Automated vectorization of historical maps using state-of-the-art Computer Vision techniques

Master's thesis
in partial fulfilment of the requirements for the degree of
Master of Science

University of Münster
Computer Science Department

Supervisor:
Prof. Benjamin Risse

First assessor:
Prof. Benjamin Risse

Second assessor:
Prof. Xiaoyi Jiang

Submitted by:
Mhd Sufian Zaabalawi

Münster, November 2020

Automated vectorization of historical maps using state-of-the-art Computer Vision techniques

---

Automatisierte Vektorisierung von Urkarten mithilfe moderner Computer Vision Techniken

Abstract

Many researchers have been interested in historical and geographical analysis over the years. Particularly in the field of historical research, the digitization of historical documents has become increasingly important, which facilitates the ability to compare and process a large number of historical documents. In this thesis, a pipeline is introduced to automate the raster-to-vector process of historical cadastral map images, which consists of transforming raster historical cadastral map into vector graphic format. The pipeline initiates a solution to solve this problem using a combination of modern and traditional Machine Learning techniques. It has been tested in detail on different cadastral maps and it achieves a modular, extensible application with satisfactory results. The results conclude that the pipeline geometrically approximates 79% of the parcels as polygons with an IoU score higher than 0.8 and classifies different map objects with an accuracy of 90% using Feature Engineering. Finally, a web application with a user-friendly interface is implemented, which provides interactive visualization and a simple control panel for the parameters used in the pipeline.

Contents

1 Introduction

Historical research has always been suffering from the time-consuming process of searching for sources within archives. With the digitization of historical documents; however, this concept is changing. The digitization of archive maps has been gaining more and more importance in the last decades, particularly, in the field of historical research, which requires the ability to compare and process large numbers of historical documents.
The digitization of historical documents is a challenging task since the data may include various information types such as text, drawings and maps. Generally, historical documents are digitized by scanning and preserving them in a digital archive as raster images.

1.1. Motivation

Cadastral maps are one of the most challenging historical data encountered in digitization. A cadastral map is a map that describes the boundaries and ownership of land parcels. It has the benefit of presenting the spatial relationships between features illustrated on the map such as location and buildings shape in addition to ownership and parcel values. Moreover, historical maps encapsulate changes over time on the environment and the urban development in a certain area. Therefore, it is convenient to transform the documents into structured and compact data such as digital characters for text or vector graphics for cadastral maps in order to facilitate the processing of historical documents. The digitization of historical cadastral maps allows easier transfer of data as well as conducting further analysis on a large number of maps. Although the problem of digitization of historical documents was solved by redrawing or rewriting the documents, the process is still resource-intensive, laborious and varies depending on the document being digitized. In the recent years, there has been a growing interest in applying modern Computer Vision techniques in

order to solve such problems, which allowed the field of digital history to gain rapid development. Yet, very few approaches have been dealing with the vectorization of historical cadastral maps using automated algorithms including Computer Vision and Machine Learning.

1.2. Problem

The problem can be partially solved by archiving maps as images; however, it adds some new concerns. Raster images are composed of a grid of pixels, which cannot be scaled without loss of resolution. Moreover, the grid representation of the raster images can produce inaccuracies in spatial and spectral information. Since raster images consist of millions of pixels, processing, altering, and transferring the images require high computational power. Therefore, vectorizing cadastral maps is the key for solving these issues. Vectorization means that the maps are redrawn in a geometrical form, i.e. in polygons and lines with points. The main advantage of vector graphic data that it is lightweight, scalable for arbitrary size, and can be analyzed interactively. This also known as a raster-to-vector conversion. Nonetheless, the redrawing of cadastral maps into vector graphics is a labor-intensive process which also slows down the process drastically, when dealing with hundreds or thousands of archived maps.

Therefore, the main objective is to obtain a vectorization pipeline, that is capable of automatically transforming cadastral map images into vector graphic format. In general, historical documents exhibit poor ink quality, paper degradation and aging artifacts. In addition to the previous challenges, historical cadastral maps present low-quality contours, overlaying parcel annotations in addition to the large amount of data contained within the map. Moreover, different maps use different annotation styles, different fonts and assorted colors which makes it even more challenging to automate the vectorization process and extract their spatial features. As seen in Fig. 1.1, the historical cadastral maps present large variations of geographic guidelines and different annotations, while some exhibit pale colors and incomplete contours. Therefore, the transformation of raster-to-vector historical cadastral maps

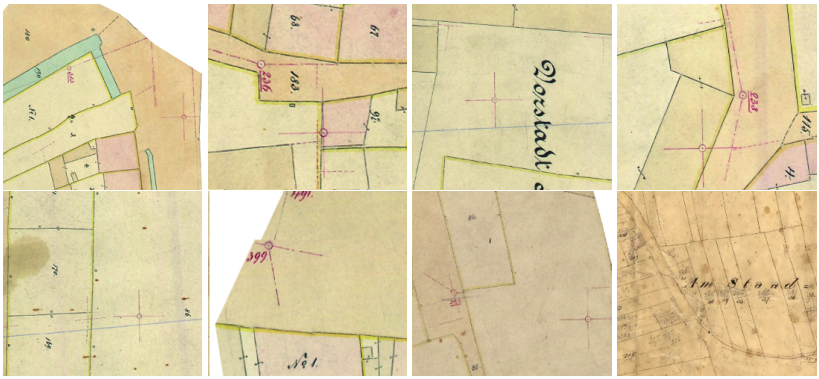


Figure 1.1.: Examples of the challenges of vectorization historical cadastral maps.

was considered until recently a beyond state-of-the-art problem [1].

As Computer Vision techniques have significantly improved in recent years, it became possible to automate the raster-to-vector process and to introduce a potential solution in order to expand the possibilities for research and to reduce labor-intensive work as well as time and error.

1.3. Overview

In this thesis, a cadastral map vectorization pipeline is introduced to automate the raster-to-vector process by applying modern and traditional Machine Learning techniques, which are discussed and compared in Chapter 2. The input of the pipeline is a historical cadastral map and the output is a GeoJSON file. Using Neural Networks in combination with a classical segmentation method, the pipeline is able to identify the regions in the map. Next, it refines and extracts the regions as polygons which are then classified using the Map Objects Classifier depending on spectral and shape features. Consequently, the pipeline is capable of extracting polygons and producing contours with adjustable thickness in vector graphic format. The structure of the pipeline and each of its steps are discussed thoroughly in Chapter 3. Finally, a comprehensive evaluation is conducted in Chapter 5, where the results conclude that the

pipeline offers a solution to the raster-to-vector problem, which could be extended and developed as noted lastly in Chapter 6.

2 Fundamentals

The main goal of image vectorization is to convert a raster image into a vector graphic that is compact, scalable, lightweight and easy to handle. Vector graphics are usually represented by points, lines, curves, polygons or regions since these representations can be drawn manually with tools for vector graphics. However, in the case of historical cadastral maps, manual map vectorization is labor- intensive and one cannot manage a large number of maps.

Typically, the process of map vectorization consists of many stages including preprocessing, object segmentation, classification, post-processing and finally extracting the vector representation. Despite the fact that Deep Learning and Computer Vision are widely used for several applications, there are very few approaches that manage to use modern Computer Vision techniques in historical cadastral map vectorization. This Chapter starts with an overview about map vectorization and recent work in this topic followed by a theoretical background about the segmentation and classification techniques that are used in this thesis.

2.1. Map Vectorization

Historical maps are considered a part of the world's heritage, since they portray characteristics associated with the human activity and culture that existed in an area at a specific period of time. Furthermore, the correlation of written history with historical maps gives a further elaboration about the environmental, cultural and developmental aspects of a specific area.
Many researchers in the field of cartography and history are interested in the geographic analysis of terrains and the development of landscapes and settlements because historical maps encapsulate changes on the environment, human behavior and human development over time. Therefore, digitization of historical maps is

gaining greater importance within the research community. Scanning and capturing historical maps as high-resolution digital images is the first step in digitization. Unfortunately, archived historical maps are generally available as scanned raster images, which hinders the possibility to perform further analysis, due to the inflexibility and static properties of raster images. In addition, the raster images require large memory capacity, which makes them difficult to transfer, share or revise for a wider range of research.
To overcome this difficulty, it is helpful to convert the raster images into a lightweight vectorized format. Vector data is particularly suitable for this task because of its lightweight and flexibility. They are easier to combine and analyze, which provides researchers and historians easier identification of spatial changes in a region over time. However, the archived raster images require a lot of effort to be transformed into a vectorized format if done by hand. Therefore, it would be much easier to have an algorithmic tool to perform the raster-to-vector conversion automatically or even semi-automatically.

2.2. Related work

There are a few related works in the field of vectorization of cadastral maps using Machine Learning techniques. Nevertheless, this problem has considerably regained more attention in the research community, even though it was complex to solve and considered much beyond a state-of-the-art problem [1].
In the earlier years, research on cadastral map vectorization was limited to the identification of characters and digits using Machine Learning, while extracting the lines and the polygons using traditional line detection approaches [1]. Later on, various methods were proposed to vectorize maps using Neural Networks by tracing lines and identifying joints, however they did not offer a significant improvement in performance [1]. Recently, this topic has even gained more interest as a generic Deep Learning architecture was designed to be applicable to segment many typologies of documents [2]. One of the first applications of a fully automated vectorization pipeline

is presented in some recent works[2], [3], which also proposed an emerging research field with the terminology of *cadastral computing*, that attempts to transform cadaster cities into *computational objects*. This approach consists of a segmentation module that predicts the contours, then uses watershed algorithm [4], and a transcription identifier module that extracts digits and identifies parcel numbers.

2.3. Geographic Data and Geographical Information Systems Applications

Over the last decades, geographic data has evolved significantly in terms of effectiveness, robustness and applicability. Remote sensing systems can collect a huge amount of geographic data, including high spatial, temporal and spectral resolutions. As technology continues to develop, more and more remote sensors have the computational capability to acquire data with higher resolution. However, the higher the quality of the data, the more computationally intensive it is, therefore, the demand for efficient and lightweight geographic data has increased.

One of the first initiatives in this matter was Shapefiles developed by Esri [1] as a geospatial vector data format [5]. It can describe the vector features spatially by using points, lines and polygons to represent, for example, rivers, streets and buildings. In addition, each feature vector can be assigned to specific properties and attributes such as name or type.

Afterwards, GeoJSON was introduced as an open standard format for representing simple geographical features, as well as non-spatial attributes [6]. It was built upon JavaScript-Object-Notation (JSON) data structure, which is lightweight and often used for serializing and transferring structured data over the internet.

Geographical Information Systems (GIS) are considered to be important tools for handling such vectorized data formats, which are widely used in various research areas. Beyond that, GIS applications are important tools in areas such as environmental planning, real estate, infrastructure locations, navigation systems and urban

[1] Environmental Systems Research Institute

infrastructure. They have been also used in historical map analysis, which has recently regained much more flexibility, usability and applicability.
GIS Applications provide more understanding of historical cadastral map information, where they can handle both static raster images and vectorized map representations. In addition, the number of historians using GIS in a relation to geospatial methods is growing rapidly, which would be reflected by an exponential increase in GIS-based historical studies in the coming years [7]. It provides useful tools for geographic analysis by combining multiple maps and modifying other types of geographic information. Furthermore, it allows the visualization of geographical patterns, aggregated data with different sizes and the integration of material from textual, cartographic and visual sources. Yet, the basic features of GIS can sometimes make the applicability to historical problems difficult, since the maps need to be geo-referenced and have a specific format.

