



UNIVERSITY OF TWENTE.

A Contribution to Computer-Assisted Deciphering of Labels from Scanned Maps

Master Thesis by Christian Haack

Supervisors:
Nikolas Prechtel from TU Dresden
R. Knippers from University of Twente

Submitted: 10.September 2018



Joined Cartography Master Program
by TU Munich, TU Vienna, TU Dresden
and University of Twente

Disclaimer:

Herewith I declare that I am the sole author of the thesis named „A Contribution to Computer-Assisted Deciphering of Labels from Scanned Maps” which has been submitted to the study commission of geosciences today. I have fully referenced the ideas and work of others, whether published or un-published. Literal or analogous citations are clearly marked as such.

Dresden, 5.9.2018:

Christian Haack

Abstract:

This thesis gives a contribution to text deciphering from scanned maps, by analyzing the properties of text within historic maps. It extracts factors that lead to improved algorithms for text extractions on these kinds of maps. Furthermore the thesis gives an overview over common methods of text extraction in general and makes an effort to create a python based solution for text extraction on historic maps.

Kurzfassung:

Diese Arbeit stellt einen Beitrag zum Entschlüsseln von Text von gescannten Karten, indem es die Eigenschaften von Text aus historischen Karten analysiert. Dabei werden Faktoren genannt, die zukünftige Algorithmen zum automatisierten Lesen von Text verbessern können. Desweiteren gibt die Arbeit eine Übersicht über bekannte Methoden der Textextraktion. Dabei wird auch eine Python Lösung für die Textextraktion von historischen Karten angestrebt.

Table of Contents

1. Introduction.....	1
1.1 Research objectives.....	2
1.2 Research questions	2
1.3 Overview.....	2
2. Analysis of a historical map.....	5
2.1 Parameters for map selection.....	5
2.2 Map selection analysis	6
2.3 Summary table	20
2.4 Fonts and toponyms	23
2.4.1 History of map fonts.....	23
2.4.2 Parameters of fonts.....	24
2.4.3 Placements	26
2.5 Noise and scanning artifacts.....	28
2.6 Useful findings for text extraction.....	29
3. Text extraction algorithms	30
3.1 Introduction to text extraction.....	30
3.1.1 What is text?	30
3.1.2 How to find text?	31
3.2 DAR System Part 1: Pre-processing and image filtering.....	32
3.3 DAR System Part 2 - Line extraction.....	34
3.4 DAR System Part 2 - Text extraction.....	34
3.4.1 Filtering by preselected parameters	34
3.4.2 Neural networks	36
4. Text recognition algorithms and OCR.....	37
4.1 OCR.....	37
4.2 Post processing.....	39
4.3 Gazetteers.....	40
5. Application development	42
5.1 Python.....	42
5.2 Free libraries.....	43
5.3 Used workflow	43
5.4 Step by step	44
5.4.1 Filter templates:	44

5.4.2 Partmatrix similarity processor	48
5.4.3 Textgroup finder	51
5.4.4 OCR and gazetteer comparison.....	53
5.5 Results	55
5.6 Further ideas and future work:	59
5.6.1 Planned additions to the program:.....	59
5.7 User interface.....	61
5.8 Target group.....	61
6. Conclusion and further research.....	62
Sources:	i
Appendix.....	iv

Table of Figures

FIGURE 1: SCREENSHOT OF HTTP://WWW.DEUTSCHEFOTOTHEK.DE/DOCUMENTS/OBJ/70302464 FROM AUGUST 2018 - SHOWING A ARCHIVED MAP OF THE VIRTUELLES KARTENFORUM 2.0 AND THE LIST OF TOPONYMS TAGS	3
FIGURE 2: MAP OF THE SCRIPTS IN EUROPE (1901) (ENLARGED VERSION IN THE APPENDIX)	23
FIGURE 3: IMAGE OF FRAKTUR SCRIPT	23
FIGURE 4: SCREENSHOT FROM: KARTE VON SACHSEN (KÖNIGREICH), 1:157 281, LITHOGRAPHIE, 1851, HTTP://WWW.DEUTSCHEFOTOTHEK.DE/DOCUMENTS/OBJ/90010910/DF_DK_0011479_0002	25
FIGURE 5: IMAGE SHOWING AMBIGUOUS TEXT. ("TEACH" OR "LEARN").....	31
FIGURE 6: DIAGRAM FOR THE DOCUMENT ANALYSIS AND RECOGNITION SYSTEM	31
FIGURE 7: RESULT TABLE FROM (JADERBERG, SIMONYAN, VEDALDI, & ZISSERMAN, 2015) FOR DIFFERENT OCR METHODS (ROWS) OVER DIFFERENT SETS OF DATA (COLUMNS).....	39
FIGURE 8: CUT-OUT TEXT FROM THE MAP PLAN VON DRESDEN. BLATT 27 [WURGWITZ, ALTFRANKEN, PESTERWITZ], 1939	40
FIGURE 9: SCREENSHOT OF THE SIMPLE GUIDED USER INTERFACE OF THE PYTHON IMAGE SIMILARITY COMPARATOR	44
FIGURE 10: 10 IMAGES, OF THE MAP PLAN VON DRESDEN. BLATT 27 [WURGWITZ, ALTFRANKEN, PESTERWITZ], 1939, FILTERED WITH THE TEMPLATE VISIBLE IN THE TOP LEFT CORNER OF EACH IMAGE. RED SPOTS SHOW HITS OVER THE GIVEN THRESHOLD.....	47
FIGURE 11: FILTERED IMAGE OF THE MAP PLAN VON DRESDEN. BLATT 27 [WURGWITZ, ALTFRANKEN, PESTERWITZ], 1939 (ORIGINAL AS CUT IN, IN THE TOP LEFT CORNER) THE LIGHTER THE FILTERED IMAGE, THE HIGHER THE SIMILARITY TO THE TWO-STROKE-TEMPLATE	49
FIGURE 12: RESULTING MATRIX OF THE PARTSUM METHOD. THE LIGHTER THE PIXEL, THE HIGHER IS THE SIMILARITY TO THE TWO-STROKE-TEMPLATE	50
FIGURE 13: RESULT OF THE TEXTAREA-GROUPING METHODE, COLORED AFTERWARDS. EACH COLOR REPRESENTS ONE FOUND TEXT GROUP	52
FIGURE 14: ORIGINAL MAP: PLAN VON DRESDEN. BLATT 27 [WURGWITZ, ALTFRANKEN, PESTERWITZ], 1939 AND THE EXTRACTED TEXT AREAS BELOW.....	55
FIGURE 15: ORIGINAL MAP: SECTION KÖTZSCHENBRODA AUS: TOPOGRAPHISCHE KARTE (ÄQUIDISTANTENKARTE) SACHSEN	56
FIGURE 16: A RANDOM SELECTION OF THE TEXT AREAS EXTRACTED FROM THE MAP SECTION KÖTZSCHENBRODA AUS: TOPOGRAPHISCHE KARTE (ÄQUIDISTANTENKARTE) SACHSEN.....	57

Abbreviations:

DAR - Document Analysis and Recognition

GIS – Geographic Information System

MSER - Maximally stable extremal regions

OCR – Optical Character Recognition

RANSAC - Random sample consensus

SLUB – Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden (Saxonian State and University Library)

1. Introduction

Whenever a human looks at a map, he can immediately decipher everything he sees. He points out the parts that show text without even taking time to think about it. Because of this ability deciphering drawn maps had been a job for a human tracer for the longest time and to this day a human is doing this job more reliable than a computer.

The problem is not to read the text itself: Optical Character Recognition (OCR) is solved by algorithms and computers already succeeded at it: The most common commercial software packages have an OCR accuracy between 97% and 99% (Holley) and they are much faster at it, than any human could be. But those results are reached only on simple documents, which contain mostly text, ordered in straight, horizontal lines, without too much noise and a uniform font.

Maps on the other hand are nothing like that: They contain mostly graphical features, while the text is only sparsely scattered over the map. The text is often horizontal, but not always, especially for line features like roads or rivers and it is disturbed by features running under it.

This provides a challenge for computer algorithms to separate text from its background to read toponyms and further map information.

Historical maps show natural and human-induced changes on earth and are often the only information source about earth. Based on geodetic techniques they preserve unique information. Digital map archives (like the Virtuelles Kartenforum 2.0¹) have been established to store the maps using software and hardware technologies. (Yao-Yi Chiang, 2014) Usually they use archive systems to keep the data manageable. These archives have been created in a manual fashion. This time consuming procedure can be replaced by automated systems.

Since the early 1980s, scientists from many fields, including computer science and geography, have been working on computational methods for the extraction and recognition of toponyms and other features from archived digitized images of maps. If it would succeed with a high enough tolerance for errors, the research would benefit numerous research fields in the spatial, social, environmental, and health sciences. (Yao-Yi Chiang, 2014)

The increase of powerful computer vision techniques and the increase in the volume of images produced over the last decade has caused a rapid development of text spotting methods. (Jaderberg, Simony, Vedaldi, & Zisserman, 2015)

¹ <https://kartenforum.slub-dresden.de/> (acc. 9.2018)

1.1 Research objectives

Beside the theoretical evaluation of the topic of text extraction of scanned maps, the result of the thesis includes a python program that provides a user the necessary tools to extract text from a map. This might include a series of image filters to pre-process the map, a visual representation of the results of the text extraction and the ability to correct faulty results and the reconciliation with (pre-filtered) gazetteers or dictionaries. To make the program as user friendly as possible a User Interface shall be provided.

The program shall be as automated as possible. Because of expected inaccuracies of the algorithm and the time constraints on the thesis itself, manual approaches will also be considered and if necessary preferred to automation.

1.2 Research questions

- What will a logical set of criteria steering the map selection for the thesis?
- What are the relevant published algorithms for text – “noise” separation?
- What are the relevant published algorithms for text recognition?
- Which open-source libraries can be used to implement the algorithm?
- How can extraction results be coupled to gazetteers of toponyms?
- How to assess the quality of the final tool and the output result?
- What will be a suitable visualisation to inform the user on the extraction results?

1.3 Overview

The topic of automated reading of text has challenged multiple disciplines of science. Facilities that store numerous amounts of analog data, historic artifacts, like maps or paper scrolls and printed knowledge like books are looking for modern ways of preserving their exhibits. Most libraries and archives already digitized their data via scanning, which created an enormous amount of data points, that need some kind of sorting system. Text extraction and recognition can be used to increase the searchability and access of the digitized version. (Korb, 2008)

Usually done by tags based on the title, the scale and the authors of the work.

For books or papers these tags may be enough, but maps usually contain tagable content beyond these. The cities or regions depicted in the map will be useful for historical or geological research within a small region. Looking for a specific village can be difficult; especially in a map series like [Topographische Karte (Äquidistantenkarte) Sachsen] - 1:25 000. - Leipzig : Giesecke & Devrient (Link:

<http://www.deutschefotothek.de/documents/obj/70302464>) with 198 mapsheets.

A detailed tagging for each map, including all depicted village names, and searchable via a software interface can reduce the work necessary to get information.

At the Virtuelles Kartenforum of the SLUB (State and University Library) in Dresden, this is already partly archived: Maps contain tags of the largest villages and cities within the map: (Figure 1). This has been done in laborious manual work and is continuously done for additional maps. (Mendt, 2015)

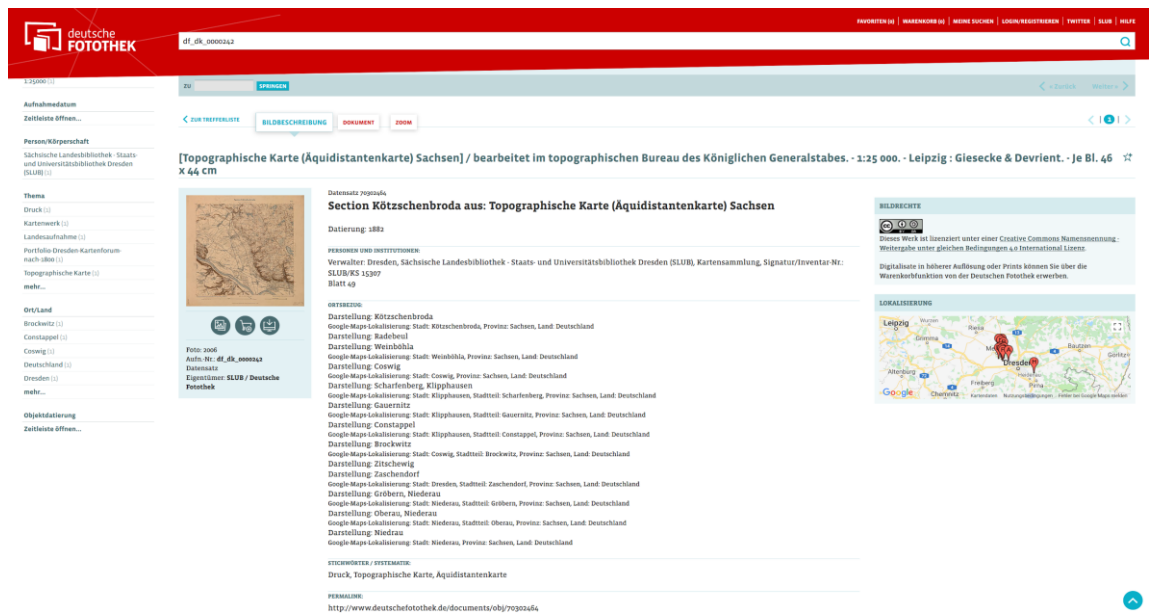


Figure 1: screenshot of <http://www.deutschefotothek.de/documents/obj/70302464> from August 2018 - showing a archived map of the Virtuelles Kartenforum 2.0 and the list of toponyms tags

But whether physical or digital, searching maps for an answer to a specific question could be laborious. The Library of Congress classification for maps on the other hand includes only the map's region along with the subject, date, and source reference system. (Weinman J., Toponym Recognition in Historical Maps by Gazetteer Alignment, 2013)

The same method can be useful to other companies that use or produce (old) printed maps. For example map publishers, which want to archive their old content.

But the topic of extracting text from analogue environments is researched much broader than just for printed maps. Computer science is looking for software solutions regarding this, too, as they are looking for systems for the extraction and recognition of text in a live (recorded) environment. For example the recognition of streets signs, for navigation or driver assistance systems, the live translation of text recorded by the users camera, or the automated transformation of books (and other documents) into digital media.

The article (Yao-Yi Chiang, 2014) presents an overview of existing map processing techniques, with the goal of bringing together the past and current research efforts in this interdisciplinary field, to characterize the advances that have been made, and to identify future research directions and opportunities.

