Matching Labels and Markers in Historical Maps: An Algorithm with Interactive Postprocessing

BENEDIKT BUDIG, THOMAS C. VAN DIJK, and ALEXANDER WOLFF,
Universität Würzburg

In this article, we present an algorithmic system for determining the proper correspondence between place markers and their labels in historical maps. We assume that the locations of place markers (usually pictographs) and labels (pieces of text) have already been determined – either algorithmically or by hand – and we want to match the labels to the markers. This time-consuming step in the digitization process of historical maps is nontrivial even for humans but provides valuable metadata (e.g., when subsequently georeferencing the map). To speed up this process, we model the problem in terms of combinatorial optimization, solve that problem efficiently, and show how user interaction can be used to improve the quality of the results. We also consider a version of the model where we are given label fragments and additionally have to decide which fragments go together. We show that this problem is NP-hard. However, we give a polynomial-time algorithm for a restricted version of this fragment assignment problem. We have implemented the algorithm for the main problem and tested it on a manually extracted ground truth for eight historical maps with a combined total of more than 12,800 markers and labels. On average, the algorithm correctly matches 96% of the labels and is robust against noisy input. It furthermore performs a *sensitivity analysis* and in this way computes a measure of confidence for each of the matches. We use this as the basis for an interactive system where the user's effort is directed to checking those parts of the map where the algorithm is unsure; any corrections the user makes are propagated by the algorithm. We discuss a prototype of this system and statistically confirm that it successfully locates those areas on the map where the algorithm needs help.

CCS Concepts: • **Theory of computation** → **Network optimization**; • **Information systems** → *Geographic information systems*; *Digital libraries and archives;*

Additional Key Words and Phrases: Historical maps, algorithms, metadata extraction, interaction, sensitivity analysis

ACM Reference Format:
Benedikt Budig, Thomas C. van Dijk, and Alexander Wolff. 2016. Matching labels and markers in historical maps: An algorithm with interactive postprocessing. ACM Trans. Spatial Algorithms Syst. 2, 4, Article 13 (November 2016), 24 pages.
DOI: http://dx.doi.org/10.1145/2994598

1. INTRODUCTION

Historical maps are a valuable source of information for researchers in various scientific disciplines. With the progressing digitization of libraries and archives, these maps become available to a larger number of scholars. Figure 1 shows a section of a map from the early 18th century. It gives an impression of the wealth of information contained in such maps: different pictographs indicating various kinds of settlements, labels for the pictographs (lettered in various ways), areas indicated by color or by little trees or hills,

Authors' address: Lehrstuhl für Informatik I, Universität Würzburg, Am Hubland, 97074 Würzburg; emails: {thomas.van.dijk, benedikt.budig, alexander.wolff}@uni-wuerzburg.de. http://www1.informatik.uni-wuerzburg.de/en/.

DOI: http://dx.doi.org/10.1145/2994598

Fig. 1. Section of a historical map from the early 18th century, containing a variety of information, including rivers, woodlands, hills, political borders, and (labeled) settlements.

Fig. 2. Place markers and place labels on several historical maps. Note the variety of visual styles, both in the pictographs and the lettering.

labels for the areas, and more. In order to make maps more easily retrievable, metadata describing all this information needs to be extracted. An example of such metadata is a georeferenced index of the contained geographical features (such as labeled cities and rivers). However, digitizing historical maps and analyzing their contents is a complex and time-consuming process. For example, it currently takes the Würzburg University Library more than 25 hours to georeference the 1,000 to 4,000 places contained in a typical map from their collection.[1]

The task of extracting information from historical maps is mostly performed manually by experts. Automated tools are scarce for a variety of reasons. For one, there is a large variety of drawing styles in historical maps. This makes it hard for a single algorithm or software tool to perform well on a large set of maps: Consider Figure 1 and contrast the other examples in Figure 2. There is also the issue of correctness: In general, algorithms for extracting semantic information from bitmap images are far from perfect. This is to be expected since these problems are truly difficult for computers. To the curators of some historical map collections, however, the correctness of metadata is of paramount importance (and, in no small measure, a matter of pride).

In this article, we concern ourselves with one specific step in the metadata extraction process: the matching of place labels to place markers. By *marker* we mean a map element – typically a pictograph – indicating the geographic position of a point of interest. A *label* is a piece of text in the map that indicates the name belonging to

[1]Personal communication with Dr. H.-G. Schmidt, head of the Manuscripts and Early Prints department, Würzburg University Library.

a certain marker. In our experiments, both markers and labels are represented by axis-aligned bounding rectangles, but the algorithms support arbitrary assignment costs.