2.4. Watershed Segmentation with Distance Transformation

Watershed segmentation is a region-based segmentation technique that has its origins in mathematical morphology [4]. It has been widely applied to a variety of medical image and autonomous driving segmentation tasks. In watershed segmentation the image is considered as a topographic landscape with ridges and valleys, which corresponds to minima and maxima of pixel intensities to define each region of the image and calculate the so-called seed points or markers. The seed points are used to mark regions, for instance, watershed segmentation requires at least one interior local minimum for each region in the image. The placement of the seed points is set for distinct regions which are either manually defined by a hand or determined automatically in other ways, e.g. by morphological operators. Hence, watershed segmentation is considered as a supervised or unsupervised region segmentation technique, based on the method used to place the markers. Having said that, using Machine Learning techniques to predict the markers for watershed segmentation is considered as a supervised segmentation because it imitates the process of manually

marking each object [8].
There are many different approaches for performing watershed segmentation such as Distance Transform Approaches, Gradient Methods and Marker Controlled Methods. The Distance Transform Approach is based on the calculation of the distance transformation matrix, i.e. the distance from each pixel to the next non-zero pixel. This approach is widely used in Computer Vision and image processing [9]. It is performed on binary images, where the intensity of the pixels is defined by either 0 or 1. Let I be a binary image where each pixel $I((x, y)) \in \{0, 1\}$, then an object $\mathbf{O}$ in the image is a set of pixels defined as follows in Eq. 2.1:

$$\mathbf{O} = \{(x, y) \mid I((x, y)) = 1\} \tag{2.1}$$

The set $\mathbf{O}$ is considered as 'foreground' with white pixels, while the 'background' consists of black pixels defined with the complement set of the foreground $\mathbf{O}^c$. The distance-transformed image is a gray-scale image and represents the distance with a gradient intensity to the nearest edge of each pixel. For instance, the distance transformation generates an image whose value in each pixel represents the shortest distance d to the background $\mathbf{O}^c$. The distance transformation function for pixel p is given as follow in Eq. 2.2:

$$D(p) := min\{d(p, q) \mid q \in \mathbf{O}^c\} = min\{d(p, q) \mid I(q) = 0\}, \tag{2.2}$$

where the map D is the transformation matrix of the binary image I.
There are several different types of distance transformations depending on which distance metric is used to determine the distance between pixels. In general, d refers to the Euclidean distance metric, given by:

$$d(p, q) = \sqrt{(p_x - q_x)^2 + (p_y - q_y)^2}. \tag{2.3}$$

The Euclidean distance transformation converts a binary image into a gray-scale image in which the topographic ridges correspond to lighter pixels and the valleys correspond to their inner areas with pixels of lower intensity. Finally, the watershed segmentation uses the information of distances as the markers in the image.

One of the main advantages of the watershed with distance transformation is the separation of connected components and poorly distinguishable objects. However, its main limitation is that if no binary image is provided the watershed distance transformation would be sensitive to noise and variations in brightness in the image, which leads to over- or under-segmentation of the image [4].

2.5. Neural Networks in image Segmentation

Deep Neural Networks have increasingly become the key solution for many applications, to achieve human-level accuracy in many image recognition tasks [10]. A neural network is a network of neurons that can be visualized as connected nodes. In general, Deep Neural Networks consist of an input layer, hidden layers and finally an output layer, where each layer contains a set of neurons. It is also called artificial Neural Networks, since it was inspired by the biological brain, where the connections of the biological neuron are modeled as mathematical weights. A positive weight signifies a certain neuron to be activated; a negative weight deactivates or inhibits the corresponding neuron; and a weight which equals zero indicates no connection between the two neurons. The inputs are modified by weight and added together, which is referred to as a linear combination. Learning algorithms estimate these weights iteratively by minimizing a given loss function which is defined in terms of the error between the ground truth and the outputs. Convolutional Neural Networks (CNNs) are similar to ordinary Neural Networks, where the learnable weights are kernels with a certain size. The CNNs use convolution operation in order to learn filters that extract specific information form input images. In contrary to Neural Networks, the convolution operation provides spatial invariance and localized extracted information, which makes a CNN invariant to object translation in image recognition and segmentation tasks.

Recently, image segmentation based on Machine Learning has become significantly popular. It has enabled the field of Computer Vision to make a rapid progress in the recent years. Depending on the problem, there is different set of approaches each

with its own benefits and effectiveness to interpret images. The coarsest type is the localization and classification of images. It consists of drawing a bounding box on an object in the image as a way of unrefined segmentation. One of its limitations is the incapability to identify more than one object per image. Object identification is another technique of segmentation, which overcomes the burden of identifying multiple objects in the image. The You Only Look Once (YOLO) method is one of the fast object identification algorithms, it can accomplish real-time object detection via a fixed-grid regression [11], which is commonly used in face detection and autonomous driving. While object Identification can localize an object in an image, still there is no information on how the exact object shape is located. Object Identification with a bounding box is a rough estimation of object location and can include background or other objects in the same bounding box. Therefore, Semantic Segmentation was introduced to give a more detailed segmentation and more informative localization of the object. It has become the key application in many image processing and Computer Vision applications. Semantic Segmentation is also referred to as pixel-level classification, since it classifies each pixel into a category, results in assigning each object to a corresponding class. Instance Segmentation on the other hand, is an improved Semantic Segmentation with the ability of identifying multiple instances of the same object. For example, in a model that segments and classifies cats, if the input is an image with multiple cats, the model is able to segment and identify each individual cat separately.

One of the first and most popular approaches within the medical field for Semantic Segmentation is the U-Net [10]. The U-Net is built with an end-to-end architecture of encoder and decoder to solve Semantic Segmentation tasks. It consists of several hidden layers and can be logically divided into two parts. The first part is an encoder that takes an image as input and computes its feature maps while downscaling the image, which results in a multi-feature representation of the image. The second is a decoder that classifies pixels-wise the multi-feature representation of the encoder, which outputs the prediction of the original input for the trained task. Typically, the encoder consists of a CNN architecture to extract feature maps of the image. The encoder consists of two convolutions at each scale, each is followed by an activation function (*ReLU*) with drop-out of 0.1. In order to down sample the feature maps, 2×2 max-pooling is applied at each level, and the encoder keeps downscaling the

features until it reaches the bottleneck. On the other side, the decoder consists of two transposed convolution at each level, which results in upscaling the feature maps and reducing the number of channels. The bottleneck in the middle part of the network is built from simply two convolutional layers. Skip Connections are used to concatenate feature maps in the decoder with the feature maps from the encoder at a corresponding resolution. The bottleneck results in the loss of features but Skip Connections are used to overcome the bottleneck problem. The final layer after the decoder is a 1×1 convolution and activated by a *Sigmoid* activation function, which projects map feature vectors to the desired number of classes.

2.6. Random Forests and Feature Extraction

In general, traditional Machine Learning techniques are more understandable compared to Deep Learning techniques. Deep Learning algorithms usually take more time to train due to a considerable number of parameters, whereas traditional Machine Learning algorithms take only a few seconds to few hours to train, for example, it is common to distinguish between two types of models: the Black Box and White Box. The Black Box model is often complex and its prediction mechanism is not interpreted easily, which is usually associated with the behavior of Deep Learning algorithms. Whitebox models, on the other hand, are simple enough that their functionality can be intuitively explained and visualized.
Decision Trees and Random Forests are among the traditional Machine Learning techniques that are widely used and easy to understand. However, if the tree gets sufficiently large, it becomes difficult to analyze [12].
Decision Trees consist of nodes and edges connecting the nodes in a tree-like architecture. The nodes contain a condition to test a particular feature within the input data. Depending of the condition, the node forwards the input data using directed edges which subsequently determine the order in which these tests are performed. Eventually, the Decision Tree reaches the bottom node in the tree which is referred to as a Leaf node and contains the final prediction class.
The training process of Decision Trees requires a top-down partitioning algorithm to

recursively divide the tree into smaller sub-trees. Therefore, a metric is required to subdivide the set of features according to a certain criterion. These metrics measure the impurity of the target feature within the input data. There are many measures and metrics to condition the partitioning of the tree, and one of the most common measures is the Gini index or Gini impurity. It measures the probability of a particular classification of the input data being incorrectly classified. The Gini index is given by Eq. 2.4:

$$Gini = 1 - \sum_{i=1}^{n} p_i^2, \tag{2.4}$$

where p_i is the probability of an object being classified into a particular class. At each subtree level, the Gini index is calculated for each subset. Then the subset with the highest index will become the definite root for the splitting [13].

The major drawback of Decision Trees is that they tend to overfit the training data easily as the model becomes more complex. Therefore, the error curve tends to keep decreasing on training data; however if the model is validated on a separate set of data, there will be a noticeable difference between the error rates. In addition, Decision Trees are more likely to be unstable and dependent on the training data; in some cases, certain changes in training data affect the final predictions of the model[13].

Random Forests, on the other hand, is an Ensemble Learning method for regression and classification [14]. They are comprised of many Decision Trees, which are randomly selected in each decision split. Each individual Decision Tree in the Random Forest yields to a class prediction, the class with the highest number of votes becomes the final prediction of the model.

Random Forests generally outperform Decision Trees and reduce the possibility of overfitting. Nonetheless, the performance can still be sensitive to feature changes which can affect the general results [13].

The reasoning in a Decision Tree or a Random Forest for model analysis can be illustrated visually and explicitly by means of a node-link diagram. For further analysis, the relevancy of the features in Random Forests can be ranked by its importance [15], which is referred to as Feature Importance. After training the model, the Feature Permutation Importance can be calculated by; firstly, permutating values of each

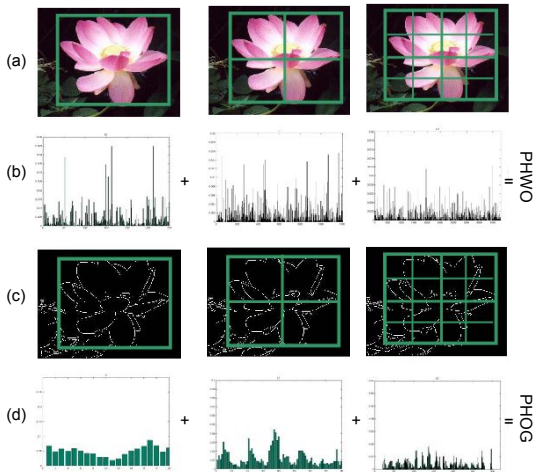


Figure 2.1.: Using both colors and shape feature descriptors for Spatial Pyramid Matching [18].

feature i.e. randomly permuting a feature in the original dataset, then calculating the error difference of the permutations. Afterwards, all the error differences will be averaged and standardized, resulting in a ranking score according to importance for each feature.

In Computer Vision tasks, the use of Random Forests is considered a challenging problem because of the complexity of features and the fact that they require handcrafted features and Feature Engineering. However, there are many examples in literature on of the use of Random Forests for Computer Vision tasks including segmentation, classification, and object recognition [16]. In addition, there is a variety of efficient Random Forest models including color, shape, and texture recognition tasks [17].
Handcrafted features are manually extracted properties of data. It is a common practice in Computer Vision and traditional Machine Learning. It also aims to derive specific meaningful information using various algorithms, for example, corner or contour features can be extracted from images by defining filters to convolve the image into a specific representation. This results in a simplified representation of the image features, for instance, handcrafted features were commonly used with

traditional Machine Learning approaches such as Support Vector Machines (SVM) and Decision Trees [18], [19]. Unlike the modern feature learning process such as CNN, the classical models must be supplied with handcrafted features, as they will not be able to extract the features from complex data such as image data. Figure 2.1 illustrates the process of simplifying the appearance and shape features of an image into a fixed sized descriptor vector. Feature Extraction also needs to be applicable to various kinds of image data to automate the process. Feature descriptors are algorithms to convert an input image into a feature vector. They describe the information in the image in a fixed sized vector. Feature descriptors encode interesting information into a series of numbers and provide numerical characteristics that can be used to differentiate one feature from another. Feature descriptor is considered robust when the algorithm extracts invariant image feature automatically, which makes it possible to retrieve and identify the image even if the image was transformed, such as the scale-invariant feature transform (SIFT) and histogram of oriented gradients (HOG) [20], [21], which are feature descriptors that detect and extract spatially localized information from images.