Most algorithms for map reading can be separated into two different stages: The first is the word detection stage, wherein the areas of a map, which contain text, will be separated from the remaining features. The second stage is the word recognition stage (Jaderberg, Simonyan, Vedaldi, & Zisserman, 2015), where the algorithm tries to read the meaning of the text.

The stages itself cannot be solved by a simple one step algorithm. Although algorithms archive a higher accuracy, when they integrate the detection and recognition stage into an "end-to-end" text recognition system (Qixiang Ye, 2015), because they can use the results of the respective other to improve themselves.

For example is it advantageous to use found and recognized words for further detection, as those words may appear again in the document. This will be supported by a gazetteer - a database for toponyms, which can be used to narrow down the words the algorithm has to look out for.

Additionally it is vital for the design of the algorithms to understand each parameter of text. (Qixiang Ye, 2015) concluded already that: "Text is a hybrid of edges, CCs [connected components (Note from the author)], strokes and texture". With this assumption in mind, several hybrid approaches have already been proposed for text detection. "Text is a character composite", (Qixiang Ye, 2015) continues and notes, that this may be the more precise conclusion. He - and many others - noticed that characters are patterns, with a strict definition, developed over complex languages. (Dai, Liu, & B., 2007) names effective methods, that use this assumption.

We can adjust those assumptions directly to historical maps. Algorithms that specifically target maps are still lacking in accuracy (often less than 80 percent) and recognition rates (often less than 60 percent) over plain text recognition algorithms, like OCR. They typically achieve recognition rates over 99 percent on scanned documents. (Qixiang Ye, 2015)

This is caused by the specific underlying semantics of maps. Toponyms ("placename" from greek: τόπος (*tópos*) = place und ὄνομα (*ónoma*) = name) follows cartographic rules, which means, that the text is not just simply ordered in lines. Labels can vary in orientations, they can appear curved and do often overlap with other graphical elements. (Yao-Yi Chiang, 2014)

In the following chapter, this thesis is going to take a look at what are parameters of text in historical maps. Which parameters would a algorithm - of whichever kind - have to look out for. What might be causing problems for an algorithm?

2. Analysis of a historical map

The first step towards an algorithm that can decipher an historic map is to understand the composition of these maps. This work will focus on the toponyms within the maps, mostly village and city names. Therefore a sample of maps has been chosen and analyzed based on toponym characteristics.

2.1 Parameters for map selection

Besides the toponym characteristics, there are other parameters that influence the map selection:

The thesis will only look at topographic maps, as thematic maps are rare among historical maps and add an additional layer of complexity that will not be analyzed here.

The maps will be chosen by the date they were created. Maps qualify as “historic” for my selection if they are older than 1950.

Another parameter comes from a technological standpoint: The maps shall be produced by printing methods. Digitally created maps can be excluded, as there usually already is a digitized version. In this case a digitization algorithm would not be necessary.

Hand drawn and handwritten maps will be included as extreme examples in this map selection. But the difficulty to decipher handwritten fonts is higher than with printed fonts and will not be attempted by the algorithm, developed within this thesis.

The graphic styles varied a lot in historic maps as there was no standard design to follow and each artist and printer used his own solutions to map design. The selection shall try to use maps of varying styles.

Maps can only be chosen, if there is a scanned version of them available. The selection will be limited to the collection of the “Digitales Kartenforum” of the SLUB in Dresden. This limits the area, where the maps are produced and which area they show. The SLUB focuses on maps made in Germany and portray German territory.

Based on the named characteristics a selection of maps is downloaded. The selection is made by random, but will exclude maps of similar age and similar style.

In one case I selected the Topographische Karte of the Bureau des Königlichen Generalstabes - 1:25.000, because this topographical map range has been produced in numerous years from 1882 to 1922. The graphic style changes slightly during this time and gives an interesting timeline of maps of that age. (compare next chapter maps 7-11)

The scale of the selected maps ranges from 1:10 000 to 1:100 000, because larger scale maps mostly contain streetnames and location toponyms, which are harder to read, due to the smaller font and their adjustment to graphical objects (streetnames follow streets for example) and there is no sufficient toponym gazetteer for these.

Maps of smaller scale (e.g. world maps) contain mostly country names - if at all that many names to decipher. As these maps do not depict much geographical or environmental change, tagging the toponyms even automatically does not provide aid to the user. The country names and the shape of the continents did undergo only minor changes.

Maps within the scale range however contain mostly small city and village names (among location toponyms and some street names), which can easily be found in the most common gazetteers for cross references. Also the amount of toponyms which are horizontally oriented and less disturbed is higher than on maps of other scales.

2.2 Map selection analysis

With the selected parameters in mind thirteen maps have been selected by random for the analysis. This chapter will go through them one by one. The first two of the set are too old and especially challenging for an algorithm and showcase the most extreme cases for text detection.

As the maps are from a german map provider as well as mostly produced in Germany, the map titles and the description will be in german:

High resolution versions of the maps can be found in the appended digital medium.

Map 1: Scultetus: Karte von Sachsen, ca. 1:1 000 000, Kupferstich, 1596



Datensatz 70403113

Beschreibung: Misniae Et Lvsatiae Tabvla / Descripta à M. Bartholomeo Sculteto Gorlitz.

Q.. - [Ca. 1:1 000 000]. - [Köln] : [Johann Bussemacher] ; [Köln] : [Lambert Andrea], [1596].

- 1 Kt. : kolor. Kupferst. ; 28 x 20 cm

Datierung: 1596

Personen und Institutionen:

Urheber: **Scultetus, Bartholomäus**

Urheber: **Quad, Matthias**

Urheber: **Bussemacher, Johann**

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek

Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS A13539

Material / Technik / Maße:

Kupferstich

Maßstab: 1:1000000

Permalink: <http://www.deutschefotothek.de/documents/obj/70403113>

Toponym analysis:

- ☐ Handwritten style - very curlicue
- ☐ some color (yellow cities, green forest, blue water)
- ☐ wrong city names: Liebe-Löbau
- ☐ orientation of toponyms varies

Map 2: Sachsenkarte (MEDITULLIUM ELECTORATUS SAXONICI.)



Datensatz 88967010

Datierung: 1680

Personen und Institutionen:

Urheber: Bergen, Melchior, Verleger

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek
Dresden (SLUB), Signatur/Inventar-Nr.: Hist.Sax.A.132

Material / Technik / Maße:

Material: Papier, Kupferstich

Maße: 18,6 x 15,5 cm

Bezüge:

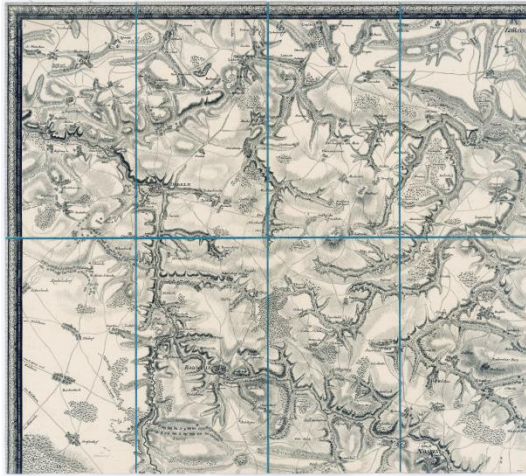
aus: : Beutel, Tobias, Cimelium Geographicum Tripartitum, Oder Dreyfaches
Geographisches Kleinod, Darinnen begriffen I. Richtig-gesuchte und verfassete
Longitudines und Latitudines aller groß- und kleinen Städte ... II. Milliographia, oder
Meilen-Beschreibung ... Dabey ein sonderbar Rosetum Geographicum, oder
Geographischer Rosen-Garten, III. Seminarium Geographicum, Geographischer Pflantz-
Garten, Buch (MEDITULLIUM ELECTORATUS SAXONICI.), SLUB, Hist.Sax.A.132

Permalink: <http://www.deutschefotothek.de/documents/obj/88967010>

Toponym analysis:

- ☐ very dense toponyms
- ☐ barely readable
- ☐ black fonts on distinguishable background
- ☐ city signatures of similar size and color to toponyms
- ☐ old blackletter font

Map 3: Petri, I.: Karte von Dresden und eines Teiles vom Kurfürstentum Sachsen, ca. 1:32 000, Kupferstich, 1762 Datensatz 70403115



Beschreibung: Accurate Situations-Card Von Einem Theile Des Churfürstenthums Sachsen, Und Hauptsachlich Von Den Gegenden 4 Bis 5 Meilen Aus Dem Centro Um Die Haupt- Und Residenz-Stadt Dresden, [...] Jahren des jetzigen Krieges 1759 und 1760 in dieser deutlichen Carte gebracht und verzeichnet worden durch Petri, und einige bei sich gehabte Preussische Ingen. Officiers. J. D. Schleuen fec.. - Auch nachhero durch eben demselben in denen folgenden Campagnen 1761 und 1762 an denen Örtern, wo man zuvor vorm Feind nicht kommen können, gänzlich verbessert und vollständig gemacht worden.. - [Ca. 1:32 000]. - Berolini : J. D. Schleuen, 1762. - 1 Kt. in 12Theilen : Kupferst. ; Gesamtgr. 233 x 158 cm, je Bl. versch. Gr.

Datierung: 1762

Personen und Institutionen:

Urheber: **Petri, Isaak Jakob von**

Urheber: **Schleuen, Johann David**

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS A16815-A16826

Toponym analysis:

- ☐ Schraffe hachures disturb text
- ☐ thick coordinate grid
- ☐ very dense topographic depiction
- ☐ toponyms in the middle of dense area signatures
- ☐ curvy rivers and creeks

Map 4: Umgebungskarte von Bremen, 1:40 000, kolor. Kupferst., um 1798



Datensatz 70302389

Beschreibung: Karte des Gebiethes der Reichs- und Hanse-Stadt Bremen : wie auch derjenigen Dörfer deren Landeshoheit im Jahre 1741, unter Vorbehalt verschiedener Gerechtsame, an Chur-Braunschweig abgetreten worden / nach trigonometrischen Vermessungen entworfen von C. A. Heineken. G. H. Tischbein sc.. - [Ca. 1:40 000]. - Bremen, 1798. - 1 Kt. : kolor. Kupferstich ; 63 x 47 cm

Datierung: 1798

Personen und Institutionen:

Urheber: **Tischbein, Georg H**, Stecher

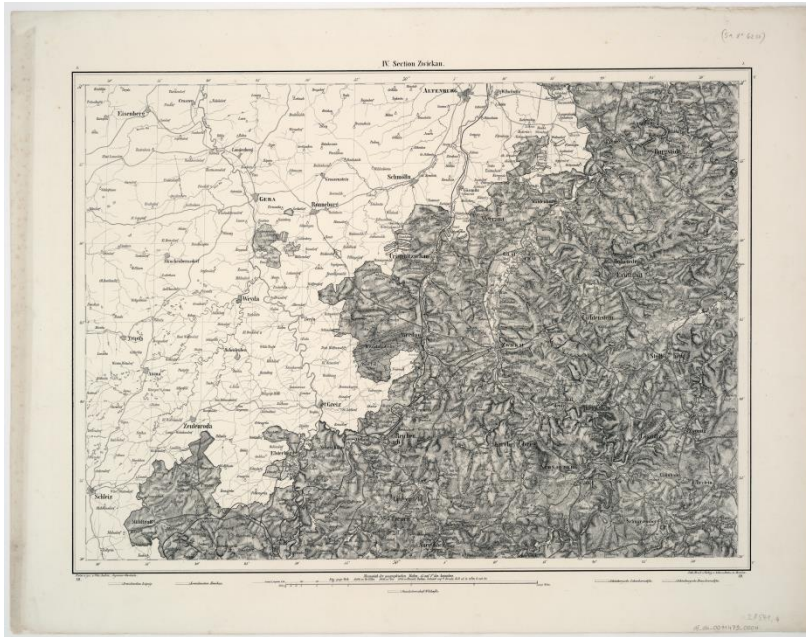
Zusammenhang: **Heineken, Christian Abraham**

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS A19352

Toponym analysis:

- ☐ curved, skewed toponyms with large spacing
- ☐ text color differs from map objects
- ☐ handwritten font

*Map 5: Karte von Sachsen (Königreich), 1:157 281, Lithographie, 1851
(Blatt 1)*



Datensatz 90010910

Beschreibung: Topographisch-orographische Special-Karte des Königreiches Sachsen : in 9 Blättern, nach den neuesten Quellen der Militär- und Finanz-Plankammer / entworfen und gezeichnet von Otto Andree. - 1:157 281. - Dresden : Adler & Dietze, 1851. - 1 Bl., 9 Kt. : schwarz/weiß Lithogr. ; 64,5 x 50 cm. Enth.: 1 Übersichtsnetz. - Section I - VIII. - Sec. I. u. d. T.: Leipzig. - Sec. II. u. d. T.: Dresden. - Sec. III. u. d. T.: Bautzen. - Sec. IV. u. d. T.: Zwickau. - Sec. V. u. d. T.: Chemnitz. - Sec. VI. u. d. T.: Zittau m. Profil d. Zittau-Löbauer Eisenbahn. - Sec. VII. u. d. T.: Plauen. - Sec. VIII. u. d. T.: Wiesenthal m. Übersichtskarte d. vorzüglichsten Höhen- und Thalpunkte d. Königreiches Sachsen. - Ortsverz. zu einz. Sektionen

Datierung: 1851

Personen und Institutionen:

Urheber: Andree, Otto, Kartenzeichner

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS 28541,1-28541,9 & 28541,1 & 28541,2 & 28541,3 28541,4 & 28541,5 & 28541,6 & 28541,7 & 28541,8 & 28541,9 & 13519

Toponym analysis:

- ☐ monochrome
- ☐ very dense texture
- ☐ text intersected by many lines and area textures
- ☐ Old names: Kosswig instead of Coswig
- ☐ very thin and handwritten font

Map 6: Topographische Special-Karte von Deutschland und den angrenzenden Staaten, 1:200 000, Kupferstich, 1806 bis ca. 1889, Bl. 1-150



Datensatz 90011133

148: Dresden, 1879

Beschreibung: 148 : Dresden / Entw. und gez. von F. Handtke. Situation u. Schrift gest. v. F. W. Kiewer, Berge v. Carl Fischer. - Rev. 1879. - [1:200 000]. - [Glogau] : [Flemming], 1879. - 1 Kt. : Stich. - Maßstabsangabe in graph. Form (Kilometer, geogr. Meile)

Datierung: 1879

Personen und Institutionen:

Urheber: Handtke, Friedrich, Kartograf

Urheber: Kiewer, Friedrich W, Kartenstecher

Urheber: Fischer, Carl, Kartenstecher

Verwalter: Signatur/Inventar-Nr.: SLUB/KS 5074

Toponym analysis:

- ☐ black font above black line signatures
- ☐ very striking train lines
- ☐ serif font
- ☐ various area textures behind toponyms

Map 7: [Topographische Karte (Äquidistantenkarte) Sachsen] / bearbeitet im topographischen Bureau des Königlichen Generalstabes. - 1:25 000. - Leipzig : Giesecke & Devrient. - Je Bl. 46 x 44 cm



Datensatz 70302464

Section Kötzschenbroda aus: Topographische Karte (Äquidistantenkarte) Sachsen

Datierung: 1882

Personen und Institutionen:

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek
Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS 15307

Druck, Topographische Karte, Äquidistantenkarte

Permalink: <http://www.deutschefotothek.de/documents/obj/70302464>

Blatt 49

Toponym analysis:

- ☐ no color difference between text and other map objects (both are black)
- ☐ toponyms are intersected by lines of similar width
- ☐ text is in front of colored areas (especially dark brown hill depiction)
- ☐ small tree signatures, which are similar to text
- ☐ no text halo
- ☐ slab serif font (I could not identify a digital font, that has this kind of ear over the small g)

Map 8: [Topographische Karte (Äquidistantenkarte) Sachsen] / bearbeitet im topographischen Bureau des Königlichen Generalstabes. - 1:25 000. - Leipzig : Giesecke & Devrient. - Je Bl. 46 x 44 cm



Datensatz 70302464

Section Kötzschenbroda aus: Topographische Karte (Äquidistantenkarte) Sachsen

Datierung: 1894

Personen und Institutionen:

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek
Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS 15309

Blatt 49

Toponym analysis:

- ☐ no color difference between text and other map objects (both are black)
- ☐ toponyms are intersected by lines of similar width
- ☐ faded letters
- ☐ small signatures (trees) behind text
- ☐ abbreviations
- ☐ different placements compared to older map
- ☐ hard to tell, which group of buildings is associated to which toponym
- ☐ Map uses coordinates calculated from the Ferro Meridian (not the modern Greenwich Meridian)

Map 9: Blatt 1-69 aus: [Topographische Karte (Meßtischblätter) Sachsen] : 4-cm-Karte / Abteilung für Landesaufnahme des Königl. Sächs. Generalstabes. - 1:25 000 Leipzig : Giesecke & Devrient. - 155 Kartenblätter : Dreifarbendruck: Grundriß und Schrift schwarz, Gewässer blau, Geländedarstellung in braunen Höhenlinien. ; Je Bl. 48 x 45 cm. Hrsg. wechselnd: Abteilung für Landesaufnahme des sächs. Generalstabes ; Reichsamt für Landesaufnahme, Landesaufnahme Sachsen. - ...



Datensatz 70302465

Section Kötzschenbroda aus: Topographische Karte (Meßtischblätter) Sachsen

Datierung: 1911

Personen und Institutionen:

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS 15316

Blatt 49

Toponym analysis:

- ☐ no color difference between text and other map objects (both are black)
- ☐ serif font
- ☐ small signatures (trees) behind text
- ☐ large spacing within some toponyms
- ☐ toponyms are intersected by lines of similar width
- ☐ toponyms usually placed in more open areas of the map

*Map 10: Blatt 1-69 aus: [Topographische Karte (Meßtischblätter) Sachsen]
: 4-cm-Karte / Abteilung für Landesaufnahme des Königl. Sächs.
Generalstabes. - 1:25 000 Leipzig : Giesecke & Devrient. - 155
Kartenblätter : Dreifarbendruck: Grundriß und Schrift schwarz, Gewässer
blau, Geländedarstellung in braunen Höhenlinien. ; Je Bl. 48 x 45 cm. Hrsg.
wechselnd: Abteilung für Landesaufnahme des sächs. Generalstabes ;
Reichsamt für Landesaufnahme, Landesaufnahme Sachsen. - ...*



Datensatz 70302465

Section Kötzschenbroda aus: Topographische Karte (Meßtischblätter) Sachsen

Datierung: 1922

Personen und Institutionen:

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek
Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS 15325

Blatt 49

Toponym analysis:

- ❑ a lot of building signatures in the same color and size than text
- ❑ small signatures (trees) behind text
- ❑ multiple fonts (including serif fonts and italic fonts)
- ❑ toponyms are intersected by lines of similar width

Map 11: Blatt 1-69 aus: [Topographische Karte (Meßtischblätter) Sachsen] : 4-cm-Karte / Abteilung für Landesaufnahme des Königl. Sächs. Generalstabes. - 1:25 000 Leipzig : Giesecke & Devrient. - 155 Kartenblätter : Dreifarbendruck: Grundriß und Schrift schwarz, Gewässer blau, Geländedarstellung in braunen Höhenlinien. ; Je Bl. 48 x 45 cm. Hrsg. wechselnd: Abteilung für Landesaufnahme des sächs. Generalstabes ; Reichsamt für Landesaufnahme, Landesaufnahme Sachsen. - ...



Datensatz 70302465

Section Radebeul aus: Topographische Karte (Meßtischblätter) Sachs

Datierung: 1937

Personen und Institutionen:

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek
Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS 16877

Blatt 49

Toponym analysis:

- ☐ (in addition to the formerly found:)
- ☐ letters follow lines
- ☐ letters are just as large as other signatures (buildings, tree-lines)

*Map 12: Wander- und Wintersportkarte Altenberger Gebiet, 1:30 000,
Lithographie, 1929*



Datensatz 70402731

Beschreibung: Kipsdorf - Frauenstein - Moldau - Geising, Altenberg. - Dresden ; Leipzig u. Berlin : Giesecke & Devrient [in Komm.], 1926. - 1 Kt. : mehrfarb. (Wander- und Wintersportkarte des Erzgebirges / im Auftrage des Sächs. Finanzministeriums herausgegeben vom Reichsamt für Landesaufnahme, Landesaufnahme Sachsen ; 4)

Datierung: 1926

Personen und Institutionen:

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS 9308

Toponym analysis:

- ☐ color printing
- ☐ only a few line objects are of similar size and color than the text
- ☐ very disturbing/detailed area signatures and contour lines

Map 13: Plan von Bad Wildbad, 1:20 000, Mehrfarbendruck, 1926



Datensatz 90005033

Beschreibung: Wildbad mit Umgebung / Bearb. vom Württ. Statist. Landesamt. - 1:20 000.
- [Stuttgart], 1926. - 1 Plan : Mehrfarbendr. ; 49 x 41 cm. Kopft. - Mit Höhenlinien

Datierung: 1926

Personen und Institutionen:

Herstellung: Württemberg-Baden / Statistisches Landesamt, Bearbeitung

Verwalter: Dresden, Sächsische Landesbibliothek - Staats- und Universitätsbibliothek
Dresden (SLUB), Kartensammlung, Signatur/Inventar-Nr.: SLUB/KS 21814

Toponym analysis:

- ☐ good contrast between background and text
- ☐ toponyms are intersected by lines of similar width and color
- ☐ curved toponyms
- ☐ multicolored fonts
- ☐ legend

2.3 Summary table

Explanation of Table 1 (see next two pages):

Number: According to Maps in previous chapter.

Map Image: A small size depiction of the map (original size can be found in the appended digital medium) and an excerpt showing text from the map.

Year: The year of production

Toponyms: List of toponym class mentioned in the map

Font: Rough categorization of the found fonts within the map, ordered by the toponym class. (Abbreviations: h = handwritten style, A = Antigua style, g = grotesque style)

Serif: If the map contains serif fonts

Text line width compared to line objects: Comparison between the thickness of the most common line feature and the average width of the text. Ordered by toponym class.

Other: Further notes of relevant observations for text in maps

Close to other text: If a toponym within the map is closer to another toponym than the average text height.








Connected: If any toponym within the map touches a similar colored background object.

Color: The text color

Orientation: Any found orientation of text. (Abbreviations: w = winding, h = horizontal)
Ordered by toponym class.

The ratio of text height to map height: The calculated ratio between the text height and the map height

Table 1: Table of Map Analysis:

Number	1	-	2	3	4	5	6
Map image							
year	1596	?	1680	1762	1798	1851	1879
Toponyms	Nations, cities, rivers	Nations, Cities	Nations, Cities	Cities, villages, hills, forest	Nations, villages, regions, landmarks, rivers	Cities, villages, rivers,	Cities, villages, regions,
font	Handwritten style	Antigua Style	Handwritten style	Hand-drawn Antigua style	Hollow Antigua style,	Antigua Style	Antigua Style
Serif	yes	yes	yes	Yes	Yes	yes	yes
text line width compared to line objects	thicker	similar, similar	thicker, thinner	thicker, thinner, t, t	thicker, thinner, similar, thinner, regular	similar, thinner, thinner	similar, thinner, similar
Other	slightly damages, varying sizes of text	Distorted by scan	Intersecting names, not north as up	capitalized, Interrupted by thick grid lines	capitalized and very spacious		oblique, spacious
Close to other text	Yes	No	Yes (very much so)	No	yes	No, Yes	Yes,
Connected	No	No	yes	yes	yes	yes	yes
color	Black	Black	black	black	Black	black	Black
orientation	all over the place	Partly twisted	All over the place, horizontal	Horizontal	Winding, horizontal, w, h, w	Horizontal, h, winding	Horizontal, h, winding
Ratio of text to map height		0.0143		0.0126	0.0078	0.0080	0.0075

Continuation of table 1:

Number	7	11	-	12	13
Map image					
year	1882	1937	1939	1926	1926
Toponyms	Cities, villages, mountain ranges, regions, rivers, seas, ...	Cities, villages, mountain ranges, regions, rivers, seas, ...	Villages, street names, street nmbrs.	Cities, villages, mountain ranges, regions, rivers, seas, ...	Cities, villages, mountain ranges, regions, rivers, seas, ...
font	Antigua style, A, handwritten style, A, A, A,	Antigua style, A, A, A, A, A,	Antigua style	Grotesque style	Antigua style, A, grotesque style, g, g, g
Serif	Yes	Yes	Yes	No	Yes, yes, no, no, no...
text line width compared to line objects	similar	similar	thicker	similar	thicker, similar, s, s, s
Other	very spacious, capitalized	oblique, very spacious, capitalized	Additional text to toponyms	Capitalized, oblique	Capitalized, oblique, spacious
Close to other text	No	No	No	yes	yes
Connected	yes	yes	yes	yes	yes
color	black	black	black	Black, blue	Black, blue
orientation	Horizontal, h, , winding, twisted, h	Horizontal, h, , winding, twisted, h	horizontal	Horizontal	Horizontal, h, winding, w, w, w
Ratio of text height to map height	0.0083	0.0075	0.0100	0.0054	0.0094

2.4 Fonts and toponyms

2.4.1 History of map fonts

Fonts for historical maps changed over time. The main reason for this, was the change of printing methods. In the early 15th and the 16th century woodcut techniques were the main way of printing.² Fonts were handwritten and different from map to map and often enough different from letter to letter.

With the invention of copper engraving as a printing method, new fonts were invented. Printed fonts like the german Fraktur font replaced the handwritten fonts (compare map 1). Gerhard Mercator decided upon a simpler font compared to the thick Fraktur and recommended an Antigua-like latin script font for maps. (Eckert-Greifendorff, 1939)

This font style dominated maps between the 17th and the late 19th century, till serif-less fonts were seen as better readable and replaced the old ones almost entirely. Almost all maps from this time period, which were chosen in this thesis use a Antigua-like font. (compare maps 5-13)

As these fonts were still hand carved into a copperplate (or sometimes other metal plates like steel), no two letters are perfectly alike and the fonts tend to be very winding and curlique, because gentle curves were easier to carve. Also it was part of the artstyle of this time to attach swashes³ and serifs to the letters. (compare maps 5-11)

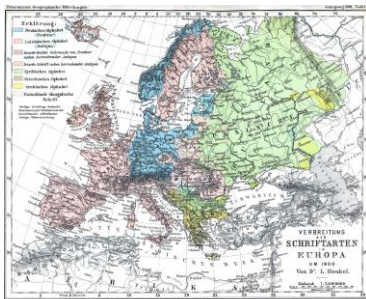


Figure 2: Map of the scripts in Europe (1901) (enlarged version in the appendix)

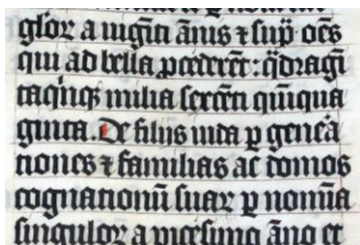


Figure 3: Image of Fraktur Script

One exception to the rule, where some maps from germany after 1941, which reused the old Fraktur font. But they were the exception even within this time period. Using a broad overview, we can determine, that in the german area, the printed documents in the 19th century used the Fraktur font. (Eckert-Greifendorff, 1939) (see Figure 2 and 3) The Kurhannoverische Landesaufnahme (country recording of Kurhannover) named Fraktur and english Current (handwritten style font) as their main toponym fonts. (Bauer, 1993)

Once lithography (among other different printing techniques) became established in the 19th century (Jacob, 2006), the fonts became less curved and clearer in their composition (compare maps 12, 13). Stamps were used to simplify the copying of letters. Antigua-like fonts still dominate maps till the beginning of the 20th century until the Grotesque style fonts took over, as the serif-less fonts deemed themselves as more readable.

² <https://www.lib.umich.edu/online-exhibits/exhibits/show/the-geography-of-colorants/a-brief-history-of-color-in-ma/mapmaking-techniques> (August 2018)

³ <https://typedecon.com/blogs/type-glossary?page=1> (August 2018)

2.4.2 Parameters of fonts

- size

Toponyms on maps can come in various sizes, but there is a limit to what is useful and necessary. Historical maps (based upon the random map selection) have a text size between 2mm as the smallest and up to 20mm for toponyms within the mapped area.

Based upon the random sample of maps, the average ratio of height for a character to the map height is about 0,00908 or roughly 1%. (see table 1)

The average text height is pretty stable (Weinman J. , Toponym Recognition in Historical Maps by Gazetteer Alignment, 2013) and can be used as estimation for an algorithm.

This number is based upon the most dominant toponyms of each map. Averaging has of course a down side, as the information about the size of the text gets lost. The size does however contain a certain information about the importance of the text.

Among city or village toponyms the larger names usually belong to larger cities, which can be more important to the map user, while smaller toponyms are less important.

- color

Black has always been the main color for ink and therefore the most important feature on any print medium - including maps - were printed in black. Although color printing has been established since late 16th century, text remained black for most cases.

Even modern maps make little exclusions of this rule, excluding the widely accepted blue font for water bodies.

