For instance, historical paintings or photographs could be used to derive statistics about the types of roofs that were more commonly used at certain time periods [Isoda et al., 2009]. Discussions with historians and other researchers in that domain could also help recover that information at the building level. Indeed, historians can usually make specific hypothesis about the type of roof that was used for a given building by inferring it from the shape of the building footprint and the city area. In that way, the accuracy of the historical 3D city models reconstructed could greatly be improved by doing some interdisciplinary work.

- **Extension to other input historical maps**: The methodology implemented relies on multicolored historical maps as input. As only the map digitalisation step depends on the type of input historical map, the methodology could be further extended to monochromatic historical maps by adding a decision tree in the workflow and implementing different maps digitalisation methods based on the characteristics of the input historical map (monochromatic vs. multicolored).
- **3D reconstruction of other features**: This research focused on reconstructing historical 3D city models made of buildings, but it would also be interesting to automate the reconstruction of other types of features such as canals, roads and trees. First, extracting other features from the historical maps could also facilitate the reconstruction of the buildings as constraints could be defined not to reconstruct buildings at the location of canals or roads. Extracting the roads could also ease the identification of the street side of the buildings in the 3D procedural modelling step (Section 3.4.2). In addition, reconstructing other types of features could also lead to additional applications. For instance, in medical mapping, the reconstruction of canals or rivers could provide additional insights on the propagation of waterborne diseases. Moreover, the reconstruction of trees and green areas could also be useful in historical ecology. However, research has shown that extracting different types of features from historical maps requires different extraction methods [Herrault et al., 2013b]. This would thus require testing and implementing other methods than the ones investigated in this thesis to further enrich this thesis workflow.

A Reproducibility self-assessment



Criteria for Reproducible Research
Input Data
[level] Data criteria
[0] unavailable (including available upon request) and not recreatable (even if documented or with metadata)
[1] documented (including metadata) and recreatable (same or similar data can be retrieved from original source)
[2] available, but non-public licenses/no license or non-permanent websites (e.g. no DOI)
[3] available, open and permanent (with DOI)
Methods
Preprocessing
[level] Methods criteria
[0] unavailable (including available on request)
[1] documented (text, pseudo code, workflow description, versions, Dockerfile, Vagrantfile)
[2] available (source code online, e.g. Github; referring to specific example from paper)
[3] available and open (runtime image/container, standardised metadata, open license)
Method, analysis, processing
Computational environment
same criteria
Results
[level] Results criteria
[0] unavailable/insufficient
[1] documented (understandable, context provided), i.e. reasonable statistical measures/summaries, textual descriptions, tables, maps
[2] available, i.e. models, "output data", scripted plots/maps
[3] available, open and permanent

Figure A.1: Reproducibility criteria. Figure from [Nüst et al., 2018].

Criteria	Score	Additional comments
Input data	2-3	Available from public websites but without DOI
Preprocessing	1	Documented
Methods	2	Source code online from Github
Computational environment	3	Open source softwares
Results	2	Models available online from Github

Table A.1: Reproducibility criteria and their score with respect to this research. Additional comments are made regarding the scores given. Github repository: https://github.com/CamilleMorlighem/histo3d.

B Additional tables

Roof type	4-sided building footprint		n-sided building footprint	
	Built before 1915	Built after 1915	Built before 1915	Built after 1915
Flat roof	10	25	60	60
Hip roof	20	25	–	–
Gable roof	20	25	–	–
Mansard roof	10	10	40	40
Crow-stepped gable roof	40	15	–	–

Table B.1: Probabilities of roof types used to reconstruct the historical 3D city models of Delft and Brussels. The probabilities are based on both a literature review (Section 3.4.2) and the capabilities of the 3D procedural modelling algorithm (i.e. fails to reconstruct cross hipped roofs). The values are expressed in percentages.

Historical map	Ground truth data points	Ground truth building plots	Training data points	Number of classes
Delft 1880	172	50	108	11
Delft 1915	496	50	101	9
Delft 1961	397	50	59	7
Delft 1982	390	50	52	7
Brussels 1700	95	15	66	8
Brussels 1890	741	15	189	9
Brussels 1924	688	15	112	8

Table B.2: Amount of ground truth data and training data that were needed to conduct this research, along with the number of classes present in each historical map