The question is, then: Which label belongs to which marker? This is in fact a nontrivial problem, even for humans. (Look ahead at Figure 13 for a tricky situation that we will discuss in more detail later.) In this article, we propose a solution to this problem that works on various kinds of maps. We have evaluated our algorithm on eight maps from the early modern period onward.

We focus on a specific subtask of the digitization process in order to get a manageable problem. This modular approach, with subgoals more modest than "understand this map," allows for rigorous problem statements and, thereby, reproducible experiments and comparability; this is in contrast to monolithic software systems, where it can be unclear how any specific detail influences the outcome. Competing systems for a certain step can then be proposed and evaluated. Such a "separation of concerns" is also advocated, for example, by Shaw and Bajcsy [2011] and Schöneberg et al. [2013].

We present related work in Section 2. After introducing our algorithm for matching labels and markers (Section 3), we present several experiments that show that the algorithm performs well on the eight historical maps that we have tested on (Section 4). Next, we present different extensions to this work. The first is an interactive postprocessing method that detects situations in which our algorithm was uncertain and shows them to a user for verification or correction (Section 5). Second, we explore a different direction by extending our initial problem formulation to matching markers and sets of label fragments (Section 6). We prove that this problem is NP-hard in general but solve a restricted version of the problem in polynomial time.

2. RELATED WORK

Since the digitization and analysis of historical maps is of increasing interest to libraries and other collectors, systems simplifying this complex process have been developed. Most of these systems provide convenient graphical interfaces but still rely heavily on users to manually annotate or even georeference the input maps (e.g., the *Georeferencer* by Fleet et al. [2012] and the systems by Simon et al. [2011, 2015]). For the postprocessing of georeferenced maps, Jenny and Hurni [2011] introduced a tool that is able to analyze the geometric and geodetic accuracy of historical maps and then visualize the identified distortions. Recently, Chiang [2015] expressed the need for a framework that would allow querying historical maps as a unified spatiotemporal data source, an effort that requires an in-depth extraction of metadata from historical maps.

The segmentation of bitmap images is a rich field of research. Some of it specifically concerns (historical) maps. Höhn [2013] introduced a method to detect arbitrarily rotated labels in historical maps; Mello et al. [2012] dealt with the similar topic of identifying text in historical maps and floor plans. Simon et al. [2014] applied image processing techniques and a combination of different heuristics to identify toponyms in historical maps. Budig and van Dijk [2015] introduced a system that combines template matching with machine learning to detect occurrences of predefined pictographs and characters in historical maps. While this approach works well for individual characters, it is not trivial to link the detected characters so that we obtain correct text labels. (We explore this problem in the context of the present work in Section 6.) All the listed methods have been developed specifically for historical maps and could be used to generate input data for the algorithm presented in this article.

Focusing on the processing of more recent maps (i.e., maps from the 19th century onward), Chiang et al. [2014] published an extensive survey covering a variety of image-based techniques. These maps have high quality when compared to the older maps, having been created and printed to modern cartographic standards. Also, older maps

often suffer from physical degradation. An example of this is the work on recognizing text in raster images by Chiang and Knoblock [2015]. While the authors conducted experiments on modern maps only, their approach might be transferable to our (much older) historical maps.

Another thread of research is performed under the term *word spotting*, where the task is to locate written language in bitmap images. It is considered a positive quality if this is performed in a language-agnostic way. (Consider that humans can usually identify written text, even when unable to read the language or when unfamiliar with the font or even the script.) Rath and Manmatha [2007] do clustering based on connected components of "ink." It is common to apply warping techniques such as Dynamic Time Warping (DTW) to align various copies of the same word, using *projection profiles* [Rath and Manmatha 2003; Moghaddam and Cheriet 2009]. The primary concern in the word spotting literature has been its application to manuscripts. Our own preliminary tests indicate that word spotting may be applicable to historical maps.

A further approach to locate text labels in historical maps is the use of Optical Character Recognition (OCR). However, existing methods for OCR do not perform well on "natural scenes" such as photographs [Epshtein et al. 2010]. This is relevant because, in terms of background noise and distracting image elements, scanned historical maps can be closer to natural scenes than to the text-on-a-page setting that might be expected for OCR. In the context of natural scenes, Epshtein et al. [2010] have introduced the *stroke width transform* image operator. Their method is purely image-based and language agnostic. It does not perform OCR as such, but instead aids in the preprocessing step of determining *where* the text is. They report a preliminary experiment that shows that this significantly increases the performance of a subsequent OCR step. This two-step approach of first recognizing where the text is and *then* trying to read it is quite common.