- toponym types

The older maps are, the less information they have available for their making. Toponyms consist mainly of the large political and social structures of mankind, as well as the largest most noticeable structures of nature: Country names, cities and villages as well as mountain ranges and rivers dominate the maps of the early half of the second millennia. Increasing amount of documentation made additional information available to the cartographers, adding region titles, street names, hills or landmarks. With the establishment of cadastre systems in the late 16th century maps started to include street- and parcel numbers, as well as altitudes and important buildings.

A problematic type of toponyms are the abbreviations, which are common in the maps for cadastre. They often depict small building, like schools, train stations, huts, etc. Most of them are as short as three letters. The map reader needs a legend to understand those toponyms.

- change of toponyms over time / maps

One noticeable problem came with the lack of tangible data is the change of toponyms over time or author of a map. Misunderstandings or printing errors did change names on maps as much as pronunciation changed them over the ages.

Before the uprising of cadastre, names of cities and places were often delivered by word of mouth. This as well as the simple misspelling of toponyms lead to a change of names from map to map. And names used to change with the time.

Focussing on the spelling and the change from the origin “Bruno’s Wiek” to the modern pronunciation “Braunschweig” (Brunswick in Lower Saxony) is pretty sevier. Some other cities had even larger changes of their name over time: The most famous example would be Istanbul: The city was once known under the name Byzanz and later Konstantinopel.

This change can be found in historic maps and need to be known to notice the historic connection between the toponyms.

Here are two examples of name changes within the example maps:

Coswig -> Koswig (compare maps 7 and Figure 4) and Liebe -> Löbau (compare map 1)

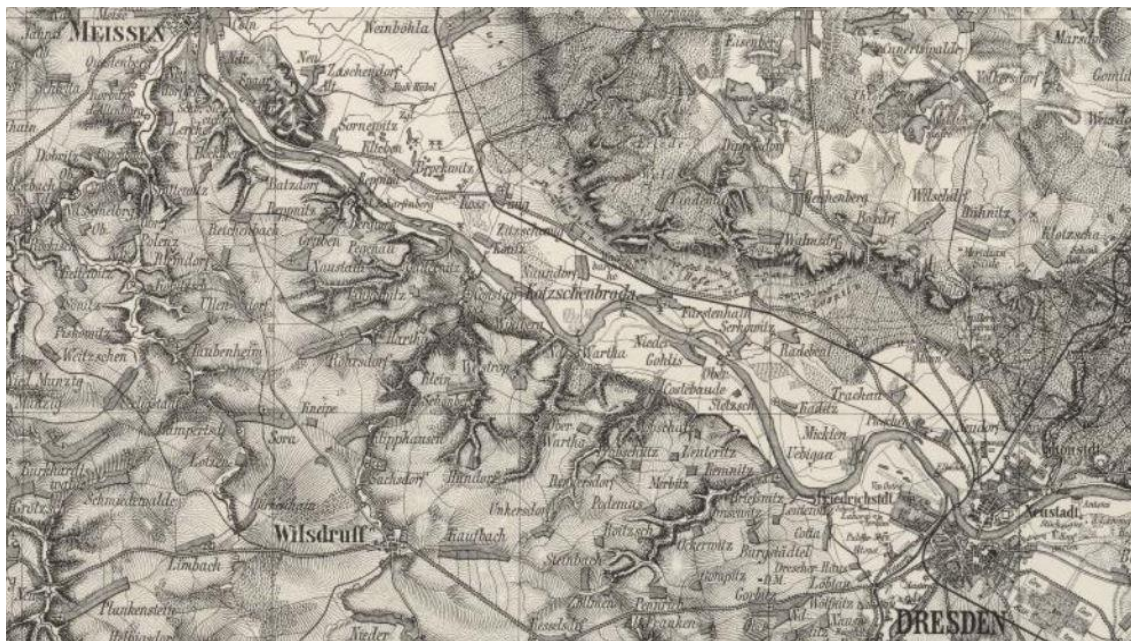


Figure 4: screenshot from: Karte von Sachsen (Königreich), 1:157 281, Lithographie, 1851, http://www.deutschefotothek.de/documents/obj/90010910/df_dk_0011479_0002

2.4.3 Placements

2.4.3.1 Placement of toponyms

Toponyms on maps are a representation of a real object of any kind. That means, their positioning is based on the position of the object in the world. Usually toponyms are connected to a signature or another graphical object. Depending on the kind of object the positioning follows different rules.

Point object toponyms are horizontally oriented and close to the center of the object they depict. This includes city and village names, as well as landmark toponyms. This does rarely follow rules, as the depicted objects can hardly be positioned precisely in the scales between 1:10 000 and 1: 100 000: A town is a collection of buildings of varying shapes, which makes a concrete placement of the toponym by rule impossible. If there is not enough space, the orientation may change as well as the proximity of the toponym to its object.

But point object toponyms are placed very dominantly and readable. In historical maps point object toponyms are the most dominant form of toponyms because cities and villages are the main objects, which position those maps wanted to show. (compare maps 3-13)

Line object toponyms follow the object and fit their curvature. Street- and rivernames are the most common toponyms for line objects. Their font size is usually smaller than the font for point object toponyms, as they are limited by the feature. The font usually is not larger than the width of the line object. The close connection to the line element results in harder to read text (compare maps 4,12, 13).

Area object toponyms tend to follow the space of their element, too, but in the hierarchy of the toponyms they are the last to be set. The letters of area object toponyms fill the gap left by the point and the line object toponyms or is spread over large areas of white space to fill the whole area it is depicting. (compare maps 1, 5, 6)

Especially older maps show a very problematic placement of area object toponyms. Single letters can be set apart and with seemingly random distances to fit between other toponyms or signatures. Their curvature to fit the area happens to be hard to determine algorithmic. (compare maps 1, 2)

2.4.3.2 Placement of other graphical features

- point objects

Point objects are graphical representations of real locations, that only have one pair of coordinates. This includes cities on small scale maps and locations, such as churches or fire stations. These objects are marked on historic maps usually with text in addition to a symbol. I did not find a historic map, where the symbol of a point objects resembles a text or text-like shape.

Text is usually set with a distance towards point objects. They disturb text objects only rarely.

- line objects

Line objects are graphical representations of linear ground coverage or human structures. This includes streets and rivers. These objects are very common on historical maps and are represented as one or two stroked lines.

Text objects do not avoid line objects, because their density is usually too high. There is not enough white space left between the usual level of generalization of streets, contour lines, rivers, etc. for text to be placed without crossing lines.

Reducing or removing the line objects will be a necessity to reduce the number of falsely recognized objects as text.

Certain line objects can look very similar to text, due to curves and straight line parts, which can match the length of character curves or straight line parts by random.

- area objects

Area objects are graphical representations of landcover and similar features, like forests, water bodies or city areas. On modern maps they are usually held in a uniform, light color which is not used for text. Color in historical maps was difficult and usually done by hand. The technique to use multiple prints with different colors was not commonly known. Area objects were therefore represented by patterns or textures, which cover large areas and often include the text.

It depends on the style of the pattern or texture, if it disturbs text. As pattern or texture are usually repetitive lines or signatures it is likely, that it can be falsely recognized as text. Especially maps of the early 18th or 19th century, which used to be black and white, had area textures with very detailed depictions. E.g. forest depicted as small trees. (compare map 3)

- number objects

Height points, distance markers and sometimes coordinate grid values are often scattered around cadaster maps. They are technically no toponyms, but may held similar information to a map reader.

- legend and map frame

The older the map the less likely it is to have a legend. Legends provide a good starting point for the map reader to understand a map. They usually give away the style and fonts of toponyms over other important information like the classification of signatures or the look of the other map features.

Most maps on the other hand have a map frame of some kind, varyingly having a coordinate grid, a scale and a map title. Some have additionally information about the time, author, the client or the depicted region/city.

- Cartouche

The cartouche is a graphical element of the map, that contains text, often including the map title, the author and additional information about the map. Depending on the font it is a easily readable part of the map, as it is a clear textbox. They were common on historical

maps, especially before the time of cadaster, when cartography was mostly an art and less a scientific representation of land.

Cartouches usually use a font that is different from the map toponyms. (compare maps 1, 2)

2.5 Noise and scanning artifacts

- Printing

Prints are never perfectly identical. And text within a print can be even more varying, depending on the used technique: Older prints, which use copper engraving techniques, were carved by hand. Although the artists, who do the carving, were trained in keeping a steady look between the characters, they would leave visible differences between some of them. (compare map 3)

A very small difference in color and its appliance is caused by the printing technique itself: Copper printing uses to have sharper edges for small structures like letters and uses thicker ink. Otherwise color in historical maps is applied by hand, which can rarely cause disturbances in the black printed fonts as well as a difference between otherwise identical objects.

The change to newer technology did ease out the differences, as characters got drawn by hand (e.g. in oil for lithography). Typeset letters changed this in the 19th century, which could be inserted into holes of the printing plate or were printed in a second step. But even letters from the same type piece can vary slightly due to different thickness of the ink or slight angle of the type pieces. (Jacob, 2006)

- age and noise

Historical maps are at least a couple decades old before the original got scanned for the first time. This means that the maps suffered from decay. Archives use methods to preserve their historic documents, but damage cannot be prevented all the time. Fading, scratches or otherwise damaged parts often cause loss of information or disturbances for toponyms.

Once the maps are scanned, digital methods of image editing can be used to minimize the effect of aging. An antifading algorithm (Yao-Yi Chiang, 2014) can reduce the impact of the fading, while gaussian filter can reduce noise within the image, caused by dirt or rough paper.

The scanning process itself can decrease the quality of a historic map as well. Raster images can contain mixed color pixels due to the scanning or compression processes (Pezeshk & Tutwiler, 2010) although the paper material and ink seem to be very uniform. Scanning causes additional disturbances like smoothing, blurring of the edges, and intersecting artifacts, caused by an imperfect alignment of pixels over the color spots on the map. (Pezeshk & Tutwiler, 2010) "Even high-quality paper maps with uniformly

colored features will yield scanned images with considerable variation in color” noticed (Yao-Yi Chiang, 2014).

2.6 Useful findings for text extraction

The analysis of the example maps lead to some findings that should be used for a text-extraction algorithm: The amount of distracting map elements in historical maps is larger than in modern maps. This is due to the monochrome printing techniques, which lessens the amount of parameters, which distinguish text from graphical elements.

Line-elements usually are depicted in simple lines, just as text is and even area-objects can look similar to text. It is definitely necessary to find algorithms that can reduce the amount of lines and area-textures, before a text extraction attempt can be made.

The font styles are unlike most modern maps and are often unique to the given map or map series. It is hard to preselect certain fonts for historical maps in general. It is recommended to have specific parameters for text extraction for each map style separately.

Toponyms share, however, a certain amount of parameters: The color for fonts is black (with the exception of some water toponyms), the font size usually does not exceed 1% of the map height and fonts are in majority oriented horizontally.

The parameter size has to be treated different from the others, as it contains valuable information about the importance of a toponym.

Difficulties can be caused by a dense placement of toponyms, as well as many similar sized signatures or area textures.

Different spelling or even historical names for places need to be noted. This and abbreviations need to be included in gazetteers. The first is included in some gazetteers for European and American countries already, the latter is less common.

Noise, damage and scanning artifacts have to be removed before a text extraction can take place in a reliable fashion.

3. Text extraction algorithms

Prelude: The difference between text extraction and text recognition is a slim one, but essential. This work is structured around the difference. Text extraction is the process of classifying a part of an image as text. At this point is not necessary to understand what the text reads, but only to notice, that it is text.

Text recognition on the other hand is the process of reading and understanding the text and to translate the image into machine readable text or a "String".

Both processes can be used to improve each other, but remain separate processes. This chapter will look into text extraction algorithms, while the following will name methods of text recognition. (Chiang & Knoblock, Recognizing text in raster maps)

3.1 Introduction to text extraction

3.1.1 What is text?

Text is what you read right now. A row of symbols, each with its own meaning, but with a different meaning if grouped together in a certain order.

This order is partly determined by rules, which include phonetic presets (as we are not able to pronounce this sample set of letters: "hvctkpq") and are limited to the sounds the human speech apparatus is actually capable of creating. A set would be for example the 26 letters we know for alphabet of the english language plus the rules of pronunciation for letter combinations: i.e.: "th".

For the written form of text, three additional parameters get added: Capitalization, fonts and punctuation. They create even more variance in the set of parameters one has to look out for, if one is looking for text.

And in the image form of text there are even more parameters, as we have style, font size, color, width, spacing, ligatures and much more.

Luckily we can simplify a bit, as it is not necessary to find positive matches for each parameter, but only to find enough matches to positively say, that this is a textlike area.

Commonly algorithms use color, edges, strokes, and texture as their main parameters (Qixiang Ye, 2015) and shape contexts (Coates, et al.) as additional decider.

Color - as discussed in the former chapters - is the easiest parameter: Text or text labels are mostly part of the black layer. For multicolored maps this means, starting with a color separator gives good results already. Separate by color and look for serial small color patches to detect text. (Chiang & Knoblock, An Approach for Recognizing Text Labels in Raster Maps, 2010) or (Leyk & Boesch, 2010)

Structured edges is another finding, as text contains out of lines, arranged in a unique way. The edges create a noticeable pattern of frequent ups and downs in edge structure.

A group of strokes; short curved lines, accompanied by straight lines is a parameter a algorithm could look for.

On a larger scale it is "a kind of texture" - sum of all the above.

However, there are many objects in map images, such as area textures, line features or signatures, that have similar edges, strokes or texture properties with text, making it difficult to separate false positives from text.

The truth still lies within all of the named parameters. It is likely that - if we look for all the parameters - text will separate itself from similar features. "Text is a character composite" says (Qixiang Ye, 2015) and it seems to be the best answer. Characters are patterns within multiple parameter fields and we have to analyze them all to be sure if it happens to be characters. Integrated approaches that share character classification results with both the detection and recognition problems have been investigated in other works already.

Text can always hold ambiguity however: Especially among less restricted text media, like handwritten fonts, there are examples of ambiguous text, like in Figure 5.

It could read "learn" just as much as it does "teach".

In this case context would be needed as additional parameter to determine the true result.



Figure 5: Image showing ambiguous text. ("teach" or "learn")

3.1.2 How to find text?

From the list of parameters for text we can conclude that no single-step method can be used to find and read text.

Document Analysis and Recognition (DAR) is the overarching method description and is best shown in Figure 6.