Step	**Parameter**	**Delft 1880**	**Delft 1915**	**Delft 1961**	**Delft 1982**	**Units**
Segmentation	Dissimilarity threshold range	**0.07–0.1**	0.07–0.1	0.07–0.1	0.07–0.1	–
	Min. segment size range	**7–10**	7–10	7–10	7–10	pixels
Generalisation workflow	α parameter	30	30	12	12	–
	Distance threshold	5	5	5	5	m
	Angle threshold	180	180	180	180	°
1:n alignment	Buffer size	6	4	**10**	10	m
	Overlap threshold	0.9	0.9	**0.6**	0.6	–
2D procedural modelling	Facade length range	5	5	5	5	m
	Facade depth range	12–16	12–16	12–16	12–16	m
	Area threshold	100	100	100	100	m^2
Reference heights assignment	Ground height	$\boldsymbol{ground_{0.20}}$	$ground_{0.20}$	$ground_{0.20}$	$ground_{0.20}$	m
	Building height (LoD1)	$\boldsymbol{roof_{0.25}}$	$roof_{0.25}$	$roof_{0.25}$	$roof_{0.25}$	m
	Building height (LoD2)	$\boldsymbol{roof_{0.99}}$	$roof_{0.99}$	$roof_{0.99}$	$roof_{0.99}$	m

Table B.3: Overview of the user-defined parameters of the complete methodology version and their values for all Delft historical maps. Green indicates the parameters that can be optionally defined by the user to replace the default values defined in this study through experimentation. The default values are in bold.

Step	**Parameter**	**Brussels 1700**	**Brussels 1890**	**Brussels 1924**	**Units**
Segmentation	Dissimilarity threshold range	**0.07–0.1**	0.07–0.1	0.07–0.1	–
	Min. segment size range	**7–10**	7–10	7–10	pixels
Generalisation workflow	α parameter or Shapely tolerance	Tolerance = 12	α = 30	α = 25	–
	Distance threshold	–	10	12	m
	Angle threshold	–	150	150	°
2D procedural modelling	Facade length range	18–25	8–15	8–15	m
	Facade depth range	35–40	23–30	23–30	m
	Area threshold	300	300	300	m^2
Reference heights assignment	Ground height	$\boldsymbol{ground_{0.20}}$	$ground_{0.20}$	$ground_{0.20}$	m
	Building height (LoD1)	6.7–17	6.7–17	6.7–17	m
	Attic height	2.6–6	2.6–6	2.6–6	m

Table B.4: Overview of the user-defined parameters of the shortened methodology version and their values for all Brussels historical maps. Green indicates the parameters that can be optionally defined by the user to replace the default values defined in this study through experimentation. The default values are in bold.

Delft 1880	**Vectorisation**	**1st generalisation**	**2nd generalisation**
Avg. area (m)	1965.218	2219.318	2300.296
Med. area (m)	506.49	683.645	754.066
Avg. perimeter (m)	305.565	186.34	202.793
Med. perimeter (m)	197	122	141.5
Med. number of vertices	75	30	5
Avg. compactness	0.221	0.63	0.535
Avg. convexity	0.663	0.994	0.996
Concave building plots	191	191	48
Med. rectangularity	0.558	0.749	0.715
Number of building plots	191	191	184
Delft 1915	**Vectorisation**	**1st generalisation**	**2nd generalisation**
Avg. area (m)	2217.148	2442.131	2497.281
Med. area (m)	772.334	851.537	886.026
Avg. perimeter (m)	277.78	181.663	195.933
Med. perimeter (m)	169.5	126	139.5
Med. number of vertices	73.5	31	5
Avg. compactness	0.343	0.714	0.6
Avg. convexity	0.732	0.994	0.996
Concave building plots	300	300	57
Med. rectangularity	0.686	0.803	0.781
Number of building plots	300	300	298
Delft 1961	**Vectorisation**	**1st generalisation**	**2nd generalisation**
Avg. area (m)	4015.915	3934.542	4053.362
Med. area (m)	2631.321	2637.864	2871.968
Avg. perimeter (m)	376.191	255.013	268.772
Med. perimeter (m)	341	237	258
Med. number of vertices	248	63	5
Avg. compactness	0.305	0.612	0.562
Avg. convexity	0.706	0.984	0.99
Concave building plots	377	377	126
Med. rectangularity	0.737	0.788	0.773
Number of building plots	377	377	366
Delft 1982	**Vectorisation**	**1st generalisation**	**2nd generalisation**
Avg. area (m)	3905.248	3756.333	3880.068
Med. area (m)	2687.729	2595.399	2745.243
Avg. perimeter (m)	397.658	250.087	261.746
Med. perimeter (m)	362	236	257.5
Med. number of vertices	246.5	61	5
Avg. compactness	0.267	0.608	0.569
Avg. convexity	0.672	0.984	0.992
Concave building plots	366	368	110
Med. rectangularity	0.719	0.804	0.821
Number of building plots	366	368	358

Table B.5: Metrics computed for Delft historical building plots after (1) the vectorisation step, (2) the first generalisation step (α-shape generalisation) and (3) the second generalisation step (generalisation of Commandeur [2012]). avg. = average and med. = median.