3 Methodology

Many studies have tried to automate the vectorization of cadastral plans and historical maps. However, very few studies have used the modern Deep Learning algorithm as a solution to the problem of the vectorization of cadastral maps [2]. This chapter introduces a cadastral map vectorization pipeline that segments and classifies the individual regions in the map, while discarding the geographic guidelines and writings. Section 3.1 presents an overview of the pipeline used for vectorization, followed by detailed descriptions of each component of the pipeline.

3.1. Cadastral Map Vectorization Pipeline

The cadastral map vectorization pipeline attempts to fully automate the process of converting raster images into vector data formats. Since the raw scanned maps were not completely registered with their vectorized images, the first pre-processing stage of the pipeline is the image registration process. This step aims to automatically align the raster images with their vectorized images, as discussed later in Section 3.2.1. Furthermore, the raw data consists of four main classes: *background*, *water*, *buildings* and *contours*. These classes were divided into groups and extracted as separate images as a preparation for the training step. The first main phase of the pipeline is to train a model that can approximate the Euclidean distance transformation using a U-Net architecture as shown later in Section 3.2.2. Using the raw scanned maps as input, the model converts the original images into a simplified form that can be processed with the watershed algorithm directly. Therefore, the Euclidean Distance Transformation image of each map is calculated from the registered contours. Afterwards, pairs of scanned images with their Euclidean distance transformation were sliced into patches with a fixed size and used as training data for the model. The goal of the first main phase is to train a model to automatically predict the markers i. e. to estimate the Euclidean distance transformation matrix of the input images.

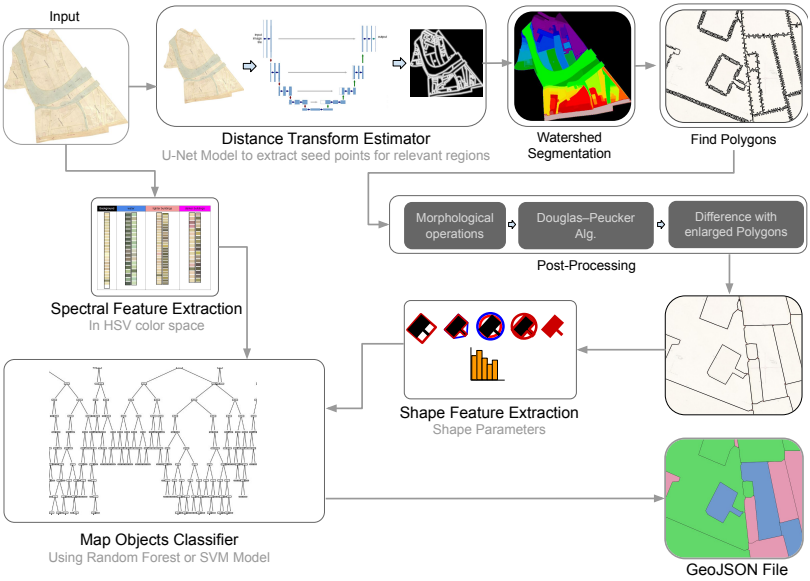


Figure 3.1.: Overview of the components of the cadastral map vectorization pipeline.

After training the model, as discussed in Section 3.2, the model was able to estimate the Euclidean distance transformation map using only the raw RGB input map.
The next stage in the vectorization procedure is to identify regions on the map using watershed algorithm with the estimated Euclidean distance transformation, as described in Section 3.3. The watershed segmentation uses dark pixels (also called local minima) of Euclidean distance transformation as potential regions to segment, and the lighter pixels (also called local maxima) as the "watersheds" or boarders that separate each region.

Since the regions of the map have been extracted from the image, each region can be converted into a binary mask. Afterwards, the contours of the binary mask can be calculated, which follows the outermost boundaries of a given mask and transforms it into a polygon with a sequence of points.
Before transforming the binary mask of each region into a polygon, a morphological closing operation is required for filling the small holes and noise artifacts that appear on the edges, while preserving the shape and size of the objects in the image. This is

the first step in the post-processing procedure, resulting in less coarse edges in the extracted polygons.
The next step in post-processing is to refine the extracted polygons and obtain smoother polygonal chains. For instance, the goal is to approximate the polygons with a smaller set of points. This can be achieved using the Douglas-Peucker algorithm, as will be discussed later in Section 3.4.1. Finally, the last step of post-processing is to eliminate the gaps between the polygons, as will be noted later in Section 3.4.2. A possible solution for this problem is to enlarge the polygon by an offset then to calculate the difference with its neighboring polygon. Thus, the polygons adopt the outer edge of the neighboring polygons, therefore, the gaps between the polygons are eliminated and the same coordinates of the shared points are preserved in both neighboring polygons. This results in polygonal chains which have no gaps or overlaps with other polygons.
Eventually, refined spatial features are extracted from raster images. In order to extract further non-spatial information such as classes (e.g. *water*, *buildings* or *background*), the Map Objects Classifier is implemented to determine the classes of each polygon, which uses a Random Forest or SVM model and will be discussed later in Section 3.5. Feature Engineering and handcrafted shape and spectral features have been used on the polygons to obtain features for the model. Therefore, after a detailed feature analysis, specific features were selected in order to provide the model with the most useful information to obtain high classification accuracy of each class, as described later in Section 3.5.1. Finally, the pipeline assembles the refined polygon chains and the attributes of each map object into a GeoJSON-file as an output to obtain a vectorized format.

3.2. Distance Transform Estimator

Distance Transform Estimator is based on Deep Distance Transform technique [22], which simplifies the input for classical segmentation algorithms. It consists mainly of estimating the Euclidean distance transformation matrix of raw images. The main advantage of this approach is that it provides markers for further segmentation

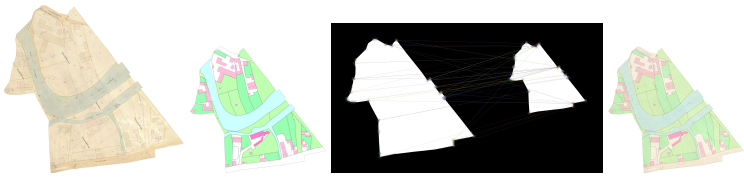
procedures without any handcrafted information about the behavior or global shape of objects. As previously discussed in Section 2.4, the markers of potential regions can be observed in the Euclidean distance transformation of the image, which can be used for creating a topological surface for the watershed algorithm. Therefore, the main objective of this step is to estimate the Euclidean distance transformation matrix of the input images. The provided dataset consists of pairs of scanned maps and their vectorized images. In order to obtain training data for the model, a preprocessing step is performed on the raw dataset.

3.2.1. Image registration and data preparation

The first issue to be handled is that the scanned maps are not fully aligned with their vectorized images. Therefore, the first step of pre-processing is image registration. Image registration is the process of aligning two or more images of the same scene. Aligning two images means that one image is transformed using a transformation technique so that it matches the other image. The process consists of aligning a *moving image* (image to register) with a *fixed image* (the reference image). Figure 3.2 illustrates the process of image registration, which consists of the four following steps [23]:

a) Feature detection: during this step, the feature detection is used to determine noticeable edges and distinctive lines of both the *fixed image* and the *moving image* which are determined by selecting points referring to the areas of these features. The selected points provide a set of *fixed points* (points selected on the *fixed image*) and *moving points* (points selected on the *moving image*), which are also called control points (CPs).

b) Feature matching: the *moving image* and the *fixed image* are correlated with each other using intensity or spatial similarities, in order to detect the features.
c) Transform model estimation: the functions for overlaying the *moving image* over the *fixed image* are called mapping functions. By means of the pairs of CPs, the



(a) (b) (c) (d)

Figure 3.2.: Using the four step image registration to automate image registration process: a) selecting control points from the original image (*fixed image*); b) selecting control points from the vectorized image (*moving image*); c) Feature Extraction and visualizing feature matching; d) An overlay of the *moving image* and the *fixed image* after performing affine transformation.

mapping functions can estimate the geometric transformation model.

d) Image re-sampling and transformation: the *moving image* is transformed according to the geometric transformation, which is defined by the mapping functions in the previous step. The geometric transformation includes re-sampling, rotating and stretching of the *moving image*.
The maps and their vectorized drawings have no mutual distinct features except for the outer edges. To overcome this problem, both images were transformed into "silhouette" masks to prevent mismatching with non-mutual features in both images, as shown in Fig. 3.2(c). Therefore, the feature detection and feature matching are based on the outermost borders of scanned maps and their vectorized images.
The raw data consists of four main classes: *background*, *water*, *buildings*, and *contours*. Each of the classes is assigned to a color in the vectorized images. Therefore, the classes are derived from the colors and split into groups as preparation for the training step.

The Euclidean distance transformation matrix can be acquired from the registered *contour* class to be used as a label to train the model, because the *contour* class comprises not only the outer border of each region but also the region itself. Afterwards, pairs of scanned images with their Euclidean distance transformation were sliced into patches with a fixed size of 256×256 and used as training data for the model. As seen in Fig. 3.3, the patches (a) and (c) are respectively the input and label data for

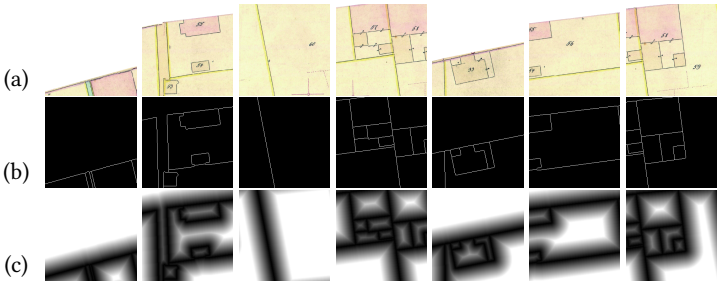


Figure 3.3.: Sliced image patches with a size of 256 × 256: (a) raw input map image; (b) *contour* class; (c) inverse Euclidean distance Transformation of the *contour* class.

the model. By means of the contour information only, the network can be designed to estimate the Euclidean distance transformation matrix.

3.2.2. Network architecture

The Network is based on the U-Net architecture as explained previously in Section 2.5, which consists of convolutional encoding and decoding parts. The Network has a depth of 5 blocks for the encoder and decoder, where each convolutional layer has a 3×3 kernel filter with no padding. For encoding, a down sampling is performed using a 2×2 max-pooling. For decoding; however, a transpose convolution is used with a 2×2 *stride* in order to upscale the image with trainable weights, rather than using a bilinear interpolation. The activation function *ReLU* is performed after each convolutional layer except the output layer, which is activated by a *Sigmoid* activation function [10]. As mentioned before in Section 2.5, by means of Skip Connections, the feature maps of the encoding part are integrated into the decoding part aiming to induce the information that might be lost after the encoding.
Figure 3.4 illustrates the network architecture based on Deep Distance Transform [22]. The network takes an input of 256×256 patches of original RGB images paired with its gray-scale Euclidean distance transform. The loss function differs from the one used in the original algorithm since the mean squared error (*MSE*) loss function

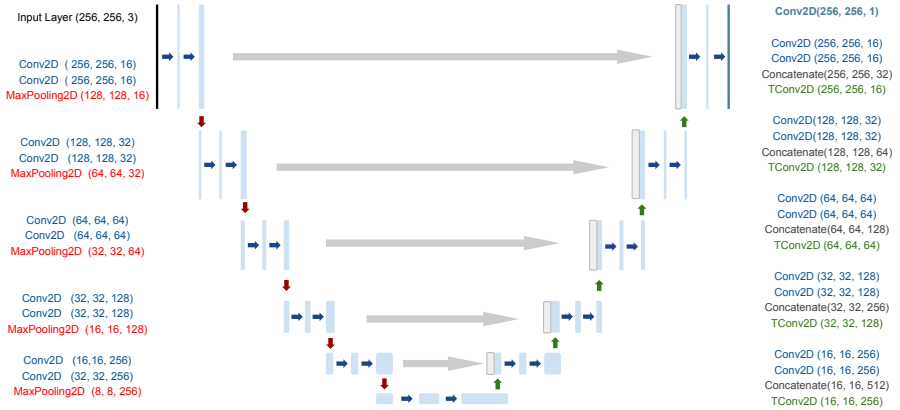


Figure 3.4.: The network architecture of Distance Transform Estimator.

was used, as shown in Eq. 3.1:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}\left(y_{pred} - y_{true}\right)^2 \tag{3.1}$$

where i represents the i-th pixel, n is the total number of pixels, y_{pred} is the predicted pixel value and y_{true} is the true pixel value. The used optimizer is Adaptive Moment Estimation ($ADAM$) with a learning rate of 0.001 with decay 0; β_1= 0.9, β_2=0.999 and the batch size equals to 64. Since, the number of input images is relatively small, they were augmented in order to obtain more image patches and a more generalized model. This was achieved by applying random shifting, re-scaling, zooming and flipping to the train images, which is later noted in Section 5.1.