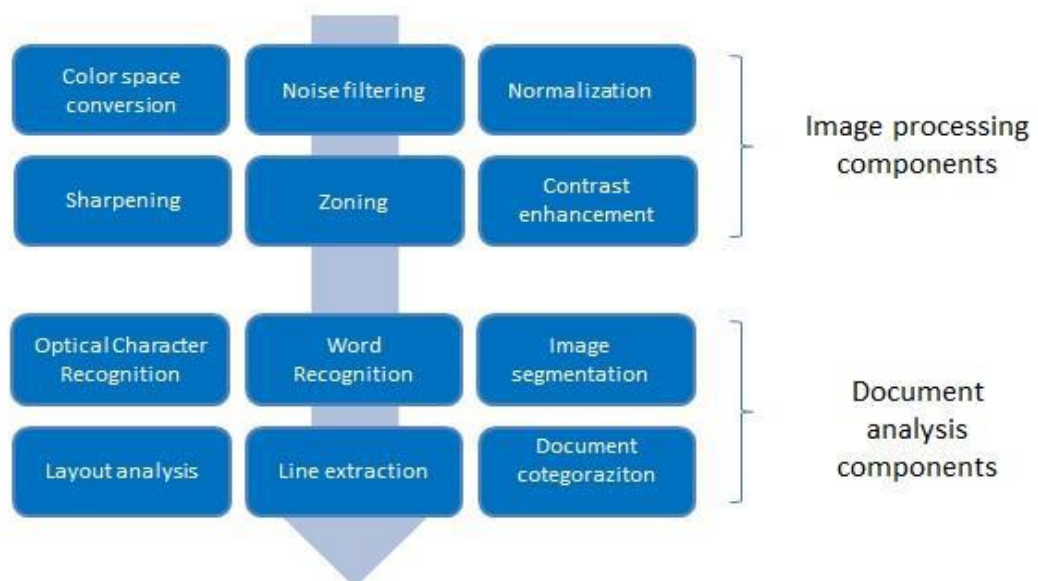


Figure 6: Diagram for the Document Analysis and Recognition system

The main objectives of DAR systems are to recognize textual and graphical components in input documents and to extract information about different characteristics or parameters of input documents to get the semantics or to extract contents (Ghorbel, 2016) by using multiple stages.

An algorithm has to be trained on handcoded features to multi-stage pipelines combining many different algorithms (Coates, et al.) to satisfy each step of a complete DAR System. Text extraction and recognition is part of a DAR system.

Theoretically, if we manage to analyze for all possible parameters, the result would be perfect. Multi-staging is probably how the human brain deciphers text internally. Strokes, characters, words, sentences, and language context get analyzed seamlessly. (Qixiang Ye, 2015) But as we do not fully grasp the functionality of our own ability to read, it is even harder to teach an algorithm the same technique.

Even if we know all the parameters, there are still limitations to the analyzing technique and the constraints of the machine (Ghorbel, 2016). Processing power and coding structure are still limitations to fully copy the biological example.

This leaves us with two ways of approaching the problem: Manually figuring out the parameters and teaching the algorithm our results or creating an algorithm that might be able to learn the parameters by itself:

Until either of these run perfectly, halve automated methods have been developed, which may bridge the gap for now. One is the manual finger tracing technique, proposed by (Y. Watanabe, 2017)

Tracing the characters by finger during reading complex documents is natural and easy for human readers and can aid the algorithm.

3.2 DAR System Part 1: Pre-processing and image filtering

Before the Preprocessing and image filtering (Korb, 2008) has some good advices for scanning documents:

The recommended scanning resolution is 300 dpi. Higher resolutions do not improve the results much, but will increase the processing time.

The scan should not be too bright and obviously include colors. And any kind of skewness of the image should be reduced. Most modern scan systems – especially the ones used in libraries or archives – already have systems to prevent such errors.

Preprocessing the scans for the recognition algorithms helps those to be less distracted by irrelevant data within the image. This includes noise, false coloring, blurry imagery or

background: Pixels that contain no necessary information. In the case of text extraction, the goal of preprocessing would be to erase all pixels in the image, which do not belong to text.

To go by the diagram, major steps of preprocessing an image are:

- color space conversion / separation
- noise filtering (Yao-Yi Chiang, 2014)
- normalization
- sharpening
- zoning (Impedovo, Pirlo, Modugno, & Ferrante, 2010)
- contrast enhancement
-

(Yao-Yi Chiang, 2014) also names Sub-layer separation, solid graphical components removal and dashed lines removal as additional steps.

(Epshtein, Ofek, & Wexler) mention a Stroke width transform, which is a local image operator that computes the width of the most likely stroke containing the pixel. This creates the basis for both line and text extraction.

Another way of preprocessing both the image is the and canny edge Algorithm (Gupta, 2017) to extract only the edges of the image. As text has very defined edges, it would clarify the contrast between text parts and its background.

The work goes on to mention various thresholds, which can be applied to the found region. Edge-area/textarea, text-block-width and text-block-height can be determined this way. Further text parameters that have been used in other algorithms are character distances, straightness and edge density as well as edge count, horizontal profile, connected component height and width. (Qixiang Ye, 2015)

Region grow and Edge detection techniques (Yao-Yi Chiang, 2014) are further zoning tools to filter image data and reducing noise. The Lucy-Richardson Algorithm (Gupta, 2017) has been developed to restore blurry or damaged images and can be used to repair old maps.

Most of these methods are trying to separate the image into parameter fields, which can be analyzed afterwards for their text likeness. Each of the segmented parts should represent an exclusive region, "such that each region is homogeneous, but the union of any two neighboring regions is not." (Yao-Yi Chiang, 2014)

This can be done by any of the named parameters or steps. None of these filter methods is by itself the solution. As text is a multi-parameter feature, preprocessing has to adapt to this too and use a multiple step approach.

Especially as map text has complicated background, partly occluded text, variations in font-styles and image noise (Gupta, 2017) which may obscure any single parameter, but can be filtered by using combinations. After the preprocessing, the altered or feature based representation of the map can be analyzed using compatible extraction algorithms mentioned in the next chapters.

3.3 DAR System Part 2 - Line extraction

Separating linear features from text [Luo and Kasturi 1998] is a major step in deciphering maps. Especially as many lines share a lot of parameters with text, as we discussed in the first chapters.

Detailed methods to find line segments within map images have been discussed in (Ni, 2017).

To prevent text being mistaken as lines zoning tools can be used. Lines can be filtered by size as text usually contains of small lines, larger ones can be assumed as line features. If text touches lines the Hough transformation is a way to determine the direction of each line, excluding long lines going one direction and keeping the shorter of the crossing lines. This way the connected components can be decomposed into line segments to recover the character strokes that touch the line. (Cao & Tan)

(Acheson, Sabbatb, & Purves, 2017) uses a similar technique by breaking the found lines into small segments at the intersection points and then merge those segments that connected to the same junction by line continuation based on similar slopes of the adjoining segments within a threshold difference. The slope of each segment is estimated by line fitting. After decomposition, they use size filters to separate constituent strokes of text from graphical lines.

Once lines and text are separated the text extraction can commence.

3.4 DAR System Part 2 - Text extraction

3.4.1 Filtering by preselected parameters

When it comes to filtering images for text using preselected parameters, there are two main categories, image based matching techniques and feature based matching techniques. (Ghorbel, 2016) Image based matching techniques use raster or matrix representations as templates or filters to analyze a picture area for area and find similarities.

Feature based matching techniques use formula or parametric representations of an image and search for text-like patterns within these.

A detailed survey of various matching algorithms and their applications can be found in (Yao-Yi Chiang, 2014).

The following methods are the ones, I selected as doable for this thesis or as giving right ideas. This selection is neither representative nor complete of all the techniques other scientists have thought about, especially the ones more advanced, as they had science teams working on them for a longer time than a master thesis. There will always be a citation of the actual work, while this chapter will only broadly describe the idea of it.

Image based matching techniques are the simplest method of detecting text. By sliding an image of each letter over the original image and comparing, each letter can be found. This however is a limited approach as it needs to derivate the size of the letter as well as a good idea on how the font looks like.

If the orientation is unknown too, the iterations for each template multiply. Modern computer systems are capable of processing a high number of comparisons in rapid succession, but the increase of time over each new parameter becomes quickly measurable.

User input, like (Y. Watanabe, 2017) or (Chiang & Knoblock, Recognizing text in raster maps) proposing, does cut down the amount of time, by preselecting parameters like rotation, size or color of the text.

Hierarchical template matching approaches for text extraction, mentioned in (Yao-Yi Chiang, 2014) do improve the performance of this approach. And legend-driven recognition system increase the recognition rates, too, but require a legend or user selected text.

All of these pattern-recognition approaches are limited by the time-consuming training or user-guided inputs and suffer from a high degree of map specificity. As each automatically learned or preselected amount of parameters will only fit the given map, this process has to be repeated to a certain degree for each new map.

(Yao-Yi Chiang, 2014) describes a symbol recognition system, that is based on Hausdorff-distance between symbol templates and the image. RANSAC can also be used to calculate similarities, based on its random sample consensus method. (Anandhi & Avudaiammal, 2017) names further features of image matrices that can be used to compare a template with an image part, like the autocorrelation, contrast, mean or variance value.

These are among many similarity functions for images, that already blur the line between image based and feature based matching techniques.

Some of the following features can be used to determine similarities between a template and an image: Cross line features, to find identical patterns, extracted from a global Wavelet representation of the image or the skeleton analysis method also mentioned in (Qixiang Ye, 2015).

Image component sizes can be used, followed by density parameters, like the percentage of black (object) pixels within the given region (Pouderoux, Gonzato, Pereira, & Guitton, 2007). These features can also be used directly as feature parameter to distinguish text from other objects, as the percentage and the size could be predetermined and cut from the original map image.

Other global image features are listed in the Word Shape Code (WSC). WSC is a list of mathematical features, which describe text (Spitz & Maghbouleh, 1999). Among those features are: Character ascenders/descenders, deep eastward and westward concavities, holes, i-dot connectors and horizontal-line intersection. (Ghorbel, 2016)

Instead of using separate parameters, there are already developed feature based extraction algorithms that use their own set of parameters to determine features: The best known is

SIFT (or the improved SURF). It calculates key points within an image, which can be compared between template and original map image.

A small scale test by the author of this thesis however did show, that SIFT technology struggles with small templates, which only contain a single character, as the algorithm does not find enough keypoints. (Iwamura, Kobayash, & Kise, 2011) is a good example.

And finally: (Gupta, 2017) starts with a Maximally stable extremal regions (MSER) detection technique, which is a complex zoning method and suitably explained in that work.

Each of these solutions did work for their specified challenge, but were not used on historic maps specifically.

3.4.2 Neural networks

All of the named methods for text extraction to this point have the downside, that the parameters have been preselected by a human developer, that set up the algorithm to look out for what he thinks is text or text like.

There have been efforts since the late mid of the last century to create a computer algorithm that is capable of learning just like a human being. Within the last couple of years the idea of neural networks – self learning networks have been created in numerous fields.

There are still limitations to the speed and the complexity of such networks, mainly due to lacking hardware. Most of them specify in a single task, like learning a game. Deep Blue is a neural network that learned to play chess just as well as the best human players.

And there are neural networks, which specify in understanding imagery. Keras is among them. It is an unsupervised feature learning algorithm that can automatically extract features from the given data and order them via a convolutional neural network (CNN) (T. Wang, 2012)

(Qixiang Ye, 2015) and (Coates, et al.) name complex variants of neural networks, that have been used for image deciphering. This thesis made an attempt to use Keras as a python library, but did not succeed to gain any usable results from it, as the task to adapt a neural network towards filtering text from historic maps, with all their mentioned problems, was too challenging, without a clear understanding for neural networks.

(Grond, 2017) however gives a clearer overview for this topic, although it approaches text in natural images rather than maps.

4. Text recognition algorithms and OCR

4.1 OCR

Text recognition – actually identifying the text as words – is usually handled by an Optical Character Recognition (OCR) software.

Most commercial OCR packages have an accuracy of between 97% and 99%. (Gupta, 2017) These percentages are based on character errors, not word errors. While 97% of all letters may be accurately found by the software, the word detection rate can be worse. (Qixiang Ye, 2015)

But Open Source OCRs, like the Tesseract OCR used in this thesis do not lack that much in performance: (Blanke, Bryant, & Hedges, 2012) (Smith, 2007)

Depending on the OCR, preprocessing steps might actually harm the performance, but this is depend on the algorithm. But certain processes can alter the image in a way that confuses the algorithm. Especially noise filtering or contrast changes. (Packer, Lutes, Stewart, Embley, Ringger, & Seppi, 2010)

This is because many OCR algorithms have their own kind of preprocessing or text extraction algorithms. (Om Prakash Sharma, 2013) For most open-source OCRs that does not mean, they will automatically fit any needs. Usually the text extraction algorithms in OCR are specialized for white paper documents. They assume uniform text lines without changing fonts.

Therefore preprocessing or separate text extraction might be necessary, if the image data is diverting too much from simple text imagery. That is the case for historic maps.

If OCR is viable depends on certain variables:

- o texts published prior to 1850 will cause trouble for most OCRs. For some languages (e.g., German) the cutoff date may be even later.
- o aged and discolored documents must be scanned in RGB mode to capture all the image data
- o Low-contrast documents should be avoided
- o inconsistent use of font faces and sizes will lower OCR accuracy.
- o Handwritten documents cannot be recognized with any degree of accuracy.

Especially the last two points do hinder typical OCR software to work with historic maps. Using the Tesseract OCR python library – that will be described in greater detail in a later chapter – on two sample historic map, does not lead to a usable outcome:

For comparison the third example is a simple document with randomly positioned toponyms. The OCR system recognizes them, without a single false character.



```
New File:
df_dk_0000054_027.tif:
Tesseract found:
Blatt 27.
```

```
Stadr,,tail. Ob er :: orb1tz ,,
,(__
```

All'rän en't"

Fester f(i:z

Dresden 1:5000.

„„\$E::5
• „ •

<->

```
Gazetteer found:
closest match for: Blatt27.
No match
closest match for:
Stadr.,tail.Ober::orb1tz,,(____
No match
closest match for:
Alll'ränen't"
No match
closest match for: Festerf(i:z
Pesterwitz 0.6666666666666666
Leuteritz 0.6
Jesseritz 0.6
closest match for:
Dresden1:5000.
Dresden 0.6666666666666666
Dresden 0.6666666666666666
closest match for: ,,,,$E::5
No match
closest match for: ,,,.
No match
```



```
New File:  
df_dk_0000243.tif:  
Tesseract found:  
,, V . . . , , , , , , , , , ,  
,,,, , _ , . _ ,, }
```

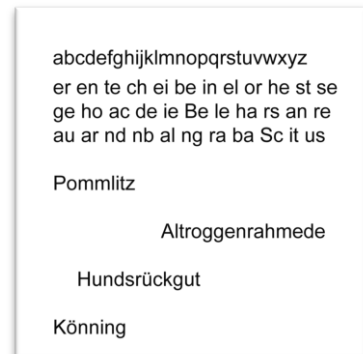
Spetion Kötzschenbroda.