Brussels 1700	**Vectorisation**	**1st generalisation**	**2nd generalisation**
Avg. area (m)	51869.925	51311.8	–
Med. area (m)	44344.31	44151.72	–
Avg. perimeter (m)	1992.537	1449.269	–
Med. perimeter (m)	1374.5	984.5	–
Med. number of vertices	415	8	–
Avg. compactness	0.281	0.498	–
Avg. convexity	0.67	0.896	–
Concave building plots	108	76	–
Med. rectangularity	0.684	0.708	–
Number of building plots	108	108	–
Brussels 1890	**Vectorisation**	**1st generalisation**	**2nd generalisation**
Avg. area (m)	16153.19	16649.936	16762.203
Med. area (m)	10907.135	10966.752	11072.548
Avg. perimeter (m)	1080.68	517.848	526.144
Med. perimeter (m)	812	464	474
Med. number of vertices	308	54	6
Avg. compactness	0.202	0.65	0.624
Avg. convexity	0.568	0.986	0.994
Concave building plots	699	697	289
Med. rectangularity	0.674	0.76	0.77
Number of building plots	699	697	687
Brussels 1924	**Vectorisation**	**1st generalisation**	**2nd generalisation**
Avg. area (m)	11080.239	10890.534	11198.124
Med. area (m)	7128.107	7035.763	7429.33
Avg. perimeter (m)	655.754	406.449	438.862
Med. perimeter (m)	544	358.5	394.5
Med. number of vertices	187	55.5	5
Avg. compactness	0.311	0.7	0.607
Avg. convexity	0.672	0.99	0.995
Concave building plots	633	631	177
Med. rectangularity	0.712	0.771	0.732
Number of building plots	633	632	616

Table B.6: Metrics computed for Brussels historical building plots after (1) the vectorisation step, (2) the first generalisation step (α-shape generalisation or Shapely simplification algorithm) and (3) the second generalisation step (generalisation of Commandeur [2012]) if implemented. avg. = average and med. = median.

Bibliography

Ares Oliveira, S., di Lenardo, I., and Kaplan, F. (2017). Machine Vision Algorithms on Cadaster Plans.

Arkin, E. M., Chew, L. P., Huttenlocher, D. P., Kedem, K., and Mitchell, J. S. B. (1991). An efficiently computable metric for comparing polygonal shapes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(3):209–216.

Arroyo Ohori, K., Ledoux, H., and Peters, R. (2020). *3D modelling of the built environment: Lesson 4.1, Semantic 3D city models*.

Arteaga, M. G. (2013). Historical Map Polygon and Feature Extractor. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on MapInteraction*, MapInteract '13, page 66–71, New York, NY, USA. Association for Computing Machinery.

Balletti, C. and Guerra, F. (2016). Historical Maps for 3D Digital Cities History. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 51:115 – 126.

Bayazit, M. (2009). Poly Decomp.

Biljecki, F., Arroyo Ohori, K., Ledoux, H., Peters, R., and Stoter, J. (2016a). Population Estimation Using a 3D City Model: A Multi-Scale Country-Wide Study in the Netherlands. *PloS one*, 11:e0156808.

Biljecki, F., Ledoux, H., Du, X., Stoter, J., Soon, K., and Khoo, V. (2016b). THE MOST COMMON GEOMETRIC AND SEMANTIC ERRORS IN CITYGMLDATASETS. *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, pages 13–22.

Biljecki, F., Ledoux, H., and Stoter, J. (2016c). An improved LOD specification for 3D building models. *Computers, Environment and Urban Systems*, 59:25–37.

Biljecki, F., Stoter, J., Ledoux, H., Zlatanova, S., and Çöltekin, A. (2015). Applications of 3D City Models: State of the Art Review. *ISPRS International Journal of Geo-Information*, 4(4):2842–2889.

Bondright Roofing Services (n.d.). BONDRIGHT ROOFING GUIDE TO RESIDENTIAL ROOFING TYPES. http://bondrightroofing.co.uk/bondright-roofing-guide-to-residential-roofing-types.php.