In result, the model was able to predict the Euclidean distance transformation of 256×256 map patches, as shown in Fig. 3.5. Most of the parcel numbers as well as irrelevant marks on the input images are not shown in the estimated distance transformation matrix, while the contours of the image were highlighted.

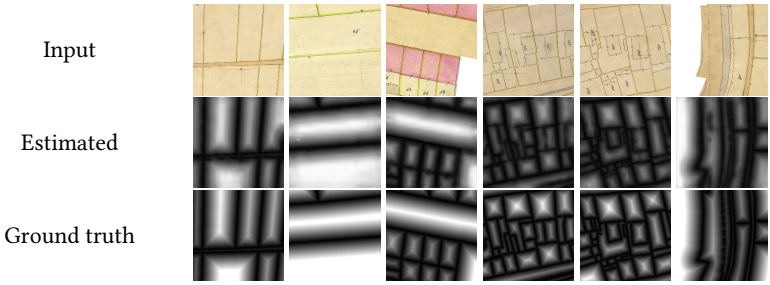
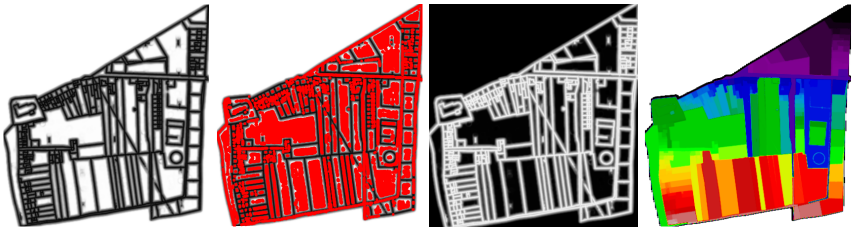


Figure 3.5.: A sample of unseen image patches of different maps compared with ground truth distance transformation images.

3.3. Watershed Segmentation with estimated distance transformation

The model is trained to estimate the Euclidean distance transformation of a 256×256 image patch. Therefore, the estimated Euclidean distance transformation of a full map can be obtained by using the Sliding Window technique over an input image. Thus, the trained model takes a 256×256 window as its input and approximates its Euclidean distance transformation. The absence of information about the other nearby image windows causes creases and artifacts to be show between the estimated window images. To deal with this issue, 80 pixels from each side of each window image is trimmed off to obtain a small rectangle in the center of each window of size 96×96. Thus, attaching the center windows achieves a smooth and crease-free full map of approximated distance transformation.

As mentioned before in Section 2.4, the watershed is considered a supervised segmentation when used with a Machine Learning model that is trained to estimate the distance transformation matrix. Moreover, the topographic surface of the image has ridges that correspond to the white pixels, while the valleys correspond to the darker pixels. Having said that, the first step would be to obtain the estimated Euclidean distance transformation using a sliding window as noted previously, then the search for local maxima of the full map is performed. The local maxima are calculated in a footprint of 3×3, after that the adjacent maxima are grouped together as a single



(a) (b) (c) (d)

Figure 3.6.: Watershed Segmentation steps: a) Calculate the approximated distance transformation of the full map; b) Find the markers (highlighted in red) using the local maxima of each region; c) Invert the distance transformation map to obtain the valleys and the ridges in white and black pixels, respectively; d) Flood each region using the watersheds and the markers to obtain the segmentation (each unique region is assigned to a spectral color).

marker. This way the markers of the image are encapsulated as potential segmentation objects in the valleys to perform the watershed segmentation. After collecting the markers from the image, the image is inverted to obtain the valleys and ridges in white and black pixels, respectively. Finally, the valleys are gradually flooded starting from the markers until the "water" reaches the watershed lines (ridges). The regions are merged into a single region when the "waters" that are coming from different flooded regions meet. Figure 3.6 illustrates the steps performed in order to obtain the segmentation using the approximated distance transform.

3.4. Post-Processing

The watershed algorithm with its region-growing and merging mechanisms produces unique binary masks for each individual region. However, these regions are sensitive to seed initialization and may produce jagged and non-smoothed boundaries. Therefore, a morphological closing operation was performed, aiming to fill the holes and produce less jagged edges, while still preserving the shape and size of the objects in the image. This operation results in less coarse edges but also maintains the spatial

features of the binary masks.
Each individual region can now be considered a unique binary mask from which a polygon with a sequence of points can be derived. The transformation of a binary mask into a sequence of points was calculated using the algorithm presented in [24]. Since the binary mask and the holes have borders with connected pixels, the outer edges can be extracted in reference to the closest outer pixels. As a result, the coordinates of the outer pixels of each region and hole were maintained as a sequence of points in order to vectorize the raster images.

3.4.1. Polygons and Curves Simplification Using the Douglas-Peucker Algorithm

After extracting the polygons from each region of the segmented map the boundaries of the polygons reveal rough and jagged edges with a large set of points. To solve this arising problem, the Douglas-Peucker algorithm is included in the Post-Processing module.
Basically, the Douglas-Peucker algorithm is a method for line approximation [25], where the polygons in this case are closed curves. The main purpose of this algorithm is to find a similar linear curve using a smaller set of points. Based on the concept of dissimilarity of the maximum distance, the algorithm recursively validates the distance between the original curve points and their simplified linear curve.

First of all, the algorithm requires a threshold ϵ, which defines the maximum dissimilarity tolerance to exclude or include points in the approximated curve. Then it measures the distance between each point of the curve and the baseline. The baseline, as shown in Fig. 3.7(b) starts with the line segment between the first and last point of the curve. Then, it tries to find the farthest point from the line segment with the maximum perpendicular distance from the baseline. Next, if the farthest point is greater than ϵ, then it must be included in the approximated curve. Otherwise, if the farthest point is closer or equal to ϵ than to the line segment, then the point is not included in the approximated curve. In the same way, the algorithm recursively calls itself until an approximated curve is generated, as shown in Fig. 3.7. The time

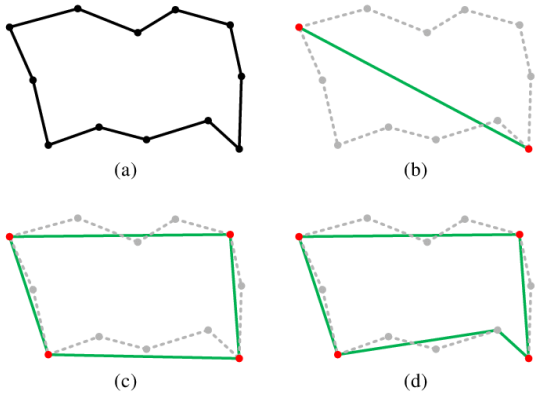


Figure 3.7.: Polygon approximation with the Douglas-Peucker algorithm. (a) The input polygon; (b) The initial start and end points (c) Simplified polygon after one iteration; (d) Final simplified polygon [26].

complexity of this algorithm is $O(n^2)$, when it runs for each point n and for each line segment $n-1$. However, using the Dynamic Convex Hull data structures speed up the Douglas-Peucker algorithm reaching a time complexity of $O(n\ log\ n)$ [27].

3.4.2. Elimination of Gaps and Overlaps

After the extraction of the polygons with watershed segmentation, normally gaps and overlaps between the polygons occur. Since there are no gaps nor intersecting lines in the raster image, each polygonized region should belong to a unique area in the vectorized format. However, adjacent polygons should share the points of their neighboring polygons.

The *polygons snapping* method represents the intuitive solution in this case. It consists of defining rules to merge the points and lines segments of the polygons with each other. The first rule is to snap each point between a polygon and its neighboring polygons within a maximum distance d using the Nearest Neighbor algorithm (NN) [28], which eliminates the gaps between d distant points, as illustrated in Fig. 3.8(a). The second rule is to project each point onto a line segment of a neighboring polygon

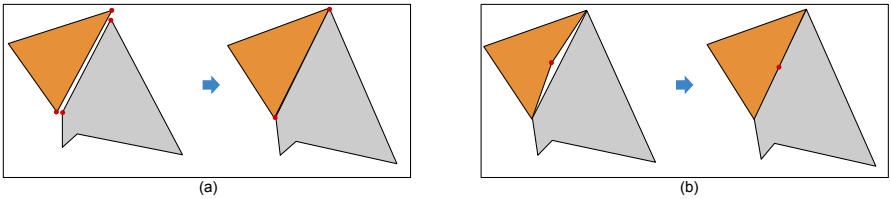


Figure 3.8.: Eliminate gaps using the *polygons snapping* method; a) The point to point rule; b) The point to line segment rule.

and calculate its perpendicular distance to the line segment. If the distance is within the maximum distance d, then a new point with the same coordinates is added to the line segment, as illustrated in Fig. 3.8(b).

The major setback of this method is its dependency on the maximum distance d, where d should be chosen according to both image resolution and the size of the objects in the map. Which could be impossible to compromise in some cases, where it can eliminate relevant small building and structures on the map. Furthermore, the implementation of this method is computationally complex since each pair of points must be compared twice for each two adjacent polygons.
Keeping in mind that, the two presented rules are not sufficient to eliminate all varieties of gaps, as there are some gaps that do not apply to the previous rules. Figure 3.10(e) show some of the gaps that could not be eliminated using both rules of the *polygons snapping* method.
The results obtained by the previous approach suggest that using a slightly more advanced algorithm helps in overcoming the problem and provide a more convenient solution.
The so-called *Difference with enlarged polygons* is the implemented method to produce gap-free polygons. It consists of two main steps: 1) Enlarging the polygon by an enlarge factor a; 2) Calculating the difference between the polygon and its neighboring polygons. This means including all the points in the enlarged polygon and discarding the intersections with the neighboring polygon. Subsequently, the polygon takes over the outer points of the neighboring polygons as new points, resulting in a seamless representation while retaining the shape and structure of the

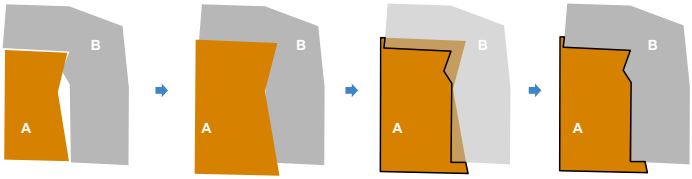


Figure 3.9.: The steps of eliminating the gaps using the *Difference with enlarged polygons*.

polygons. The steps of this approach are illustrated in Fig. 3.9. Firstly, increasing the size of **A** by an enlarging factor a, which results in filling the gaps between the polygons **A** and **B**, which could also yield to some overlaps between the polygons. Next, the difference between **A** and **B** highlights the points in the enlarged **A** while excluding all the other points that are in **B**, which results in seamless adjacent polygons with no intersections. This method is done globally, which provides a way to ensure the validity of the resulting polygons. Figure 3.10 shows the seamless contours produced by the *Difference with enlarged polygons* in comparison to the *polygons snapping* method.