Segen

. m ,, mm «.. «\ vw \mm
[lung «my d,,v,,n ,,mp . MA
"" ,\4144 WM } %.
/i'jf7

< - >

Gazetteer found:
 closest match for:
 „V...,,,,,,,,, _ , _ „} }
 No match
 closest match for:
 SpetionKötzschenbroda.
 Kötzschenbroda
 0.7777777777777777
 closest match for: Segen
 Sieggraben 0.6666666666666666
 Pegenau 0.6666666666666666
 Belgern 0.6666666666666666
 closest match for:
 .m,mm«. .«\vw\mm[lung«myd,,v
 „,n,mp.MA““ ,\4144WM}%.
 No match
 closest match for: /i'jf7
 No match



```
New File:
Schrift.tif:
Tesseract found:
abcdefghijklmnopqrstuvwxyz
yz
er en te ch ei be in el
or he st se

ge ho ac de ie Be le ha
rs an re
au ar nd nb al ng ra ba
Sc it us
```

Pommlitz
Altroggenrahmede
Hundsrückgut
Könning

(Jaderberg, Simonyan, Vedaldi, & Zisserman, 2015) did a study on accuracy of different OCR systems. It resulted in the table below, mentioning various systems with varying accuracy.

Model	IC03-50*	IC03-Full*	IC03-50k*	IC03	SVT-50*	SVT	IC13	IIIT5k-50*	IIIT5k-1k*
Baseline ABBYY (Wang et al. (2011))	56.0	55.0	-	-	35.0	-	-	24.3	-
Wang et al. (2011)	76.0	62.0	-	-	57.0	-	-	-	-
Mishra et al. (2012)	81.8	67.8	-	-	73.2	-	-	-	-
Novikova et al. (2012)	82.8	-	-	-	72.9	-	-	64.1	57.5
Wang et al. (2012)	90.0	84.0	-	-	70.0	-	-	-	-
Goel et al. (2013)	89.7	-	-	-	77.3	-	-	-	-
Bissacco et al. (2013)	-	-	-	-	90.4	78.0	87.6	-	-
Alsharif & Pineau (2014)	93.1	88.6	85.1	-	74.3	-	-	-	-
Almazán et al. (2014)	-	-	-	-	89.2	-	-	91.2	82.1
Yao et al. (2014)	88.5	80.3	-	-	75.9	-	-	80.2	69.3
Jaderberg et al. (2014c)	96.2	91.5	-	-	86.1	-	-	-	-
Gordo (2014)	-	-	-	-	90.7	-	-	93.3	86.6
DICT Jaderberg et al. (2014a b)	98.7	98.6	93.3	93.1	95.4	80.7	90.8	97.1	92.7
CHAR	98.5	96.7	92.3	85.9	93.5	68.0	79.5	95.0	89.3
JOINT	97.8	97.0	93.4	89.6	93.2	71.7	81.8	95.5	89.6

Figure 7: Result table from (Jaderberg, Simonyan, Vedaldi, & Zisserman, 2015) for different OCR methods (rows) over different sets of data (columns)

This study shows good results. It however focuses on classical OCR problems, where the characters are monotone and on fixed backgrounds. Historic maps do provide a more complicated challenge, with challenges in texture, line recognition as well as not standard fonts and sizes.

4.2 Post processing

Improvement to standard OCR systems can be found in integrated methodologies like word and language models (Issam Bazzi, 1999). The findings of the OCR systems are cross checked with typical word construction rules (i.e. for the english language, there will always be a vocal within a word). If the document contains whole sentences a check with a language model can help to fill gabs, where characters might be falsely interpreted by the OCR.

This is already a kind of post processing, which will be shortly explained in the next chapter.

Other steps, that can improve the OCR result in general, are spell checkers and dictionary checks. A dictionary for words the OCR has to look out for, can be preselected by the user (typically a dictionary for the language the document is written in), but it can also be derived from a training dataset of images, containing all the words necessary to be found. (Weinman, Learned-Miller, & Hanson, 2009)

On the other hand, one can use the results of OCR software and create templates for text extraction from them. By knowing that a word is correctly recognized, the text area in the

original map image can be cut and used as template to search for further – similar – text areas within the map, using the processes explained earlier in this thesis.

4.3 Gazetteers

Gazetteers are tables of toponyms - usually for a secluded geographical or political region. They can be used as look-up dictionaries for found terms, especially for spell checking OCR results.

Depending on the region gazetteers can be found in various quality and quantity. This thesis focussed on germany. Gazetteer coverage for germany and surrounding states is good. (Compare gazetteers for each country on <http://download.geonames.org/export/dump/> from Geonames Gazetteer)

GeoNames is the best-known placename data source as of today, often used in academic works. At the moment it has the highest coverage with 10 million entries worldwide of all known gazetteers. It is freely available online and is updated almost daily. As most gazetteers it contains mostly city and village names. This is the reason why this thesis focuses on city and village toponyms on historical maps. Gazetteers make for a good ground truth dataset as well as a look-up table for geographical coordinates.



Figure 8: Cut-out text from the map Plan von Dresden. Blatt 27 [Wurgwitz, Altfranken, Pesterwitz], 1939

The use of a large dictionary may help additionally, as historic maps sometimes contain words that are not toponyms like “Flurteil” (meaning land lot in german). Also it may help recognize business names and street names (Qixiang Ye, 2015).

Besides correcting OCR results, gazetteers hold much other valuable information, like historic names and spellings, population information and hierarchical information such as containing administrative areas including countries. The toponyms are often linked to latitude-longitude points, giving a secondary attribute for proving errors. (Acheson, Sabbath, & Purves, 2017) By knowing the position the result can be checked, if it makes sense to have the found features within the same map. On the other hand, the covered geographic space of the map can be determined too, by knowing which features are shown in the map and what geographic position they have.

When it comes to comparing the found OCR result with the gazetteer, misread characters, missing characters and misplaced wordparts are common and have to be dealt with. A simple look-up of the OCR result within the gazetteer will not be sufficient.

A well-known method for string comparison is the Levenshtein Distance (Ghorbel, 2016) that is a numeric value for the similarity of two strings. By using this numeric value, the similarity of all toponyms within a gazetteer can be calculated and the most likely string can be chosen.

This does however still lead to errors, as some toponyms are too similar to distinguish correctly. The geographic position can be used as secondary decider.

5. Application development

The second part of this thesis had been to develop an application, that can extract and read toponyms from a historical map. This application is supposed to become part of a bigger toolbox, with multiple functionalities, including various picture enhancement tools, image analysis tools or line extraction tools to decipher historical maps.

As of this thesis only a couple of these tools exist. These tools are standalone applications, but it is planned to bundle them in one toolbox. A visual graphic interface shall be used to chain different tools and manage input and outputs of the tools. This increases the intuitive usability for the novice user.

As this toolbox application is not fully functional at the time of this thesis, this application for text extraction will be designed as a standalone. The functionality will later be copied into the full toolbox. The design will be influenced by this premise.

Tools that already exist will be assumed as useable pre-processing methods for this text extraction tool. Their functionality will therefore not be implemented within this standalone project. This includes line and area extraction tools (Hough transformation, cluster analysis tool) as well as noise reduction algorithms or other image processing methods that could be used to perform pre-processing on the image to simplify the text extraction.

In the finished toolbox, those methods can be used prior to this algorithm.

The necessary preprocessing steps for this text extraction and recognition algorithm were mostly done by hand, if needed for testing. Otherwise the following steps are assumed to be automated within the finished application:

- deleting line features
- reducing noise in image
- finding similar colored regions (maybe detecting color ratios in whole image to select certain colors)
- and separating fore and background

5.1 Python

The language of choice for the application is python. The high level language is widely used in gis applications and allows the usage of low level (and open source) libraries, that prevent redundant work.

5.2 Free libraries

The libraries used within the application are the following:

- Tesseract

Tesseract is an open-source OCR library, originally developed for C++, but has been ported to run within a python environment. It has been originally developed by HP between 1984 and 1994, was given to Google later and then released as free software.

Tesseract has been improved to recognize various different languages and is to this day still one of the best open source software solutions for text recognition. (Smith, 2007) explains the functionality rather nicely in his essay.

- Tkinter

Tkinter is the de facto standard GUI for python and has been developed by Fredrik Lundh. It is free and released under the Python license. It is used to create a simple GUI for this application.

- OpenCV

OpenCV (Open Source Computer vision) is a programming library for real time computer vision. It was developed by Intel associates, but was released as an open source project. Just as Tesseract, it is created as a C++ library, with bindings to python.

OpenCV has among a lot of other functionality good image handling functions, for which it is used in this application.

- numpy

NumPy is a library for python to handle matrices and arrays. It is open-source and still under heavy development by numerous developers. As images can be handled as matrices, the application uses numpy functionality.

- Keras

Keras is a multi-purpose open source neural network for python. It can handle image classification, for which it was looked at for this thesis, but not implemented. (see chapter neural networks)

5.3 Used workflow

The main idea behind the algorithm lies in the observation, that text separates itself from other graphic elements by its unique structure. The structure can be detected by using a simplified version of this structure as a template. By comparing this template with every part of a given image of a map, text-like areas can be detected.

Another characteristic of text is, that it is usually horizontally orientated and contains multiple characters, that make up a toponym. By detecting multiple text like features within a horizontal line, we can find the whole toponym.

5.4 Step by step

Once the application is started the user started the GUI provides the questions for two inputs and an output directory to the user. See Figure 9.

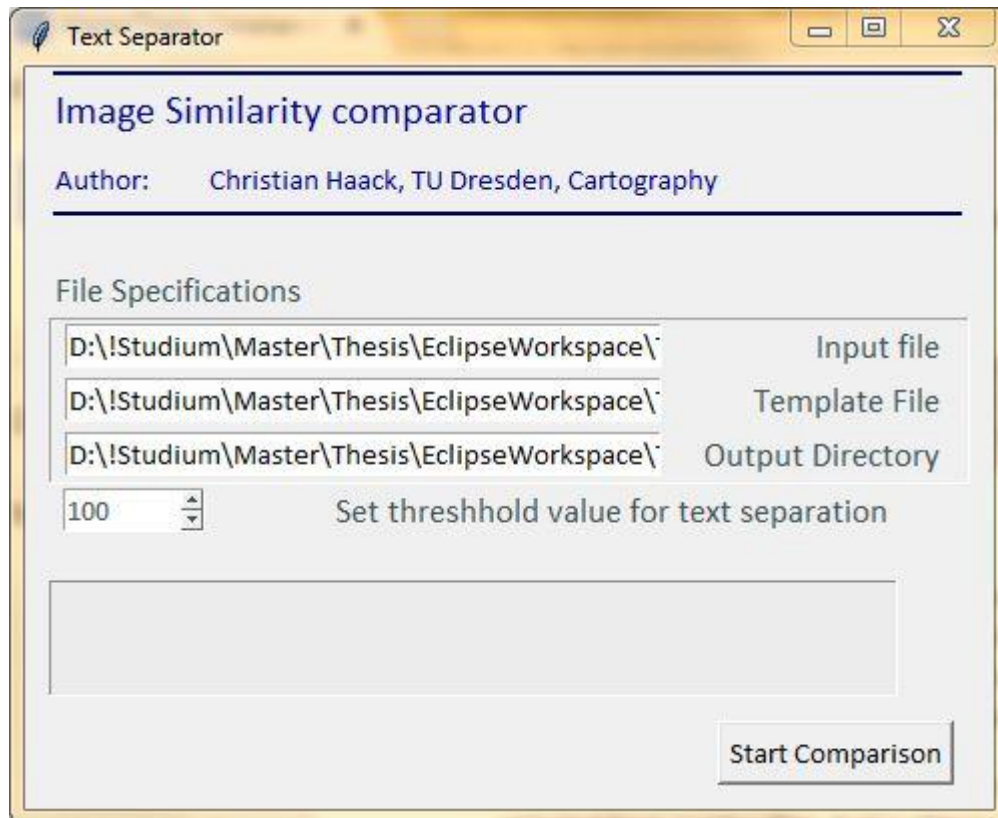


Figure 9: Screenshot of the simple guided user interface of the python Image Similarity comparator

The first entry is for the input map, which should be analyzed with the template.

The second entry is the template, with which the similarity shall be calculated.

5.4.1 Filter templates:

The user defines a template to detect text like areas. This can be an extraction of a character within a sample map or a filter that emphasizes text like features. The best results with a single filter could be archived by using a two stroke filter, which detects horizontal lines.

That has been determined by testing various templates:



Square template (threshold: 60)



a-template (threshold: 100)



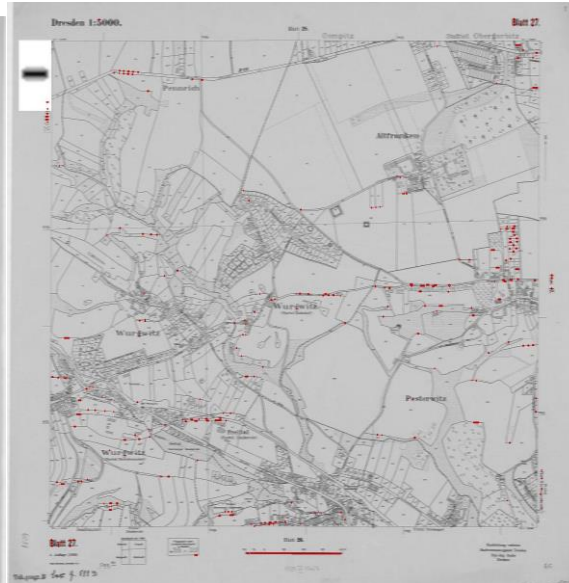
e-template (threshold: 80)



x-template (with gradient) (threshold: 100)



x-template (threshold: 80)



line-template (threshold: 20)



slash-template (threshold: 40)



big-line-template (threshold: 100)



line-template (threshold: 40)

two-stroke-template (threshold: 150)

Figure 10: 10 images, of the map Plan von Dresden. Blatt 27 [Wurgwitz, Altfranken, Pesterwitz], 1939, filtered with the template visible in the top left corner of each image. Red spots show hits over the given threshold.

Figure 10 highlighting in red, which areas of the map match the template in the upper left corner (which is scaled up) above a certain threshold. The similarity is calculated using the formula below. The threshold is chosen by the user for the best result and is noted below the picture. Just visual analysis already shows, that the two-stroke-template matches the text areas the best of the set.