Breiman, L. (2001). Random Forests. *Machine Learning*, 45(1):5–32.

Bshouty, E., Shafir, A., and Dalyot, S. (2019). Towards the generation of 3D OpenStreetMap building models from single contributed photographs. *Computers, Environment and Urban Systems*, 79:101421.

Budig, B., van Dijk, T. C., Feitsch, F., and Arteaga, M. G. (2016). Polygon Consensus: Smart Crowdsourcing for Extracting Building Footprints from Historical Maps. In *Proceedings of the 24th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, SIGSPACIAL '16, New York, NY, USA. Association for Computing Machinery.

Commandeur, T. (2012). Footprint decomposition combined with point cloud segmentation for producing valid 3D models.

de Boer, A. (2010). Processing old maps and drawings to create virtual historic landscapes.

de Pange, I. (2004). *Histoire de l'architecture à Saint-Gilles. Inventaire du Patrimoine architectural de la Région de Bruxelles-Capitale.*

Douglas, D. H. and Peucker, T. (1973). ALGORITHMS FOR THE REDUCTION OF THE NUMBER OF POINTS REQUIRED TO REPRESENT A DIGITIZED LINE OR ITS CARICATURE. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 10:112–122.

Drolias, G. C. and Tziokas, N. (2020). Building Footprint Extraction from Historic Maps utilizing Automatic Vectorisation Methods in Open Source GIS Software.

Dukai, B. (2018). BAG 3D. `https://tudelft3d.github.io/bag3d/intro.html`.

Dukai, B., Ledoux, H., and Stoter, J. (2019). A MULTI-HEIGHT LOD1 MODEL OF ALL BUILDINGS IN THE NETHERLANDS. *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, IV-4/W8:51–57.

Ebi, N., Lauterbach, B., and Anheier, W. (1994). An image analysis system for automatic data acquisition from colored scanned maps. *Machine Vision and Applications*, 7(3):148–164.

Espindola, G., Câmara, G., Reis, I., Leonardo, B., and Monteiro, A. (2006). Parameter selection for region-growing image segmentation algorithms using spatial autocorrelation. *International Journal of Remote Sensing - INT J REMOTE SENS*, 27:3035–3040.

Fan, H., Zipf, A., Fu, Q., and Neis, P. (2014). Quality assessment for building footprints data on OpenStreetMap. *International Journal of Geographical Information Science*, 28(4):700–719.

Fernandes, F. M. d. C., Martins, E. d. S., Pedrosa, D. M. A. S., and Evangelista, M. d. S. N. (2017). Relationship between climatic factors and air quality with tuberculosis in the Federal District, Brazil, 2003-2012. *The Brazilian journal of infectious diseases : an official publication of the Brazilian Society of Infectious Diseases*, 21:369–375.

Follin, J.-M., Fahrasmane, M., and Simonetto, E. (2016). An open-source based toolchain for the georeferencing of old cadastral maps.

Frischer, B., Abernathy, D., Guidi, G., Myers, J., Thibodeau, C., Salvemini, A., Müller, P., Hofstee, H., and Minor, B. (2008). Rome Reborn. page 34.

Galanaud, P., Galanaud, A., and Giraudoux, P. (2015). Historical Epidemics Cartography Generated by Spatial Analysis: Mapping the Heterogeneity of Three Medieval "Plagues" in Dijon. *PloS one*, 10:e0143866.

Gholami, R. and Fakhari, N. (2017). Chapter 27 - Support Vector Machine: Principles, Parameters, and Applications. In Samui, P., Sekhar, S., and Balas, V. E., editors, *Handbook of Neural Computation*, pages 515–535. Academic Press.

Gobbi, S., Ciolli, M., La Porta, N., Rocchini, D., Tattoni, C., and Zatelli, P. (2019). New Tools for the Classification and Filtering of Historical Maps. *ISPRS International Journal of Geo-Information*, 8:455.

Guidi, G., Frischer, B., and Lucenti, I. (2012). Rome Reborn - Virtualizing the ancient imperial Rome.

Guidi, G. and Russo, M. (2011). Diachronic 3D reconstruction for lost Cultural Heritage. volume XXXVIII-5/W16.

Gurung, K. (2017). *Fractal Dimension in Architecture: An Exploration of Spatial Dimension*. PhD thesis.