In addition, this method allows generating a *contour* class as vector graphic by adjusting the first step of the algorithm. Specifically, instead of enlarging one polygon, both polygons are enlarged so that the polygon takes over the outer points of the enlarged neighboring polygon instead of its original outer points. Thus, it forms a small distance between the polygons which can be masked out of the original map, creating a *contour* class, as shown in Fig. 3.10(g). This property; in particular, has a beneficial application in terms of evaluating the pipeline. Since it produces a very similar construction of the ground truth vectorized images it gives the ability to perform a comprehensive evaluation of the extracted polygons, as discussed later in Chapter 5. Fig. 3.10 shows a comparison between each of the used post-processing methods starting from the extracted polygons of watershed segmentation and ending with the final refined polygons.

The enlarge factor is selected according to the desired results where the *contour* size can be adjusted according to the preferable thickness, as seen in 3.10(g). For instance, when $a = 2.85$ it produces the most similar results to the vectorized ground truth

Figure 3.10.: Comparison between different steps of the post-processing module including; (a) original input map; (b) extracted polygons without post-processing; (c) morphological closing; (d) using the Douglas-Peucker algorithm; (e) the *polygons snapping* method to close the gaps with $d = 4$ pixels; (f) the *Difference with enlarged polygons* method to close the gaps with an enlarging factor $a = 2.85$; (g) the *Difference with enlarged polygons* method to close the gaps including *contour* class with an enlarging factor $a = 2.85$.

images. However, the *contour* class can optionally be excluded from the vector data if the objective is to obtain slightly larger polygons with no *contour* class.

One of the main advantages of this method is that it is computationally less expensive unlike the previous method. To enlarge a polygon by the factor a the Minkowski Sum is calculated between the polygon $\mathbf{A}$ and a circle $\mathbf{C}$ with a radius is equal to the factor a. The Minkowski Sum is a geometric operation that can enlarge polygons efficiently [29], which is calculated as follows in Eq. 3.2:

$$\mathbf{A} \oplus \mathbf{C} = \{x + y \mid x \in \mathbf{A}.y \in \mathbf{C}\}, \tag{3.2}$$

The complexity of the calculation of the Minkowski Sum depends on the shape of the polygons. The circle $\mathbf{C}$ is always a convex polygon, while in the worst case if $\mathbf{A}$ is non-convex, the Minkowski Sum can be calculated with a time complexity of $O(nm)$.

The *Difference with enlarged polygons* method is more capable of handling very large spaces between polygons. For instance, when the polygon is enlarged in the first step the main goal is to create an overlap with an adjacent polygon. However, if the gap is larger than a, the enlarged polygon fills some of the space between the adjacent polygons without creating any overlap. Then, while processing the adjacent polygon, the gap is now much smaller than before, meaning, it will be easier to reach the other processed polygon and create an overlap. Hence, a seamless structure or smaller gap is obtained despite the original large gap, which considerably reduces the possibility of having unresolved issues with large gaps.

3.5. Map Objects Classifier

In the last step before exporting the vectorization as a GeoJSON file, more information about the maps can be included as attributes. For instance, extracted polygons can represent either *background*, *water*, or *building*. Therefore, the classification of each extracted polygon is performed to induce more information into the vectorization format. As previously noted, the classes *background*, *water* and *building* are derived

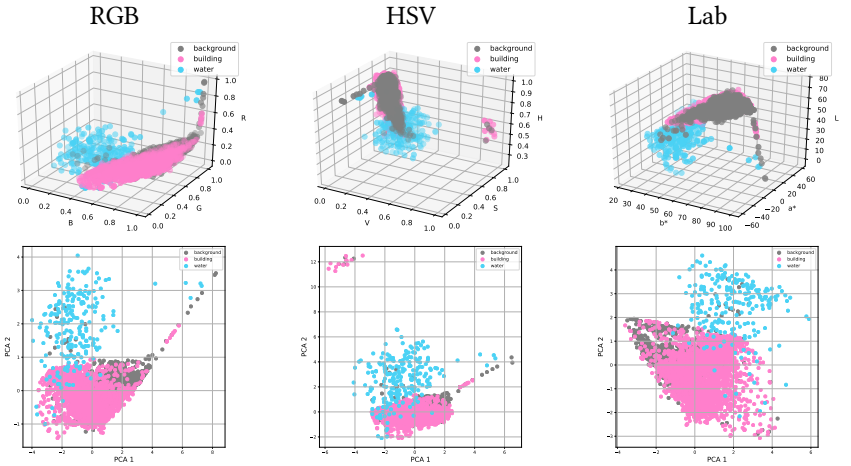


Figure 3.11.: Visualization of the classes with corresponding colors in multiple color spaces; First row 3-dimensional representation; Second row using PC1 and PC2 of Principle Component Analysis (PCA) as 2-dimensional representation.

from the vectorized train images as binary masks, which is beneficial to train a classification model to predict the class of each polygon.

In order to decide which model is suitable for this problem, a data analysis was carried out at the beginning of the experiment. Keeping in mind, that the most distinct feature associated with each class is the color information, the colors were extracted from each polygon and analyzed in different color spaces. Figure 3.11 shows the distribution of each class visualized in 3- and 2-dimensional representations for different color spaces. The result suggests that the Map Objects Classifier can be designed using traditional Machine Learning algorithms with a fixed sized feature descriptor using color features and further sophisticated Feature Engineering.

3.5.1. Feature Extraction

As discussed before in Section 2.6, the performance of traditional Machine Learning algorithms is highly dependent on feature descriptors extracted from the data,

therefore, further data analysis were carried out. The visualization of color features in Figure 3.11 showcases, that the class *water* is almost linearly separable in all of color spaces. However, there is a significant color correlation between the classes *buildings* and *background* in the input data. The distribution of data in HSV space reveals the most compact and enclosed clusters. It suggests that the use of HSV as color features in combination with some shape features allows data separation and object classification with high accuracy.
Two types of Feature Extraction are performed; spectral Feature Extraction based on color values of RGB images, and shape Feature Extraction based on shape parameters. A similar Feature Extraction approach was performed on a Plant Recognition Model to classify and identify different kinds of plant leaves, which have unique shape and color features [17].

Spectral features

As noted earlier, the HSV color space provides a compact cluster of data that can eventually be converted into a feature vector. In order to obtain the feature vector, a histogram equalization is performed on the RGB input images first, then a median filter is applied to remove the noise. Afterwards, the colors are transformed into HSV color space and the image is vectorized into 1×3 vector. Contours, words and numbers are denoted on the map in dark colors, which correspond to lower intensity value in the value-channel of the HSV space. Thus, pixels with intensity in the value-channel less than 0.3 are removed from the image vector in order to obtain polygons with more pure and vibrant colors. Finally, statistical measures are calculated to obtain a value for each channel and then they are combined into a feature vector. The statistical measures are selected after conducting a comprehensive Feature Importance Analysis, as discussed later in Chapter 5.
The statistical measures include the *median*, the *median absolute deviation*, the *skewness*, the *mean* and the *standard deviation* of pixel values.

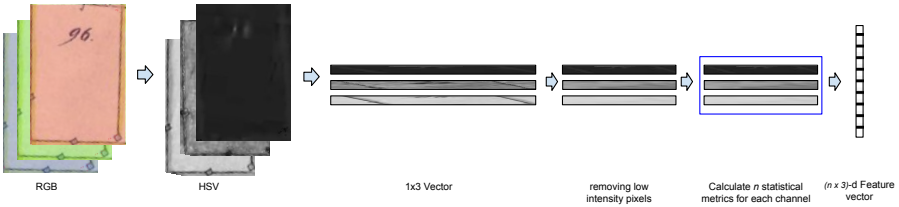


Figure 3.12.: The steps of extracting spectral feature vector.

Shape features

In addition to the spectral features, map objects have distinctive shape and geometrical features. For example, polygons with the *buldings* class are mostly rectangular and more likely to be convex polygons. On the other hand, the *water* classed polygons tend to be elongated along the map, which results in a high aspect ratio, but they are more likely to be non-convex with relatively large areas. The *Background* class polygons; however, have even larger areas in the map and tend to have less holes. These observations can be interpreted into mathematical measures, which are given in the following Equations as:

$$Rectangularity = \frac{Polygon\ Area}{Minimum\ Bounding\ Rectangle\ Area}, \tag{3.3}$$

$$Solidity = \frac{Polygon\ Area}{Convex\ Area}, \tag{3.4}$$

$$Convexity = \frac{Convex\ Perimeter}{Polygon\ Perimeter}, \tag{3.5}$$

$$Aspect\ ratio = \frac{length}{breadth}, \tag{3.6}$$

$$Relativ\ Area = \frac{Polygon\ Area}{Map\ Area}, \tag{3.7}$$

$$Circularity = \frac{Polygon\ Area}{Perimeter^2}, \tag{3.8}$$

$$Elongation = \frac{|length\ -\ breadth|}{length}. \tag{3.9}$$

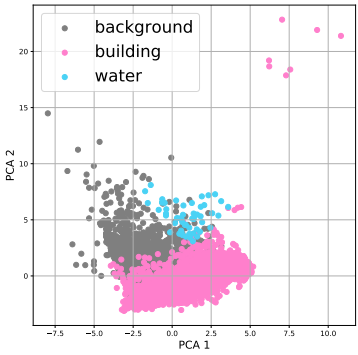


Figure 3.13.: The shape and spectral feature vectors in 2-dimensional space using PCA visualization.

Similar to spectral features, the measures are selected after performing Feature Importance analysis to exclude redundancy in the feature vectors, as discussed later in Chapter 5.

3.5.2. Classification Models

The shape and spectral features are combined together to create a feature vector to feed into a classification model. Figure 3.13 illustrates the benefit of extracting shape and spectral features for each polygon. It is worth noting that after combining spectral and shape features, the clusters of each class are visually easier to separate and have more refined characteristics to distinguish.
The feature vectors are normalized to avoid imbalanced classification problems due to the different numerical values of different features. The Map Objects Classifier is tested to perform a classification of polygons using the extracted features as input. A Random Forest Model [14] and a Support Vector Machine (SVM) [19] are trained to classify the polygons, then they are compared against each other, as discussed later in Section 5.3.

4 Implementation

The cadastral map vectorization pipeline consists of several components that can be customized for different preferred results. Therefore, a web application is implemented to provide more control and usability of the individual components.

4.1. Web Application

The web application is developed to provide user-friendly web interface of the cadastral map vectorization pipeline. The main control panel allows to determine the Douglas-Peucker parameter ϵ and the choice of the gaps-elimination method used including its parameters, as shown in Fig. 4.1. The *Difference with enlarged polygons* method allows to choose weather a contour class is generated with adjustable thickness depending on the enlarge factor a. Furthermore, the *Contours sensitivity* defines the threshold for the watershed segmentation, which can be chosen according to drawn contours on the cadastral maps. This value represents a trade-off between the smoothness of the polygons and the number of identified polygons. Therefore, if the contours on the map are mostly complete and clear, it is recommended to select a lower sensitivity. However, if the contours are not complete or the ink is mostly faded, it is recommended to select a higher sensitivity to detect more contours and achieve a better identification of polygons. Users can submit a historical cadastral map into the server and set the variables in the main input panel. Then the server responds with the resulting GeoJSON file and provides an interactive visual representation of the vectorized map, as shown in Fig. 4.2. The web-based visualization includes the number of points in each polygon and the probability of each classification. Finally, the user can download a GeoJSON file, where each polygon is represented as a feature and the coordinates correspond to the index of the original image pixels.