In this version - due to time and skill restrictions - the templates have to be prepared by hand or selected from existing files. It also allows only for a single template. It would clearly be advantageous for the user to have a selection of filters to choose from. And it would improve the results massively if multiple templates could be added up, to narrow down the selection of extracted map areas.

Once the template is set, the user has to name a directory for the resulting files.

And lastly the accuracy for the similarity value is set. This value defaults to 150, but as has been shown by the varying templates (Figure 10) has to be determined experimentally, as it is different for each map and template.

Next step would be to hit “Start Comparison) and the automated process starts:

The main algorithm contains of two consecutive methods: The text extraction, containing a Partmatrix similarity processor and a textgroup finder. And the OCR assisted text recognition part, with gazetteer comparison.

5.4.2 Partmatrix similarity processor

Both the input map and the template are stored as Matrix class objects, containing the matrix, as well as additional information, which will be used throughout the algorithm.

```
# load and validate input
mtx = Matrix(cv2.imread(US.inpth))
Dialogue.insert('1.0', mtx.validate() + "\n")

templmtx = Matrix(cv2.imread(US.tempt))
Dialogue.insert('1.0', templmtx.validate() + "\n")
```

This includes the validity check value, the matrix, the matrix size, the first filtermatrix, the partsummatrix, the scale factor between the partsummatrix and the original input matrix, as well as the resulting bounding boxes.

After the validation of the inputs, the similarity function gets called:

```
mtx.getSimilarity(templmtx)
```

A preprocessing step is included to improve the results:

```
maskmtx = templmtx.thresholding(40, 1, 255)
b = to_grayscale(np.array(maskmtx))
```

It thresholds the image by the value of 40 and converts the image to a grayscale image. This should be done in a mentioned preprocessing step, but is included directly in this code for simplicity.

It cuts the given map into parts, exactly the size of the template. See last line of following code snippet. The calculation beforehand calculates the coordinates of the partmatrix and making sure it is within the original map matrix (`withinbounds()`) and iterates over the whole image.

In case of an uneven amount of pixels as size of the template the `unevenx` and `uneveny`-values will compensate.

This loop starts at the top left corner and steps pixel by pixel to the right, till the end of the first line of the map is reached. The algorithm proceeds with the next line:

```
halbx = int(templsize[0] / 2)
halby = int(templsize[1] / 2)
unevenx = 0
uneveny = 0
if templsize[0] % 2:
    unevenx = 1
if templsize[1] % 2:
    uneveny = 1
end = imgsize[0] - templsize[0] + 1

for i in range(imgsize[0] - templsize[0] + 1):
```

```

for j in range(imgsize[1] - templsize[1] + 1):
    # creating submatrix for each iteration
    lowx, highx, lowy, highy = withinbounds([i - halbx, i +
    halbx + unevenx, j - halby, j + halby + uneveny], 0,
    imgsize[0], 0, imgsize[1])

    partmtx = Matrix[lowx:highx , lowy:highy]

```

It will calculate the similarity of each part of the given map and stores the resulting value as a new matrix. The similarity calculation in pseudo code:

Multipliedmatrices = multiply(255-Mappartmatrix, Templatematrix)

Resultmatrix[i][j] = (Multipliedmatrices.sum() / (Templatematrix * 255).sum()) * 255

The resultmatrix is stored within the Matrix class and return. Figure 11 shows the matrix: The brighter the higher is the likeness.



Figure 11: Filtered image of the map Plan von Dresden. Blatt 27 [Wurgwitz, Altfranken, Pesterwitz], 1939 (original as cut in, in the top left corner) The lighter the filtered image, the higher the similarity to the Two-Stroke-Template

The next step within the main routine is:

```
mtx.partssum(templmtx, 150)
```

The resulting matrix is rescaled using the partsum-method.

The image gets rescaled by the factor of the size of the template, to summarize all hits made by the similarity check that likely belong to the same character. This way a found character is represented by only one pixel in the new result matrix.

Also the method includes a threshold on what amount of likeness is actually considered as a character. This number is defined by the template. The template finding test used said threshold (compare chapter part: Finding a unique filter) and is defaulted on 150, but can be changed in the Interface.

```
if partmtx.max() >= threshold:
    value = partmtx.sum()
else:
    value = 0

resmtx[o][p] = value
```

This is the resulting Matrix:



Figure 12: Resulting matrix of the partsum method. The lighter the pixel, the higher is the similarity to the Two-Stroke-template

5.4.3 Textgroup finder

After the rescaling the algorithm runs through the matrix to look for consecutive hits within the new result matrix. Text usually has more than one character, so we exclude all single pixels marked as a hit:

```
clasmatrix, ggrouplist = mtx.partsummtx.textareagroup(100)
```

`textareagroup` will save areas with more than one consecutive hit, but less than twenty-six⁴ consecutive hits, although it allows for gabs of one pixel size to still be noted as a hit. This is necessary, because the filter template does not hit on each letter. It searches for consecutive pixels over and under the given row.

The parameter 100 is another threshold for safety, whereas pixels below the value of 100 will not be grouped. This excludes unreliable single pixels close to groups created by the former `partsum` method.

```
for line in range(0, matrix.shape[0] - 1):
    found = matrix.shape[0]
    for column in range(0, matrix.shape[1] - 1):
        for increment in (-1, 0, 1):
```

Each hit gets saved by position in a dictionary:

```
if (i, j) in pixeldict.keys():
    pixeldict[(i, j)] = pixeldict[(i, j)] + [group]
else:
    pixeldict[(i, j)] = [group]
```

As this method leads to multiple group assignments per pixel, duplicates have to be erased, which happens within the next steps. The center of each group gets calculated:

```
x1 = key[0]
y1 = key[1]
x2 = groupcenter[0]
y2 = groupcenter[1]
dist = float(sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2))
```

and the pixel gets assigned to the closest center:

```
if dist == 0:
    continue
elif dist < distmin:
    distmin = dist
    group = each
```

⁴ The longest consecutive german village name is Schmedeswurtherwesterdeich (26 letters) according to: <https://www.bz-berlin.de/ratgeber/welcher-ist-der-laengste-ortsname-in-deutschland> (September 2018)

The result is a dictionary of groups with the positions of each pixel within the group; the BoundingBox Object. And a matrix will be saved, where each pixel has its group number as value. This is the Clasmatrix Object.



Figure 13: Result of the textarea-grouping method, colored afterwards. Each color represents one found text group

Using the BoundingBox Object the next step in the main routine will be called:

```
claxsmatrix, boundingboxes = clasmatrix.textareabb(ggrouplist)
```

textareabb() is the function to calculate Boundingboxes (bb) for the textareas, by iterating through the grouplist (BoundingBox Object) and calculating top-left and bottom-right coordinates for each group.

```
bbxs[grpnbr] = [xmin, xmax, ymin, ymax]
if line < bbxs[grpnbr][0]:
    bbxs[grpnbr][0] = line
if line > bbxs[grpnbr][1]:
    bbxs[grpnbr][1] = line
if column < bbxs[grpnbr][2]:
    bbxs[grpnbr][2] = column
if column > bbxs[grpnbr][3]:
    bbxs[grpnbr][3] = column
```

If two groups are within a given distance (neighborhood) they will be merged. The finished list of bounding boxes will be returned.

In the last step the algorithm will save each of the bounding boxes as a separate image file to the given result directory.

The last part of the main routine is to save each matrix and the bounding boxes as images into the output folder:

```
cv2.imwrite(US.outpt + "\\filtermtx.tif", mtx.filtermtx.mtx)
cv2.imwrite(US.outpt + "\\partsum.tif", mtx.partsummtx.mtx)

coloredclassmtx = clasmatrix.classmatrix_to_color(ggrouplist)
cv2.imwrite(US.outpt + "\\classmatrix.tif", coloredclassmtx)

coloredclassmtx = claxsmatrix.classmatrix_to_color(ggrouplist)
cv2.imwrite(US.outpt + "\\classbbmatrix.tif", coloredclassmtx)

bbxc = boundingboxes.addmargin(templmtx.mtxsz[0]/2,
templmtx.mtxsz[1]/2,  mtx.mtxsz[0], mtx.mtxsz[1])

mtx.save_areas(bbxc, US.outpt)
```

5.4.4 OCR and gazetteer comparison

A second algorithm contains the OCR and gazetteer methods. It will use the output directory from the text extraction algorithm as input and run the tesseract OCR routine over each image file within the directory:

```
files = os.listdir(dir_path)
for file in files:
```

The results of this are cross checked with a gazetteer containing all village and city names of Germany, including historic names and writings. If it finds a hit, it will output the result with the highest likeness.

```
a = difflib.get_close_matches(element, searchlist)
```

It uses the difflib library for python to compare each gazetteer element with the found text within the image. The console output shows the most likely hits.

New File: Ausschnitt-v-s-man.png:
 Tesseract found:
 Wur QWi tz

(? lüMil Koblsdorf)
 <->
 Gazetteer found:
 closest match for: WurQWitz
 Wurgwitz 0.75
 Wurgwitz 0.75
 Wuhnitz 0.6666666666666666
 closest match for: (?lümilKoblsdorf)
 Kobelsdorf 0.6666666666666666
 Knobelsdorf 0.6428571428571429
 Züllsdorf 0.5384615384615384

 New File: Ausschnitt-Wurgwitz.png:
 Tesseract found:
 \fVu P @\Vi tz
 <->
 Gazetteer found:
 closest match for: \fVuP@\Vitz
 No match

 New File: Ausschnitt.png:
 Tesseract found:
 %Vur gw1 Ä

„ „. \m ,,,mm;
 <->
 Gazetteer found:
 closest match for: %Vurgw1Ä
 No match
 closest match for: „„. \m ,,,mm;
 No match

 The results of the OCR algorithm can be connected to the results of the former text extraction algorithm to archive a list of found toponyms and their position in the original map, which has not been implemented. This among other steps could not be completed within the given time of the thesis.

5.5 Results

On the simple testmap, each of the few large toponyms have been positively found. On the other hand there are still a lot of false positives among the found results:



Figure 14: original map: Plan von Dresden. Blatt 27 [Wurgwitz, Altfranken, Pesterwitz], 1939 and the extracted text areas below.

On the more difficult map the results are below usable amount, as there are too many false positive matches:



Figure 15: original map: Section Kötzschenbroda aus: Topographische Karte (Äquidistantenkarte) Sachsen

This resulted in a lot of hits: some of them can be neglected because of their small size, but the software does not look out for that. It still shows a lot of false positives, of which I only include some. It had 116 hits, of which 17 actually contained toponyms:



Figure 16: A random selection of the text areas extracted from the map Section Kötzschenbroda aus: Topographische Karte (Äquidistantenkarte) Sachsen

It is clear, that the detailed area object texture did confuse the algorithm a lot.

Neither is of satisfactory accuracy to be used professionally. The OCR struggles to read the found results in most cases without any further processing of the image:

This is the log for the first, simpler map (not the same order as the images):

The algorithm takes in a new file, runs the Tesseract OCR and displays the results. After the <-> symbol, the gazetteer comparison happens and shows the most likely matches and their likeness percentage.

New File: testresult12.tif:
 Tesseract found:
 Dresden 125000.
 <->
 Gazetteer found:
 closest match for: Dresden125000.
 Dresden 0.6666666666666666
 Dresden 0.6666666666666666

 New File: testresult15.tif:
 Tesseract found:
 Blatt 27.
 <->
 Gazetteer found:
 closest match for: Blatt27.
 No match

 New File: testresult21.tif:
 Tesseract found:
 Blu
 <->
 Gazetteer found:
 closest match for: Blu
 Bühlau 0.6666666666666666

 New File: testresult22.tif:
 Tesseract found:
 2R,
 <->
 Gazetteer found:
 closest match for: 2R,
 No match

 New File: testresult231.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult280.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult288.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult296.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult310.tif:
 Tesseract found:
 <>1 ,,,

& Fre) n1

„WU ; „

<->

Gazetteer found:
 closest match for: <>1,,,
 No match
 closest match for: &Fre)n1
 No match
 closest match for: „WU;„
 No match

 New File: testresult326-tt100.tif:
 Tesseract found:
 Nrm-mv
 <->
 Gazetteer found:
 closest match for: Nrm-mv
 No match

 New File: testresult326.tif:
 Tesseract found:
 mewn nm,,wm
 <->
 Gazetteer found:
 closest match for: mewnnm,,wm
 No match

 New File: testresult33.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult34.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult350.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult361.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult369.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult381.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult382.tif:
 Tesseract found:

Could not identify any text in image!

 New File: testresult398.tif:
 Tesseract found:
 BMA 26,
 <->

Gazetteer found:
 closest match for: BMA26,
 No match

 New File: testresult405.tif:
 Tesseract found:
 Blatt 27.
 <->

Gazetteer found:
 closest match for: Blatt27.
 No match

```

-----
New File:    testresult406.tif:
Tesseract found:
Elm 26,
<->
Gazetteer found:
closest match for: Elm26,
No match
-----
New File:    testresult43.tif:
Tesseract found:

Could not identify any text in image!
-----
New File:    testresult430.tif:
Tesseract found:
,f!!!
<->
Gazetteer found:
closest match for: ,f!!!
No match
-----
New File:    testresult436.tif:
Tesseract found:
Tnb.geagr..
<->
Gazetteer found:
closest match for: Tnb.geagr..
No match
-----
New File:    testresult437.tif:
Tesseract found:
"JM;
<->
Gazetteer found:
closest match for: "JM;
No match
-----
New File:    testresult76.tif:
Tesseract found:

Could not identify any text in image!
-----

```

It is clearly visible by the amount of “Could not identify any text in image!” messages: The Tesseract OCR did not find text within most of the text cut-outs. The images are not further processed, which could help the OCR, but without it, the text in the image cannot be read.

Without any clear results of the OCR, the gazetteer comparison cannot find any close matches, which is why it responds with “No match”.

5.6 Further ideas and future work:

Before and during the development of the algorithm I thought about another ideas and methods, to be implemented and tested, but could not finish any of them due to time constraints:

5.6.1 Planned additions to the program:

- *backwards comparison between gazetteer result and image*

It is definitely advantageous to use the result of the gazetteer search and convert it into an image representation, which can be used as a secondary step for reassuring the correct hit. By comparing the created image of the string to the text-extraction cut-out, a further certainty can be calculated.

- *Bounding box to geographical coordinates*

The implementation of the geographical coordinates of the found toponyms would be a very useful feature, as they are important metadata, which is often lacking. (Yao-Yi Chiang, 2014) Doing this in an automated fashion is however very difficult, as it would require the automated detection of map location, datum and projection. At that point the remaining problem would be to recalculate the found coordinates of the found toponyms to the known projection.