In contrast to this, Wang et al. [2011] report higher performance when using an integrated approach, directly looking for certain words in an image. Their approach does require a list of possible words as input (a lexicon). This may limit the applicability of their approach: While they sketch several scenarios where the availability of a reasonably sized lexicon is realistic, historical maps may not be one of them. The spelling of place names across historical maps is notoriously inconsistent. For historical maps of Germany, one could could use the Integrated Authority File (*Gemeinsame Normdatei*, GND), which lists many historical spellings of geographical place names. For example, its alternatives for Würzburg include Wurzbourgh, Wirtzburg, and Herbipolis. However, the size of this database (over 2 GB) is likely to make its use as a lexicon in their approach infeasible. In contrast, the system of Weinman [2013] is explicitly designed to take large gazetteers into account, in combination with spatial considerations. Using a probabilistic reasoning, his approach aligns image toponyms with known places from the gazetteers but requires their location on the map as an input.

In summary, we should note that while there are many promising approaches, producing the input required for our method (bounding rectangles for all markers and labels) is in general not a solved problem.

There is little research available on fully algorithmic information retrieval specifically from historical maps. Automatic approaches exist, but only for restricted inputs (i.e., developed specifically to digitize a particular corpus). For example, Leyk et al. [2006] describe a method to find forest cover in a specific set of 19th century topographic maps. Arteaga [2013] extracts building footprints from a set of historic maps from the New York Public Library (NYPL), particularly georectified scans of insurance atlases published in the 19th and early 20th centuries. The effectiveness of these approaches is in part due to the homogeneity of these relatively recent maps. The tests in this article are performed on much older maps (16th and early 18th century). For detecting and georeferencing places in such early maps, Höhn et al. [2013] propose a

system that finds place markers and suggests possible place names based on both a modern-day map and markers that the user has previously identified. While a modern-day map can certainly be a valuable source of information, in this article, we take a different approach to identify contained places: finding the corresponding text labels in the map.

3. MATCHING MARKERS AND LABELS

In our model, both markers and labels are represented by axis-aligned bounding rectangles. This is a reasonable simplification on many maps but could easily be generalized if needed (rotated rectangles, arbitrary polygons, etc.). We assume these rectangles are available to the algorithm from some earlier step in a pipeline: Let P be the set of markers contained in a historical map, and let L be the set of contained labels. Recall that our goal is to identify the correct correspondence between place labels and place markers. We assume that this correspondence is a *matching*: Every $p \in P$ is assigned to at most one $\ell \in L$, and every $\ell \in L$ is assigned to at most one $p \in P$. However, we do not assume that there is a one-to-one correspondence: Indeed, all eight maps we have tested contain unlabeled markers or stray labels. Not having the one-to-one assumption also provides robustness in case not all markers and labels were correctly identified earlier in the pipeline.

The basis for our algorithm is the simple observation that labels are generally positioned *near* the marker they belong to. This is the basic assumption underlying our matching model. For a marker $p \in P$ and a label $\ell \in L$, we define the distance $d(p, \ell)$ as the Euclidean distance[2] between the rectangles (i.e., the smallest distance between a point in p and a point in ℓ). This distance can be easily determined. In addition, we assume that labels are never located more than some distance r from the marker they belong to, which helps in deciding whether a marker is unlabeled. One could worry that this parameter r has to be chosen carefully because an insufficiently large value might disallow the correct matching. In practice, a suitable value is easily found: See Section 4.4 for an experimental discussion.

Let a *matching* be a set of pairs of a marker in P and a label in L such that every marker and every label occurs at most once. (This is indeed the graph-theoretical concept of a matching in the complete bipartite graph between P and L.) Our goal is now to find a matching $M \subseteq P \times L$ such that:

(C1) The size of the matching M is large.
(C2) The sum of the distances in the matching is small; that is, the sum of $d(p, \ell)$ over all $(p, \ell) \in M$ is small.
(C3) No match $(p, \ell) \in M$ has distance $d(p, \ell) > r$.

We choose to minimize the sum of distances rather than, for example, to minimize the maximum distance since, in that case, a single distant assignment would allow all shorter assignments to be almost arbitrary.