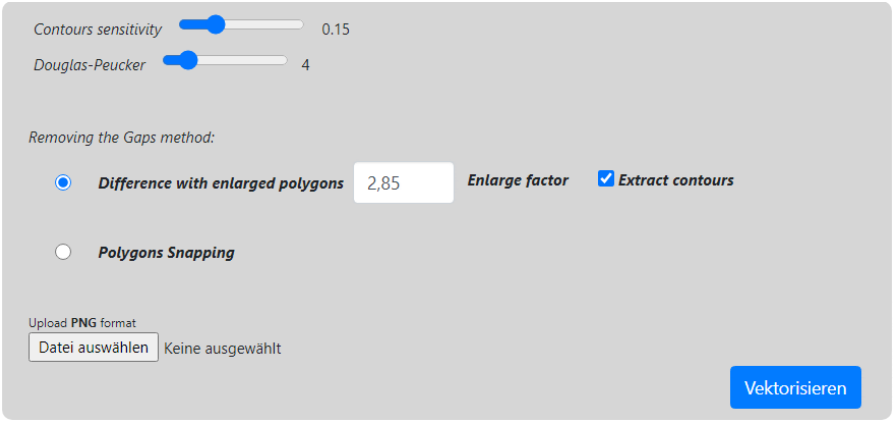


Figure 4.1.: The user-interface of the web application to use the cadastral map vectorization pipeline.

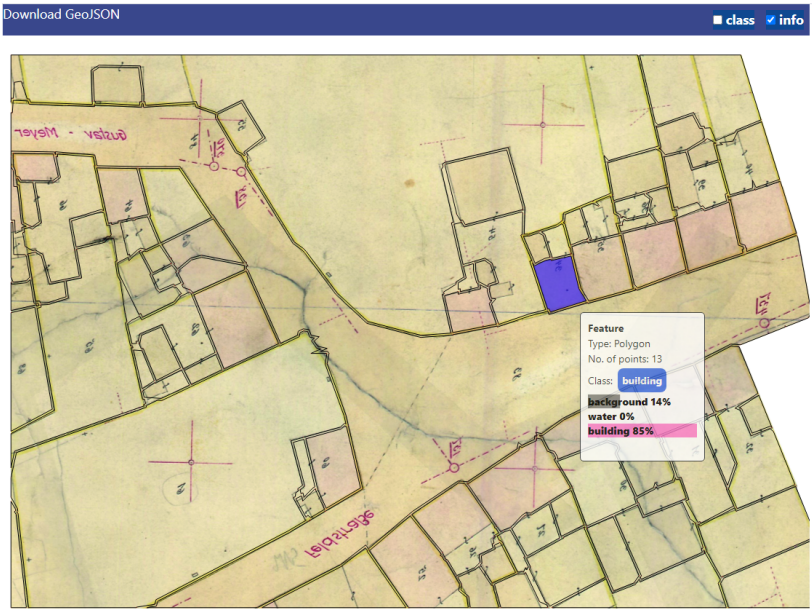


Figure 4.2.: The results of the web application with the interactive visualization using d3.js library [30].

5 Experimental Results

Since the pipeline is composed of multiple steps, the quantitative evaluation is divided into two phases: Geometrical evaluation, and Map Objects classifier evaluation. This Chapter begins by giving a brief overview of the dataset and the evaluation methods used, followed by a discussion of the obtained results with respect to quantitative and subjective evaluations.

5.1. Dataset

The experiments consist of using 27 different annotated scanned cadastral maps of Mühlhausen provided by Institut für vergleichende Städtegeschichte (IStG) [31], where the images have similar resolutions with an average of 2899×3168 and dots per inch (dpi) of 600. In order to train the Distance Transform Estimator each map is divided into 900 patches with a fixed size of 256×256. Then 80% of the patches is used for training, while the rest is used for testing and validation. Since the pipeline operates on the full maps, three full maps were excluded from training in order to assess the performance on unseen data; leaving 24 maps for training. Yet, the original image patches were not used for training but rather replaced with new augmented image patches after applying a series of random transformations including shifting, scaling, zooming, and vertical and horizontal flipping. The main goal of using this approach is to reduce the generalization error and to use the original images to assess the performance.

5.2. Geometrical Evaluation

The number of polygons predicted after the post-processing step is 6576 polygons, while having 7454 ground truth polygons. Since no geo-referenced data is provided

to eliminate map structures with small area, the area of the polygons is calculated in pixels, where polygons with an area between 1500 pixels and 300000 pixels are considered for the evaluation.
The metric used for evaluating the pairs of polygons is the Intersection over Union (IoU), also known as the Jaccard index, which is an evaluation metric used to measure the accuracy of an object detector on a particular task [11]. Specifically, it compares the similarity between two arbitrary shapes by measuring the number of intersected pixels between the target and prediction masks ($area\ of\ overlap$) divided by the total number of pixels present in both masks ($area\ of\ union$), as shown in Eq. 5.1. This metric ranges between 0–1; where 0 implies no overlap and 1 implies a perfect overlap. Therefore, the IoU metric can quantify the percentage of overlap between the ground truth and the predicted masks.

$$IoU = \frac{area\ of\ overlap}{area\ of\ union}. \tag{5.1}$$

Nonetheless, the IoU does not assess performances in terms of true positives (TP), true negatives (TN), false positives (FP) or false negatives (FN), which are important values when evaluating a model in order to calculate precision, recall and F1-score. However, those values can be derived using different thresholds $\tau \in [0, 1]$ of IoU. Thus, the predictions are compared using different thresholds with the ground truth to be considered as TP when IoU is above a certain τ. Based on that, it is possible to consider a prediction with an $IoU \geq \tau$ as a TP, a prediction with an $IoU < \tau$ as a FP and a ground truth with no predicted pixels as a FN. For a certain threshold τ and a set of predicted polygons $\mathbb{P}$, TP is defined as follows in Eq. 5.2:

$$TP = |\{\mathbf{p} \mid \mathbf{p}_{IoU} \geq \tau \textbf{ and } \mathbf{p} \in \mathbb{P}\}|. \tag{5.2}$$

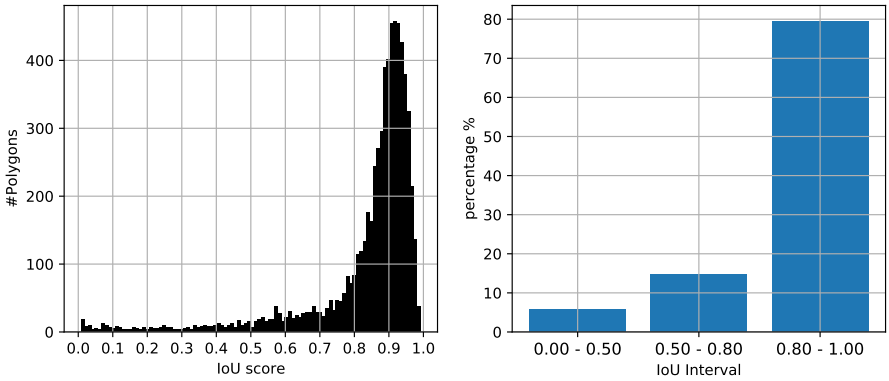


Figure 5.1.: *IoU* score distribution of the extracted polygons; number of polygons with the corresponding *IoU* scores on the left; On the right the bars represents the percentage of the polygons for $[0.00 - 0.50]$, $[0.50 - 0.80]$, and$[0.80 - 1.00]$ *IoU* intervals.

Therefore, it is possible to compute precision, recall and F1-score as follows:

$$precision = \frac{TP}{TP + FP} = \frac{TP}{\#predictions},$$

$$recall = \frac{TP}{TP + FN} = \frac{TP}{\#ground\ truth},$$

$$F1 = 2 \times \frac{precision\ .\ recall}{precision + recall}.$$

In order to extract each individual polygon, the property of obtaining a *contour* class was used, as explained in Section 3.4.2 using the *Difference with enlarged polygons* method, where the enlarge factor $a = 2.85$. This characteristic facilitates the imitation of ground truth images which makes the extracted polygons more similar to the ground truth polygons in terms of shape, formation and position. In addition, the Douglas-Peucker algorithm is set with the parameter $\epsilon = 4$ in order to simplify the polygons for the given resolution. Afterwards, the predicted polygons and the ground

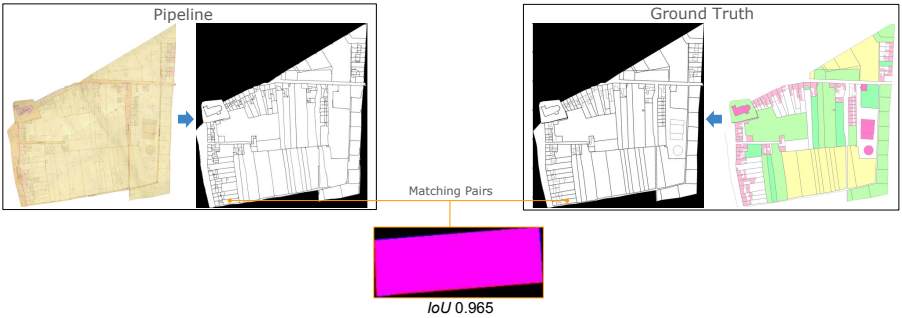


Figure 5.2.: The approach used for Geometrical evaluation: By generating *contours* with adjustable thickness, the pipeline is able to create very similar masks of ground truth images.

truth polygons should be matched and compared in pairs to obtain the *IoU* score of each pair. Finding each pair of polygons was accomplished by two main steps; First, the inner centroid point of the predicted polygon is calculated, then the polygon containing this centroid point is selected from the ground truth image. Finally, the *IoU* score of each pair is calculated.

Figure 5.2 illustrates the Geometrical evaluation approach that has been used to assess the quality of the extracted polygons. As mentioned previously, the values *TP*, *FN*, and *FP* were used to obtain recall, precision and F1-score metrics. These values were calculated by measuring the *IoU* score with the annotated ground truth polygons for different thresholds $\tau \in \{0.5, 0.7, 0.8, 0.85, 0.9\}$, as shown in Table 5.1. Similar to the evaluation method done by a previous work [3], the results can be divided into three main categories, namely, $\tau = 0.5$ (acceptable), $\tau = 0.7$ (good) and $\tau = 0.9$ (excellent). For the excellent case, where $\tau = 0.9$, the pipeline has achieved the precision and recall of 0.46 and 0.40, respectively, which is comparatively lower than the other thresholds. However, the consistent relationship between precision and recall is an indication of the robustness of the pipeline.
Furthermore, Fig. 5.1 provides a histogram of *IoU* scores of all extracted polygons. The most remarkable result to emerge from the data is that *IoU* score of 79.5% of extracted polygons ranged between [0.8 - 1.0]. In more details; 79.5% of the polygons have a range *IoU* between [0.8 - 1.0], 14.7% ranged between [0.50 - 0.80],

IoU	*TP*	Precision	Recall	F1
τ=0.50	6200	0.94	0.83	0.88
τ=0.70	5728	0.87	0.77	0.82
τ=0.80	5230	0.80	0.70	0.75
τ=0.85	4555	0.69	0.61	0.65
τ=0.90	3009	0.46	0.40	0.43

Table 5.1.: Quantitative evaluation of extracted polygons using different *IoU* thresholds to calculate precision, recall, and F1-score

Class	average *IoU*
background	0.80
building	0.86
water	0.28
Total	0.83

Table 5.2.: Average *IoU* scores for individual classes.

and the remaining 5.7% have *IoU* scores lower than 0.5, which concludes a good approximation of the polygons. Furthermore, the precision value of 0.87 for the good category indicates that most of the detected polygons at this threshold were detected correctly.