As most map archives underwent the effort to georeference their maps,

There are several solutions for recalculation the coordinates, including any spreadsheet software or GIS solution.

- *more specific position of toponym*

With the given algorithm the toponym coordinates include the centroid of the word bounding polygon, as well as several points along the perimeter (bounding box of text extract) (Weinman J. , Toponym Recognition in Historical Maps by Gazetteer Alignment, 2013), but never the actual position of the object the toponym belongs to. It however requires a multi-variant detection algorithm to connect a toponym to its object, as it often is nothing more than a loose collection of buildings or even a large area within the map. The amount of possibilities, on what the toponym actually depicts is way too high for a simple algorithm to handle. (Budig, Dijk, & Wolff, 2016) did a good effort into such an algorithm.

- *reducing workload by detecting just vocals first*

Apart from general templates like the found double stroke template, just using the vocals for each font used in the map is a good start, as most toponyms have at least a vocal within them. This however would need the manual selection of each vocal as template beforehand.

- *separating letters via opening and closing filters*

Opening and closing filters would be an excellent preprocessing method to reduce the amount of noise and improve the readability of the small letters.

- *using the result of the OCR to refine line features*

Found results from the text extraction and recognition can be used to refine the results of other algorithms like the line feature extractor.

- *creating an UI, that allows for interactive error correction and addition of labels*

As the only result the user is getting from the current state of the software is only a plain text file, it is not nice to error check. It would need another UI, were every toponym and its word area on the original map would be depicted to allow for a human user to cross check the result if necessary and correct erroneous hits.

- *Alternative algorithms*

This is an alternative idea for an algorithm, highlighting the bounding box method, but it falls short to the few colors used in historical maps.

creating bounding boxes for each region of similar color value (saving color information)

calculating size of bounding box

ignoring all boxes above and below a size threshold

selecting boxes with certain colors

selecting boxes with certain background to foreground ratios

selecting boxes with certain Hough lines

grouping spatially close boxes of similar size (at least three) and orientation

group size thresholding

outputting group bounding boxes for OCR

OCR:

taking group for OCR (reorient, noise filtering, color filtering)

dictionary / gazetteer (starting with larger features to close the geographic boundaries in for further searches)

5.7 User interface

As the user interface is supposed to be part of the larger toolbox, which uses a visual programming interface, the interface for the standalone application is kept rather simple to prevent doubling the work.

The interface design had been programmatically set up by former work and is reused for this application.

There are always at least two input text boxes, where the user has to put in the directory path to the used files. Below is a status text field, where outputs or error messages can be communicated to the user. The simple design can be compared to the Esri ArcGIS toolbox dialogs.

There have been efforts to move the modules, including this one, to a visual programming interface. This would simplify chaining various modules together, which allows for multi-step batching processes.

5.8 Target group

The application is aimed at people who frequently use historic maps and want to digitize or organize them. This includes librarians or scientists. We can expect some expertise in terms of maps and their metadata. But we cannot expect to find expertise in programming or visual coding. The application needs to be as self-explanatory as it can be. This includes clearly labeled text fields and simplified steps for each functionality.

6. Conclusion and further research

The results of the software in its current state are mediocre at best. As explained within former chapters this is mainly due to its simplicity: It is only using one parameter - the likeness towards a template - and a size and arrangement limitation to find text. Almost all information about the text have to be known before using the software (size, color, orientation) and a template has to be created externally.

As it is, the effort on creating a software solution on my own did not stand the comparison to what is actually needed to create software remotely capable of handling the given problem of text extraction and recognition.

The proposed algorithms and further ideas might help develop a capable solution, as does the revelation of the size of the topic.

Sources:

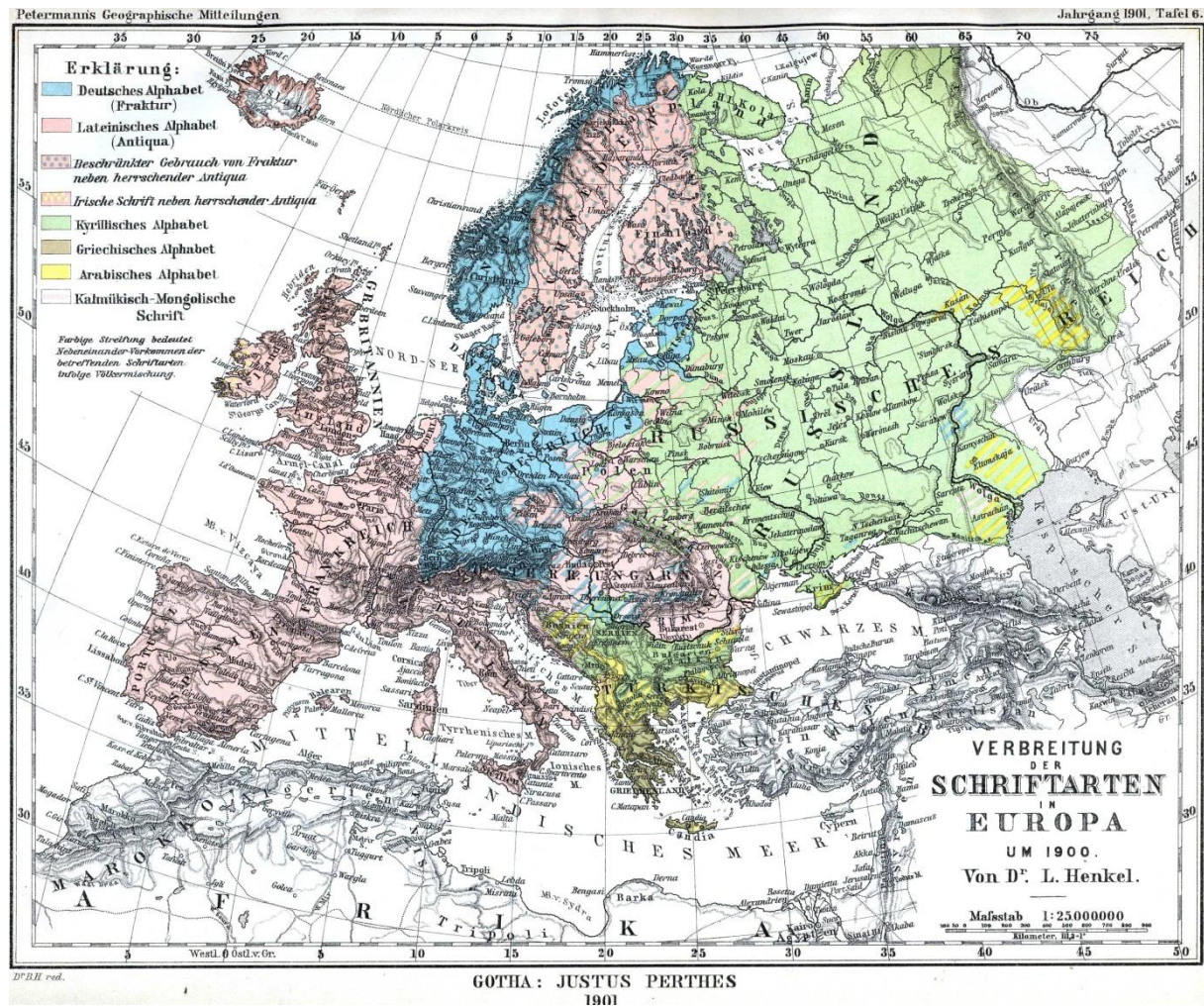
- Acheson, E., Sabbath, S. D., & Purves, R. S. (2017, April 13). A quantitative analysis of global gazetteers: Patterns of coverage for common feature types. *Computers, Environment and Urban Systems*, pp. 309-320.
- Anandhi, N., & Avudaiammal, R. (2017). Segmentation and recognition of text from image using pattern matching. *2017 International Conference on Communication and Signal Processing (ICCSP), Chennai*, 0066-0069.
- Bauer, H. (1993). *Die Kurhannoversche Landesaufnahme des 18. Jahrhunderts*. Hannover: Niedersächsisches Landesverwaltungsamt.
- Blanke, T., Bryant, M., & Hedges, M. (2012). Open source optical character recognition. *Journal of Documentation*, Vol. 68 Issue: 5, pp. 659-683.
- Budig, B., Dijk, T. C., & Wolff, A. (2016, November). Matching Labels and Markers in Historical Maps: An Algorithm with Interactive Postprocessing. *ACM Transactions on Spatial Algorithms and Systems*, Vol. 2, p. Article 13.
- Cao, R., & Tan, C. L. (n.d.). *Text/Graphics Separation in Maps*.
- Chiang, Y.-Y., & Knoblock, C. A. (2010). An Approach for Recognizing Text Labels in Raster Maps. *International Conference on Pattern Recognition*, (pp. 3199-3202).
- Chiang, Y.-Y., & Knoblock, C. A. (n.d.). Recognizing text in raster maps. *Geoinformatica*.
- Coates, A., Carpenter, B., Case, C., Satheesh, S., Suresh, B., Wang, T., et al. (n.d.). *Text Detection and Character Recognition in Scene Images with Unsupervised Feature Learning*. Stanford University.
- Dai, R., Liu, C., & B., X. (2007). Chinese character recognition: History, status and prospects. *Proc. Front. Comput. Sci. China*, pp. 126–136.
- Eckert-Greifendorff, M. (1939). *Kartographie: Ihre Aufgabe und Bedeutung für die Kultur der Gegenwart*. Berlin: de Gruyter.
- Epshtein, B., Ofek, E., & Wexler, Y. (n.d.). *Detecting Text in Natural Scenes with Stroke Width Transform*. 2010.
- Ghorbel, A. (2016). *Generalized Haar-like filters for document analysis : application to word spotting and text extraction from comics*. La Rochelle: Université de La Rochelle.
- Grond, M. M. (2017). *Text detection in natural images using convolutional neural networks*.
- Gupta, A. K. (2017). Detection and Recognition of Text from Image using Contrast and Edge Enhanced MSER Segmentation and OCR. *IJO-Science Volume III Issue III 2017*.
- Impedovo, S., Pirlo, G., Modugno, R., & Ferrante, A. (2010). Zoning Methods for Hand-Written Character Recognition: An Overview. *12th International Conference on Frontiers in Handwriting Recognition*, (pp. pp. 329-334.). Kolkata.

- Issam Bazzi, R. S. (1999, June). An Omnifont Open-Vocabulary OCR System for English and Arabic. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, VOL. 21, NO. 6, 495.
- Iwamura, M., Kobayash, T., & Kise, K. (2011). Recognition of Multiple Characters in a Scene Image Using Arrangement of Local Features. *2011 International Conference on Document Analysis and Recognition, Beijing*, 1409-1413.
- Jacob, C. (2006). *The Sovereign Map: Theoretical Approaches in Cartography Throughout History*. Chicago: The University of Chicago Press.
- Jaderberg, M., Simonya, K., Vedaldi, A., & Zisserman, A. (2015, May 7). Reading Text in the Wild with Convolutional Neural Networks.
- Jaderberg, M., Simonyan, K., Vedaldi, A., & Zisserman, A. (2015). DEEP STRUCTURED OUTPUT LEARNING FOR UNCONSTRAINED TEXT RECOGNITION. *ICLR*, (pp. 1-10).
- Korb, J. (2008). *Survey of existing OCR practices and recommendations for more efficient work*. Austrian National Library.
- Leyk, S., & Boesch, R. (2010, January). Colors of the past: Color image segmentation in historical topographic maps based on homogeneity. *GeoInformatica 14(1)*, pp. 1-21.
- Ni, J. (2017). *A Contribution to Computer-Assisted Reconstruction of Selected Line Features from Scanned Maps*. Dresden.
- Om Prakash Sharma, M. K. (2013). Recent Trends and Tools for Feature Extraction in OCR Technology. *International Journal of Soft Computing and Engineering (IJSCE)*, ISSN: 2231-2307, Volume-2, Issue-6.
- Packer, T. L., Lutes, J. F., Stewart, A. P., Embley, D. W., Ringger, E. K., & Seppi, K. D. (2010). *Extracting Person Names from Diverse and Noisy OCR Text*. Toronto.
- Pezeshk, A., & Tutwiler, R. L. (2010). Improved Multi Angled Parallelism for separation of text from intersecting linear features in scanned topographic maps. *2010 IEEE International Conference on Acoustics, Speech and Signal Processing, Dallas, TX*, 1078-1081.
- Pouderoux, J., Gonzato, J.-C., Pereira, A., & Guitton, P. (2007). Toponym Recognition in Scanned Color Topographic Maps. *Proceedings of ICDAR 2007: 9th International Conference on Document Analysis and Recognition, Sep 2007, Brazil*, 531-535.
- Qixiang Ye, D. D. (2015). Text Detection and Recognition in Imagery: A Survey. *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE*, VOL. 37, NO. 7, 1480ff.
- Smith, R. (2007). An Overview of the Tesseract OCR Engine. *Proc. Ninth Int. Conference on Document Analysis and Recognition (ICDAR)*, IEEE Computer Society, 629-633.
- Spitz, A. L., & Maghbouleh, A. (1999). *Text Categorization using Character Shape Codes*.
- T. Wang, D. J. (2012). End-to-end text recognition with convolutional neural networks. *Proceedings of the 21st International Conference on Pattern Recognition (ICPR2012)*, Tsukuba, 3304-3308.

- Weinman, J. (2013). Toponym Recognition in Historical Maps by Gazetteer Alignment. *2013 12th International Conference on Document Analysis and Recognition, Washington, DC*, 1044-1048.
- Weinman, J. (2013). Toponym Recognition in Historical Maps by Gazetteer Alignment. *2013 12th International Conference on Document Analysis and Recognition, Washington, DC*, 1044-1048.
- Weinman, J. J., Learned-Miller, E., & Hanson, A. R. (2009, October). Scene Text Recognition using Similarity and a Lexicon with Sparse Belief Propagation. *IEEE Trans Pattern Anal Mach Intell. 2009 October* , pp. 1733-1746.
- Y. Watanabe, K. T. (2017). Exploring Old Maps by Finger Tracing of Characters. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR), Kyoto*, 15-19.
- Yao-Yi Chiang, S. L. (2014, April). A survey of digital map processing techniques. *ACM Comput. Surv. 47, Article 1*, 1.

Appendix

Enlarged Version of Figure 2:



From: [https://de.wikipedia.org/wiki/Datei:Scripts_in_Europe_\(1901\).jpg](https://de.wikipedia.org/wiki/Datei:Scripts_in_Europe_(1901).jpg)