A first attempt at achieving these goals might be to use a greedy algorithm that simply matches the closest label-marker pair and repeats, but this can perform very poorly in the worst case. See Figure 3 for an instance where the greedy algorithm gets everything wrong. These gadgets can be repeated to get arbitrarily large instances with this behavior. This construction is somewhat contrived, but not entirely unrealistic. In fact, we will see in Section 4 that the greedy algorithm performs poorly in practice.

A natural way to combine these criteria into a proper optimization objective is as follows: Any pair $(p, \ell) \in M$ gives us some fixed benefit (criterion C1), but also has a

[2]It is also possible to use other distance measures, for instance squared Euclidean, L_1, L_∞, etc., but the Euclidean distance (as a natural choice) already yields good results in our experiments; see Section 4.

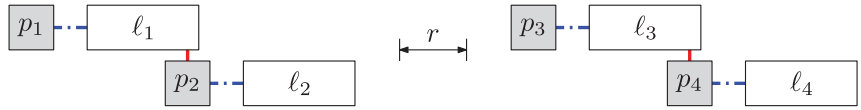


Fig. 3. A situation in which the greedy algorithm performs poorly. It returns the solid red lines matching $\{(p_2, \ell_1), (p_4, \ell_3)\}$, which consists purely of incorrect matches. Note that for larger values of r, the greedy algorithm would additionally match (p_1, ℓ_2) and (p_3, ℓ_4), which are also incorrect. In contrast, an optimal solution in terms of our optimization objective yields the correct matching (dashed blue lines). This solution does not change for larger values of r.

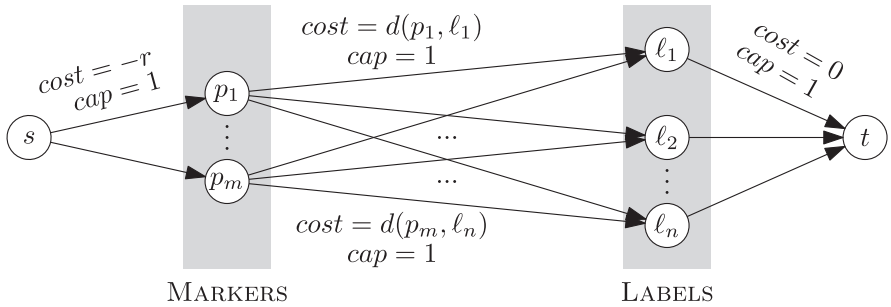


Fig. 4. The flow network G.

cost $d(p, \ell)$ (criterion C2). We ensure criterion C3 by setting the benefit equal to r. This is expressed by the following formula:

$$\text{Maximize } f_{\text{obj}}(M) = \sum_{(p,\ell)\in M} (r - d(p, \ell))$$

We define the LABEL ASSIGNMENT problem as maximizing f_{obj} subject to M being a matching. Note that criterion C3 will always hold in an optimal solution since any pair (p, ℓ) with $d(p, \ell) > r$ in M decreases the objective value. The parameter r thus has another interpretation: It limits the "marginal cost" of adding an additional match (p, ℓ) to M; that is, r tells us by how much the sum of distances in M is allowed to increase in order to match an additional label.

An alternative to the weighted cost-benefit approach would be to consider just the costs (criterion C2) and compute a Pareto frontier for matchings of different cardinality. We have not investigated this.

	LABEL ASSIGNMENT
Instance:	A set P of place markers.
	A set L of labels.
	A distance function $d(p, \ell)$: $P \times L \to \mathbb{R}^+$.
	A parameter $r \in \mathbb{R}^+$.
Objective:	Find a matching M of place markers and labels such that $f_{\text{obj}}(M)$ is maximized.

We solve the LABEL ASSIGNMENT problem using the flow-based approach illustrated in Figure 4. Let $G = (V, E)$ be a directed acyclic graph with $V = \{s\} \cup P \cup L \cup \{t\}$. It has a source s with arcs toward all nodes in P. All nodes in P have arcs to all nodes in L. Finally, all nodes in L have an arc to the sink t. With capacity 1 everywhere, this is the standard flow network to model bipartite matching [Cormen et al. 2009]. We reduce

our problem to a *minimum cost* flow problem on G by translating our maximization problem into a minimization problem and setting arc weights accordingly. Each arc (s, p) leaving s has $cost(s, p) = -r$: This corresponds to a benefit of r for establishing a match. For each arc (p, ℓ) we set $cost(p, \ell) = d(p, \ell)$: the distance-based cost. Each remaining arc (ℓ, t) has $cost(\ell, t) = 0$. Then, a flow in G corresponds precisely to a solution M of LABEL ASSIGNMENT, where the flow cost equals $-f_{\text{obj}}(M)$.