The average *IoU* for each individual class is calculated, as shown in Table 5.2. The classes *background* and *building* have an average *IoU* of 0.80 and 0.86, respectively. However, the average *IoU* for polygons with the *water* class is 0.28, which indicates low quality of segmentation. The most likely explanation for this rather contradictory result is due to the low number of *water* classed polygons in the dataset and due to its distinctive shape and form compared to other classes. Nonetheless, the mean *IoU* score of all extracted polygons, namely, the global segmentation efficiency is 0.83. In fact, one of the main advantages of the *IoU* score is that, it considers all shape properties of polygons, i.e. position, dimension and orientation. Furthermore, the *IoU* score tends to penalize individual cases of poor classification, which has strengthened the conviction that the pipeline provides good approximation for the polygons with such high *IoU* scores.
Furthermore, similar to the previous approach, the pipeline is evaluated on the three

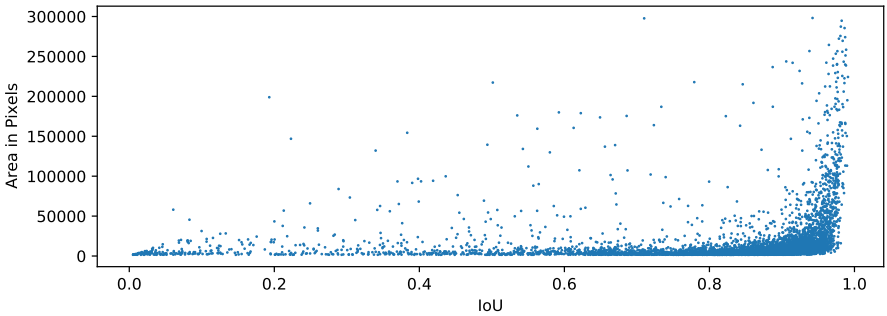


Figure 5.3.: The correlation between *IoU* score and polygon size.

cadastral maps that were not used for training. As shown in Appendix A, the pipeline is able to obtain very high precision and recall values for different *IoU* thresholds on the test experiments, as shown in Table A.1. Although the maps exhibit difficult and challenging artifacts for the segmentation and object recognition problem, the pipeline output has a very good approximation of the objects present on the maps with an average *IoU* of 0.85, where more than 82% of the polygons have a higher *IoU* than 0.8, as seen in histogram distribution of *IoU* in Fig. A.4. The figures A.1, A.2 and A.3 of the test experiments show that the pipeline succeeds in highlighting the most important contours and omitting text, numbers and paper degradation.
Finally, additional analysis were carried out on the extracted polygons. As shown in Figure 5.3, the correlation between polygon sizes and *IoU* scores reveals a significant fact about the used data. For instance, the higher the size of the polygon, the more likely it is to have a high *IoU* score. Thus, the results suggest that the use of higher resolution images in the pipeline will ultimately lead to a significant improvement in the pipeline performance.

5.3. Map Objects classifier Evaluation

For a comprehensive evaluation, the experiments on Map Objects classifier are divided into three different sets of input features. In the first experiment, the feature

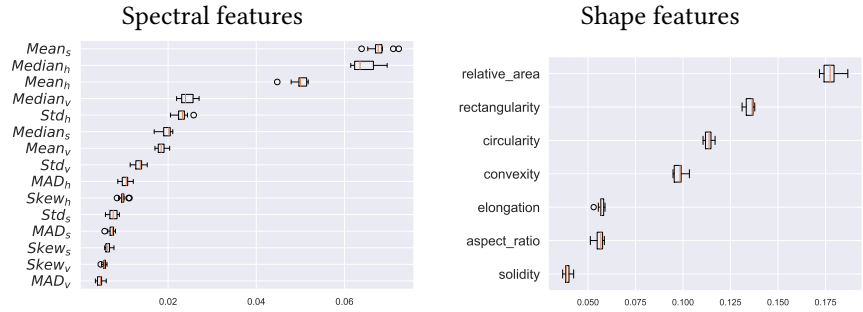


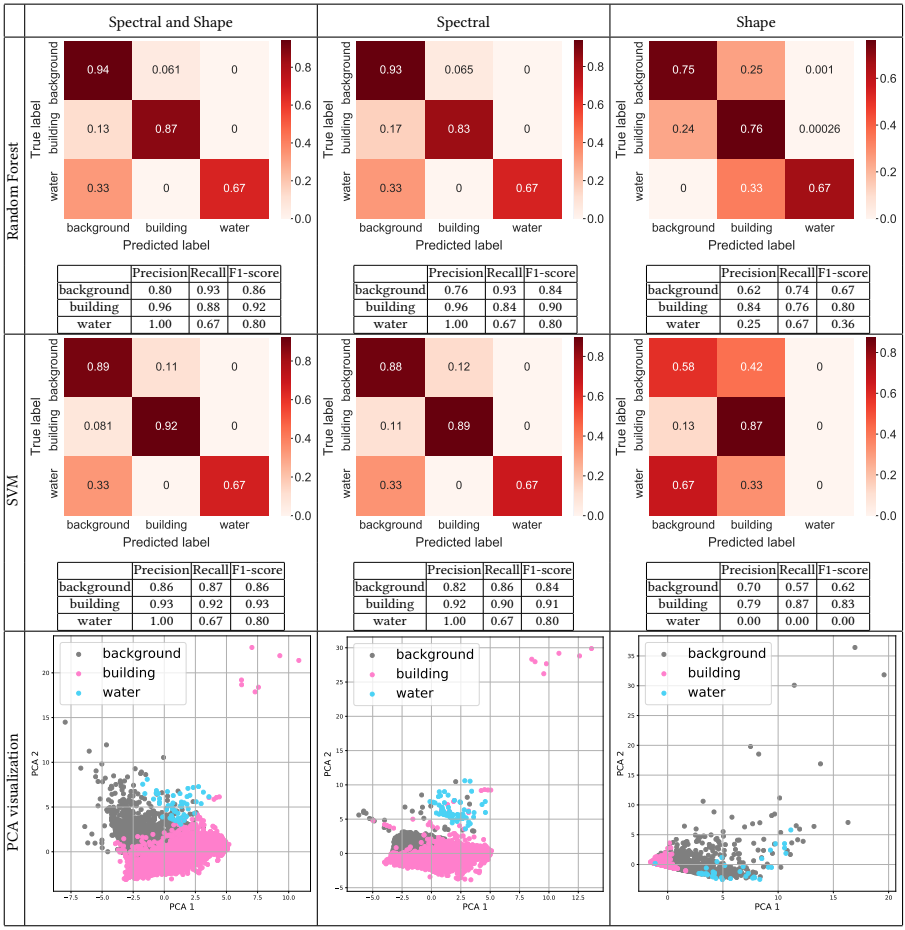
Figure 5.4.: Feature Importance for each of the spectral features and shape features.

vector includes only spectral features. In the second, it includes only shape features. In the last experiment, the vector includes both spectral and shape features. Furthermore, a detailed Feature Importance is performed on the Random Forest model to select relevant features depending on the ranking of relevancy. Since many of the images in the dataset have aging artifacts and pale colors, 60% of the dataset are used to train the models of the Map Objects classifier.

Feature analysis is performed on two sets of features i.e. Feature Permutation Importance for spectral features and shape features. Figure 5.4 illustrates the relevancy of spectral and shape features. The analysis reveals that the highest ranked spectral features are the *mean* and the *median* for all of HSV-channels and *standard deviation (Std)* for hue and value channels. Moreover, the most four relevant shape features are selected including *Relative Area*, *Rectangularity*, *Circularity* and *Convexity* corresponding to Equations 3.7, 3.3, 3.8 and 3.5, respectively.

In order to assess the performance of the Map objects classifier, precision, recall, and F1-score are calculated. Then the normalized confusion matrix with recall, precision and F1-score for each individual class is also included to determine the accuracy of each class. The comparison includes the three experiments with the three different sets of input features on the Map Objects classifier using the Random Forest and SVM, as shown in Fig 5.5. An additional visualization of each feature set is provided in the comparison to highlight the effect of combining spectral and shape features.

The 6576 filtered polygons of the Geometric evaluation are used onwards to evaluate the Map Objects Classifier. The polygons require further filtering since they contain



Random Forest – Spectral and Shape	Precision	Recall	F1-score
background	0.80	0.93	0.86
building	0.96	0.88	0.92
water	1.00	0.67	0.80

Random Forest – Spectral	Precision	Recall	F1-score
background	0.76	0.93	0.84
building	0.96	0.84	0.90
water	1.00	0.67	0.80

Random Forest – Shape	Precision	Recall	F1-score
background	0.62	0.74	0.67
building	0.84	0.76	0.80
water	0.25	0.67	0.36

SVM – Spectral and Shape	Precision	Recall	F1-score
background	0.86	0.87	0.86
building	0.93	0.92	0.93
water	1.00	0.67	0.80

SVM – Spectral	Precision	Recall	F1-score
background	0.82	0.86	0.84
building	0.92	0.90	0.91
water	1.00	0.67	0.80

SVM – Shape	Precision	Recall	F1-score
background	0.70	0.57	0.62
building	0.79	0.87	0.83
water	0.00	0.00	0.00

Figure 5.5.: Comparison of the results of Random Forest and SVM models using the three experiments performed with different features to compare the results, followed by a PCA visualization of the spectral and geometric features in the three experiments.

low quality segmented polygons. Therefore, only the polygons with a *IoU* score greater than 0.7 (the good category) are only included, leaving 5728 polygons, which include 1940 *background*, 3785 *building* and 3 *water* classed polygons.
The highest accuracies in both models are achieved by using both spectral and shape features, where Random Forest and SVM achieved 89% and 90% accuracy, respectively. Although the SVM model has a higher average accuracy, the Random Forest model can classify *background* class by 94%, which is the highest accuracy among the other classes in all experiments.
The most remarkable observation to emerge from the feature comparison is that the shape features provide a slightly better accuracy of *background* and *building* classed polygons, which is highlighted in the PCA visualization in Fig. 5.5. It should be noted that the Random Forest and SVM models in experiments using only spectral features have an average accuracy of 86% and 88%, respectively.
Since the Map objects classifier is dependent on the quality of extracted polygons, very few *water* classed polygons are included in the experiments, which, unfortunately, prevents further examination of *water* classed classifications and comparing the effect of spectral and shape features.

6 Conclusion and Outlook

This Chapter summarizes this thesis by giving a comprehensive overview of the handled subjects followed by some issues and suggestions for possible further research.

6.1. Summary

Historical maps, especially cadastral maps, are one of the most challenging data encountered in digitization of historical documents, their spatial characteristics are difficult to distinguish and the large amount of data contained on the map exacerbates the problem. In this thesis, the main focus is to obtain a modular automatic vectorization pipeline, that can fully automate the process of vectorization i.e. the raster-to-vector process on historical cadastral map images. First, the regions of the map were identified by estimating the Euclidean distance transformation with watershed segmentation, which floods the areas furthest from the boundaries till it reaches them. Afterwards, the polygons of identified regions were created and then refined using morphological operations and the Douglas-Peucker algorithm. The gaps and overlaps between the polygons were then eliminated using the *Difference with enlarged polygons* method, which can also generate contours as vector graphic with adjustable thickness. Lastly, the Map Object Classifier is trained to identify the individual polygons on the map, comparing Random Forest and SVM models with different sets of handcrafted features, which were designed to obtain shape and spectral information.
The pipeline was then evaluated in two parts i.e. Geometrical and Map objects Classifier evaluation. The Geometrical evaluation concludes that over 79% of the polygons obtained *IoU* higher than 0.8, despite the fact that the *IoU* metric is sensitive to error and deformation. The pipeline exhibits a limitation to identify small structures such as many *water* classed objects. However, the predicted polygons with larger areas on

the used maps tend to have higher *IoU* scores, which suggests the benefit of using a higher resolution images in further experiments. In addition, test experiments were conducted on three different maps which were not used in any form for training. As shown in Appendix A, the pipeline obtained very high precision and recall values for the test experiments. Despite the fact that the maps exhibit difficult and challenging artifacts such as text, geographical guidelines and paper degradation, the pipeline was able to reach an average *IoU* of 0.85 for a total of 544 polygons present on the maps.