Hadjiprocopis, A., Ioannides, M., Wenzel, K., Rothermel, M., Johnsons, P. S., Fritsch, D., Doulamis, A. D., Protopapadakis, E., Kyriakaki, G., Makantasis, K., Weinlinger, G., Klein, M., Fellner, D., Stork, A., and Santos, P. (2014). 4D reconstruction of the past: the image retrieval and 3D model construction pipeline. In *International Conference on Remote Sensing and Geoinformation of Environment*.

Heitzler, M. and Hurni, L. (2020). Cartographic reconstruction of building footprints from historical maps: A study on the Swiss Siegfried map. *Transactions in GIS*, 24(2):442–461.

Held, M. and Palfrader, P. (2017). Straight skeletons with additive and multiplicative weights and their application to the algorithmic generation of roofs and terrains. *Computer-Aided Design*, 92:33–41.

Henderson, T., Linton, T., Potupchik, S., and Ostanin, A. (2009). Automatic Segmentation of Semantic Classes in Raster Map Images. *8Th IAPR International Workshop on Graphics Recognition*, 2009.

Herrault, P.-A., Sheeren, D., Fauvel, M., Monteil, C., and Paegelow, M. (2013a). A comparative study of geometric transformation models for the historical 'Map of France' registration. *Geographia Technica*.

Herrault, P.-A., Sheeren, D., Fauvel, M., and Paegelow, M. (2013b). Automatic Extraction of Forests from Historical Maps Based on Unsupervised Classification in the CIELab Color Space. In *AGILE Conf.*

Home Designer® Software (2019). Creating a Cross Gable Roof. https://www.homedesignersoftware.com/support/article/KB-01043/creating-a-cross-gable-roof.html.

Ignjatić, J., Nikolic, B., and Rikalovic, A. (2018). Deep learning for historical cadastral maps digitization: overview, challenges and potential.

Isoda, Y., Tsukamoto, A., Kosaka, Y., Okumura, T., Sawai, M., Yano, K., Nakata, S., and Tanaka, S. (2009). Reconstruction of Kyoto of the Edo Era Based on Arts and Historical Documents: 3D Urban Model Based on Historical GIS Data. *International Journal of Humanities and Arts Computing*, 3:21–38.

Jones, S. (n.d.). Leaning Houses in Amsterdam. https://thetravelbunny.com/canal-leaning-houses-amsterdam/.

Kersten, T. P., Keller, F., Saenger, J., and Schiewe, J. (2012). Automated Generation of an Historic 4D City Model of Hamburg and Its Visualisation with the GE Engine. In Ioannides, M., Fritsch, D., Leissner, J., Davies, R., Remondino, F., and Caffo, R., editors, *Progress in Cultural Heritage Preservation*, pages 55–65, Berlin, Heidelberg. Springer Berlin Heidelberg.

Kim, D.-S. (1998). Polygon offsetting using a Voronoi diagram and two stacks. *Computer-Aided Design*, 30(14):1069–1076.

Király, G., Walz, U., Podobnikar, T., Czimber, K., Neubert, M., and Kokalj, (2008). *Georeferencing of historical maps – methods and experiences*, pages 53–63.

Lánský, I. (2020). Height Inference for all USA Building Footprints in the Absence of Height Data. Master's thesis, Delft University of Technology.

Laycock, S., Brown, P., Laycock, R., and Day, A. (2011). Aligning archive maps and extracting footprints for analysis of historic urban environments. *Computers Graphics*, 35(2):242 – 249. Virtual Reality in Brazil Visual Computing in Biology and Medicine Semantic 3D media and content Cultural Heritage.

LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep Learning. *Nature*, 521:436–44.

Ledoux, H., Arroyo Ohori, G., and Peters, R. (2020). *Computational modelling of terrains*.

Lee, W. (2019). *Supervised Learning—Classification Using K-Nearest Neighbors (KNN)*, pages 205–220.

Liu, C., Wu, J., Kohli, P., and Furukawa, Y. (2017). Raster-to-Vector: Revisiting Floorplan Transformation. *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 2214–2222.

Liu, T., Xu, P., and Zhang, S. (2019). A review of recent advances in scanned topographic map processing. *Neurocomputing*, 328:75 – 87. Chinese Conference on Computer Vision 2017.

Maiwald, F., Henze, F., Bruschke, J., and Niebling, F. (2019). GEO-INFORMATION TECHNOLOGIES FOR A MULTIMODAL ACCESS ON HISTORICAL PHOTOGRAPHS AND MAPS FOR RESEARCH AND COMMUNICATION IN URBAN HISTORY. *ISPRS - International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, XLII-2/W11:763–769.