Finding a flow of minimum cost over all admissible flows in G gives an optimal solution for the model just introduced: Marker p and label ℓ are matched if and only if the flow value of edge (p, ℓ) is 1. This minimum-cost flow problem can be solved in polynomial time using standard algorithms (e.g., Goldberg [1997]). Note that the standard formulation of minimum-cost flow requires a flow demand to be given as input and will minimize the cost over flows of exactly this value. Instead, we want the minimum cost over all admissible flows – of any value. This problem variant can still be solved efficiently.

Since our instances have integer capacities, we can also use (fractional) linear programming to find an optimal solution: This is valid because the fractional flow LP is integer when the capacities are integer [Schrijver 2003]. This allows us to use off-the-shelf fractional LP solvers. It may sound unintuitive that this would be more efficient in practice, but, having implemented both approaches, we found that our LP-based implementation significantly outperformed the one based on a combinatorial minimum-cost flow library. The following experiments have therefore been run using the former implementation.

4. EXPERIMENTS

We implemented the algorithm just described for experimental evaluation. The experiments were run on a desktop PC with a 3.40GHz quad-core CPU; memory was no issue. We used CPLEX v12.5.1 for solving linear programs.

4.1. Data and Ground Truth

We tested on historical maps from the *Franconica* collection[3] maintained by the Würzburg University Library. The collection contains more than 800 maps created between the 16th and 19th centuries, many featuring several thousand place markers. For this article, we manually extracted all markers and labels contained in six full maps from this collection:

—the *Franckenland* map[4] from 1533,
—the *Franciae Orientalis* map[5] created between 1570 and 1612,
—the *Nova Franconiae* map[6] from 1626,
—the *Franconia Vulgo* map[7] from 1650,
—the *Bisthum Würzburg* map[8] from 1676, and
—the *Circulus Franconicus* map[9] from 1706.

[3] http://franconica.uni-wuerzburg.de/.
[4] Sebastian von Rotenhan. *Das FranckenLandt = Chorographi Franciae Orien[talis]*, 1533. 36/G.f.m.9-14,136.
[5] Sebastian von Rotenhan and Abraham Ortelius. *Franciae orientalis (vulgo Franckenlant) descriptio*, between 1570 and 1612. 36/A 20.39.
[6] Abraham Goos. *Nova Franconiae descriptio*, 1626. 36/G.f.m.9-12,139.
[7] Willem Janszoon Blaeu. *Franconia Vulgo Franckenlandt*, 1650. 36/A 10.19.
[8] Johann Heinrich Seyfried and Johann Jakob Schollenberger. *Das Bisthum Wurtzburg In Francken*, 1676. 36/A 10.12.
[9] Frederik De Wit. *Circulus Franconicus: in quo sunt episcopatus Wurtzburg, Bamberg et Aichstet, Status Equitum Teutonicor(um), Ducatus Coburgensis, Marchionatus Cullembach et Onspach, Comitatus Henneberg,*

Fig. 5. Examples of unlabeled markers (top row) and stray labels (middle row). The three situations in the bottom row show limitations of our modeling. On the bottom left, there are two labels sharing a common word ("Ertal"), thus not allowing for a matching. In the middle, a label is split in three parts that are arranged around the corresponding place marker, leading to an excessively large bounding box. On the bottom right, some labels use leaders pointing toward their corresponding markers, information that is not handled by our model. Exceptional situations like those in the bottom row occur a small number of times on each of the tested maps.

For two additional maps from the collection, we extracted markers and labels from a rectangular crop containing approximately 700 map elements each:

—the *Carte Topographique D'Allemagne* map[10] from 1787 and
—the *Fürstenthum Würzburg* map[11] from 1805.

This set covers nearly a three-century span of historical maps and was chosen to contain a variety of visual styles.

For all maps, we matched markers and labels by hand and use this as a ground truth for our experiments. To ensure that the ground truth was not influenced by knowledge of our algorithmic modeling, this annotation task was performed by a teaching assistant who was unaware of the work in this article. He was instructed to use his best effort to resolve ambiguous cases (as opposed to making historical inquiries using external sources). All these maps contained some unlabeled markers or stray labels. Examples of these, as well as some situations that show the limitations of our algorithmic modeling, are given in Figure 5.