For the Map Objects Classifier, a detailed and comprehensive evaluation was conducted. It can be concluded that the performance highly depends on the quality of the images and the colors in the original raster images. Three experimental comparisons were performed on Random Forest and SVM models; the first experiment includes only spectral features, the second includes only shape features, and the last experiment includes both spectral and shape features. Results show that the highest accuracies were achieved by using both spectral and shape features for both models. For instance, the shape features have slightly improved the classification accuracy when combined with spectral features.

In order to extend the usability of the pipeline, a web application is implemented, which provides an interactive visualization and variability of the parameters used in the pipeline.

The pipeline initiates a solution to automate the vectorization process of historical maps, it aims to create a modular, extensible application with satisfactory results. The clarity and resolution of the maps have a substantial impact on the pipeline results, which suggests further examinations with a wider range of labeled datasets.

6.2. Future Work

Aiming to obtain possible improvements in terms of performance, this section represents a few interesting research topics to investigate in the future. With a focus towards possible improvements in the performance achieved in this thesis, further research could be conducted in the following areas:

Higher spectral resolution of the dataset: hyperspectral document imaging acquisition could overcome some of the challenging problems of document image analysis including; document aging, incomplete contours and information retrieval from historical documents [32]. Generally, the spectral information contains RGB channels that cover the visible spectrum of wavelengths. Hyperspectral sensors; however, obtain more spectral information about the captured images. Therefore, having 10 to hundreds of channels of multispectral or hyperspectral images covering a wider spectrum of wavelengths would ultimately obtain more additional information about the image. This additional information could eventually improve watershed segmentation and Map Objects Classifier, which leads to a better pipeline performance.

Higher spatial resolution of the dataset: the results obtained in Chapter 5 on the correlations between polygon sizes and the quality of segmentation suggest that by using higher resolution sensors to capture the images, the pipeline performance would improve significantly.

Pipeline Enhancement: the pipeline could possibly be improved by a pre-processing module to clean the data, such as denoising algorithms. In addition, the modularity of the pipeline tolerates the replacement and extension of its components, which increases the possibility of testing other algorithms in terms of segmentation and object recognition to achieve higher performances.

A Appendix

For test experiments three maps of the dataset were excluded from the training in order to extend the evaluation of the pipeline on external data.

Figure A.1.: The results of raster input (top) to a vector output (bottom) of the pipeline on a fragment of *Mühlhausen of Flur 38-06*.

Figure A.2.: The results of raster input (top) to a vector output (bottom) of the pipeline on a fragment of *Mühlhausen of Flur 46-04*.

Figure A.3.: The results of raster input (top) to a vector output (bottom) of the pipeline on a fragment of *Mühlhausen of Flur 76-11*.

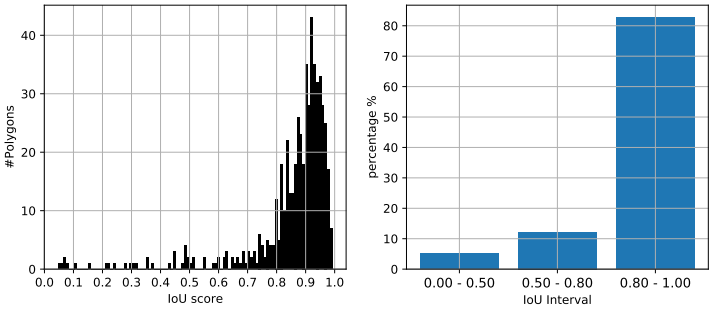


Figure A.4.: *IoU* scores distribution the of test experiments.

IoU	*TP*	Precision	Recall	F1
τ=0.50	519	0.95	0.87	0.91
τ=0.70	495	0.90	0.83	0.87
τ=0.80	452	0.83	0.76	0.79
τ=0.85	382	0.70	0.64	0.67
τ=0.90	273	0.50	0.46	0.48

Table A.1.: Quantitative results of the test experiments.

List of Figures

List of Tables

Bibliography

[1] J. Ignjatić, B. Nikolić, and A. Rikalović, "Deep learning for historical cadastral maps digitization: Overview, challenges and potential", 2018 (cit. on pp. 3, 6).

[2] S. Ares Oliveira, I. di Lenardo, and F. Kaplan, "Machine vision algorithms on cadaster plans", in *Premiere Annual Conference of the International Alliance of Digital Humanities Organizations (DH 2017)*, 2017 (cit. on pp. 6, 7, 17).

[3] S. A. Oliveira, I. D. Lenardo, B. Tourenc, and F. Kaplan, "A deep learning approach to cadastral computing", 2019 (cit. on pp. 7, 42).

[4] S. Beucher, "The watershed transformation applied to image segmentation", *Scanning Microscopy Supplement*, pp. 299–314, Jan. 1992 (cit. on pp. 7, 8, 10).

[5] *Esri*. [Online]. Available: `https://support.esri.com/en/white-paper/279` (cit. on p. 7).

[6] *The geojson format*. [Online]. Available: `https://tools.ietf.org/html/rfc7946` (cit. on p. 7).

[7] A. K. Knowles, A. Hillier, *et al.*, *Placing history: how maps, spatial data, and GIS are changing historical scholarship*. ESRI, Inc., 2008 (cit. on p. 8).

[8] R. Bandara, "Image segmentation using unsupervised watershed algorithm with an over-segmentation reduction technique", *arXiv preprint arXiv:1810.03908*, 2018 (cit. on p. 9).

[9] R. Fabbri, L. D. F. Costa, J. C. Torelli, and O. M. Bruno, "2d euclidean distance transform algorithms: A comparative survey", *ACM Computing Surveys (CSUR)*, vol. 40, no. 1, pp. 1–44, 2008 (cit. on p. 9).

[10] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation", in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015, pp. 234–241 (cit. on pp. 10, 11, 22).

[11] J. Redmon and A. Farhadi, "Yolov3: An incremental improvement", *arXiv preprint arXiv:1804.02767*, 2018 (cit. on pp. 11, 40).

[12] X. Wu, V. Kumar, J. R. Quinlan, J. Ghosh, Q. Yang, H. Motoda, G. J. McLachlan, A. Ng, B. Liu, S. Y. Philip, *et al.*, "Top 10 algorithms in data mining", *Knowledge and information systems*, vol. 14, no. 1, pp. 1–37, 2008 (cit. on p. 12).

[13] J. Ali, R. Khan, N. Ahmad, and I. Maqsood, "Random forests and decision trees", *International Journal of Computer Science Issues (IJCSI)*, vol. 9, no. 5, p. 272, 2012 (cit. on p. 13).

[14] G. Caiola and J. P. Reiter, "Random forests for generating partially synthetic, categorical data.", *Trans. Data Priv.*, vol. 3, no. 1, pp. 27–42, 2010 (cit. on pp. 13, 35).

[15] L. Breiman, "Random forests", *Machine learning*, vol. 45, no. 1, pp. 5–32, 2001 (cit. on p. 13).

[16] N. Apostolof and A. Zisserman, "Who are you?-real-time person identification.", in *BMVC*, Citeseer, 2007, pp. 1–10 (cit. on p. 14).

[17] A. Caglayan, O. Guclu, and A. B. Can, "A plant recognition approach using shape and color features in leaf images", in *International Conference on Image Analysis and Processing*, Springer, 2013, pp. 161–170 (cit. on pp. 14, 33).

[18] A. Bosch, A. Zisserman, and X. Munoz, "Image classification using random forests and ferns", in *2007 IEEE 11th international conference on computer vision*, Ieee, 2007, pp. 1–8 (cit. on pp. 14, 15).

[19] R. Burbidge and B. Buxton, "An introduction to support vector machines for data mining", *Keynote papers, young OR12*, pp. 3–15, 2001 (cit. on pp. 15, 35).

[20] D. G. Lowe, "Object recognition from local scale-invariant features", in *Proceedings of the seventh IEEE international conference on computer vision*, Ieee, vol. 2, 1999, pp. 1150–1157 (cit. on p. 15).

[21] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection", in *2005 IEEE computer society conference on computer vision and pattern recognition (CVPR'05)*, IEEE, vol. 1, 2005, pp. 886–893 (cit. on p. 15).

[22] S. Klemm, X. Jiang, and B. Risse, "Deep distance transform to segment visually indistinguishable merged objects", Oct. 2018 (cit. on pp. 19, 22).

[23] B. Zitova and J. Flusser, "Image registration methods: A survey", *Image and vision computing*, vol. 21, no. 11, pp. 977–1000, 2003 (cit. on p. 20).

[24] S. Suzuki *et al.*, "Topological structural analysis of digitized binary images by border following", *Computer vision, graphics, and image processing*, vol. 30, no. 1, pp. 32–46, 1985 (cit. on p. 26).

[25] U. Ramer, "An iterative procedure for the polygonal approximation of plane curves", *Computer Graphics and Image Processing*, vol. 1, no. 3, pp. 244–256, 1972, ISSN: 0146-664X. DOI: https://doi.org/10.1016/S0146-664X(72)80017-0. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0146664X72800170 (cit. on p. 26).

[26] B. Xiong, S. Oude Elberink, and G. Vosselman, "Footprint map partitioning using airborne laser scanning data", *ISPRS Annals of Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. III-3, pp. 241–247, Jun. 2016. DOI: 10.5194/isprs-annals-III-3-241-2016 (cit. on p. 27).

[27] J. Hershberger and J. Snoeyink, "Speeding up the douglas-peucker line-simplification algorithm", CAN, Tech. Rep., 1992 (cit. on p. 27).

[28] J. M. Keller, M. R. Gray, and J. A. Givens, "A fuzzy k-nearest neighbor algorithm", *IEEE transactions on systems, man, and cybernetics*, no. 4, pp. 580–585, 1985 (cit. on p. 27).

[29] I.-K. Lee, M.-S. Kim, and G. Elber, "Polynomial/rational approximation of minkowski sum boundary curves", *Graphical Models and Image Processing*, vol. 60, no. 2, pp. 136–165, 1998 (cit. on p. 31).

[30] M. Bostock, *D3.js - Data-Driven Documents*. [Online]. Available: https://d3js.org/ (visited on 10/28/2020) (cit. on p. 38).

[31] *Istg*. [Online]. Available: https://www.uni-muenster.de/Staedtegeschichte/ (cit. on p. 39).

[32] R. Qureshi, M. Uzair, K. Khurshid, and H. Yan, "Hyperspectral document image processing: Applications, challenges and future prospects", *Pattern Recognition*, vol. 90, pp. 12–22, 2019 (cit. on p. 51).

Declaration of Academic Integrity

I hereby confirm that this thesis on

Automated vectorization of historical maps
using state-of-the-art Computer Vision techniques

is solely my own work and that I have used no sources or aids other than the ones stated. All passages in my thesis for which other sources, including electronic media, have been used, be it direct quotes or content references, have been acknowledged as such and the sources cited.

Mhd Sufian Zaabalawi, Münster, November 9, 2020

I agree to have my thesis checked in order to rule out potential similarities with other works and to have my thesis stored in a database for this purpose.

Mhd Sufian Zaabalawi, Münster, November 9, 2020