McLeod, K. S. (2000). Our sense of Snow: the myth of John Snow in medical geography. *Social Science Medicine*, 50(7):923 – 935.

Mesqui, J. (1998). Denis Rolland, Architectures rurales en Picardie. le Soissonnais. Editions Créer, Nonette, 1998.

Müller, P., Wonka, P., Haegler, S., Ulmer, A., and Van Gool, L. (2006). Procedural Modeling of Buildings. *ACM Trans. Graph.*, 25(3):614–623.

Nijhuis, S., van lammeren, R., and van der Hoeven, F. (2011). *Exploring the Visual Landscape. Advances in Physiognomic Landscape Research in the Netherlands*.

Nobajas, A. and Nadal, F. (2015). From historical map to online 3D recreation: the 1861 cadastral map of Horta (Barcelona). *Cartography and Geographic Information Science*, 42(3):211–223.

Nüst, D., Granell, C., Hofer, B., Konkol, M., Ostermann, F., Sileryte, R., and Cerutti, V. (2018). Reproducible research and GIScience: An evaluation using AGILE conference papers. *PeerJ*, 6:e5072.

Oliveira, S. A., Kaplan, F., and Lenardo, I. D. (2017). Machine Vision Algorithms On Cadaster Plans. In *DH*.

Parish, Y. I. H. and Müller, P. (2001). Procedural Modeling of Cities. In *Proceedings of the 28th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '01, page 301–308, New York, NY, USA. Association for Computing Machinery.

Pezeshk, A. and Tutwiler, R. (2008). Contour Line Recognition Extraction from Scanned Colour Maps Using Dual Quantization of the Intensity Image. pages 173 – 176.

Reyes, O., Lee, E. C., Sah, P., Viboud, C., Chandra, S., and Bansal, S. (2018). Spatiotemporal Patterns and Diffusion of the 1918 Influenza Pandemic in British India. *American Journal of Epidemiology*, 187(12):2550–2560.

Riche, M. (2020). Identifying Building Footprints in Historic Map Data using OpenCV and PostGIS. pages 19–31.

Ryckaert, M. (2012). Bruges (Belgium): houses at the Potterierei. https://commons.wikimedia.org/wiki/File:Brugge_Houses_Potterierei_R02.jpg.

Samal, A., Seth, S., and Cueto1, K. (2004). A feature-based approach to conflation of geospatial sources. *International Journal of Geographical Information Science*, 18(5):459–489.

Stiny, G. (1980). Introduction to Shape and Shape Grammars. *Environment and Planning B*, 7(3):343–351.

Sun, K., Hu, Y., Song, J., and Zhu, Y. (2020). Aligning geographic entities from historical maps for building knowledge graphs. *International Journal of Geographical Information Science*, 0(0):1–30.

Suzuki, S. and Chikatsu, H. (2003). RECREATING THE PAST CITY MODEL OF HISTORICAL TOWN KAWAGOE FROM ANTIQUE MAP.

Tong, X., Shi, W., and Deng, S. (2009). A probability-based multi-measure feature matching method in map conflation. *International Journal of Remote Sensing*, 30(20):5453–5472.

Uhl, J. H., Leyk, S., Chiang, Y.-Y., Duan, W., and Knoblock, C. A. (2017). Extracting Human Settlement Footprint from Historical Topographic Map Series Using Context-Based Machine Learning.

Ullrich, T., Schinko, C., and Fellner, D. (2010). Procedural modeling in theory and practice. *Fraunhofer IGD*.

Urbanik, J. and Tomaszewicz, A. (2014). Flat Roof - Advantage or Disadvantage of Modern Movement Buildings.

Walter, V. and Fritsch, D. (1999). Matching spatial data sets: A statistical approach. *International Journal of Geographical Information Science*, 13:445–473.

Wirth, M. (2004). Shape Analysis and Measurement.

Xavier, E. M. A., Ariza-López, F. J., and Ureña Cámara, M. A. (2016). A Survey of Measures and Methods for Matching Geospatial Vector Datasets. *ACM Comput. Surv.*, 49(2).

Ying, S., Li, L., Gao, Y., and Min, Y. (2011). Probabilistic matching of map objects in multi-scale space. In *Proceedings of the 25th International Cartographic Conference*.

Younes, L., Romaniuk, B., Desjardin, , and Bittar, E. (2013). Reconstruction 3D de bâtiments à partir de cartes postales anciennes. vers un SIG 4D collaboratif.

Colophon

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