Wertheim, Holach, Reinec, Papenheim, Erpach, Schwartzenberg, et Castel, Baronatus Sensheim et Territorium Norinbergense, 1706. 36/A 1.17.

[10]Daniel Adam Hauer. *Carte Topographique D'Allemagne Contenant une Partie de l'Evêchés de Wurtzbourg et Bamberg et Fulde, les Duchés de Saxe Cobourg, Gotha, Meinungen, Hildbourgshausen et une Partie de Saxe Weimar, le Comté de Schwartzbourg, le Baillage de Smalcalden, le Territoire de Schweinfurt*, 1787. 36/A 1.16-41.

[11]Carl von Fackenhofen. *Das Fürstenthum Würzburg*, 1805. 36/A 50.8.

Table I. Statistics of Our Experimental Results for the Maps *Franckenland* (FL), *Circulus Franconicus* (CF), *Bisthum Würzburg* (BW), and *Fürstenthum Würzburg* (FW)

Experiment	FL1	FL2	CF1	CF2	BW1	BW2	FW1	FW2
Number of markers	517	539	1,644	1,663	1,150	1,190	347	349
Number of stray markers	—	22	—	19	—	40	—	2
Number of labels	517	524	1,644	1,669	1,150	1,158	347	347
Number of stray labels	—	7	—	25	—	8	—	0
Correct matches	515	503	1,636	1,630	1,123	1,076	339	337
Incorrect matches	2	14	8	16	27	77	8	10
Correctly unassigned markers	—	18	—	14	—	22	—	1
Correctly unassigned labels	—	6	—	22	—	4	—	0
Error measure	0.8%	5.4%	1.0%	1.8%	4.5%	12.3%	4.5%	5.6%
Runtime	0.6 s	0.6 s	1.7 s	1.8 s	1.1 s	1.1 s	0.5 s	0.5 s
Greedy error measure	30.5%	28.0%	9.7%	10.1%	36.9%	35.4%	23.9%	23.8%

Unless otherwise noted, we used a fixed value of $r = 150$ px on all maps. (We discuss this value in Section 4.4; for now, see Figure 12 for an indication of scale.)

4.2. Balanced Case

First, we run experiments with our algorithm on *balanced* input data. This means that the ground truth data is a one-to-one assignment: This input data admits a *perfect matching*. This is not the case in all historical maps, even if our input P and L perfectly model the actual contents of the map: These experiments are run on a version of the ground truth where we have manually removed a small number of unlabeled markers and stray labels. We define the *error measure* of our experiments as calculating the Jaccard distance[12] between the set of assignments returned by the algorithm and the set of assignments from the ground truth. This definition is chosen for comparability with further experiments presented in the next section. Note that, for balanced input data, this error measure is two times the *precision*; that is, the ratio of correct assignments to all assignments returned by the algorithm.

The filtered input data for the *Franckenland* map thus consists of 517 markers and labels. Our algorithm matches 515 labels correctly and makes 2 incorrect matches (experiment FL1). This took 0.6 seconds of runtime. On one map (*Franconia Vulgo*), the algorithm is able to assign all markers and labels correctly without making any mistakes (experiment FV1). The algorithm performs worst on the *Carte Topographique D'Allemagne* map, where 19 of the 369 calculated assignments are incorrect (experiment CT1). Tables I and II contain these and further statistics; the experiments referred to in this section have the suffix "1."

In three of our eight experiments, the error measure is equal to or below 1%. For another three, it is below 5%. The remaining two experiments (5.4% and 9.8%) suffer from areas with particularly dense element placement, which causes the bounding boxes of some labels to overlap with each other and multiple markers. In this situation, all affected elements have distance 0 and, as a consequence, are assigned arbitrarily. Improved assignment costs $d(p, \ell)$ – not based on axis-aligned bounding boxes – might be able to solve this problem.

The average error measure of the experiments is 3.7%, which we consider a good result given the dense and sometimes inconsistent placement of map elements. For comparison, we also implemented the greedy algorithm discussed earlier. Recall that it

[12]Based on the Jaccard index [Jaccard 1912], the Jaccard distance d_J between two sets A and B is defined as $d_J(A, B) = (|A \cup B| - |A \cap B|) / |A \cup B|$.

Table II. Statistics of Our Experimental Results for the Maps *Franciae Orientalis* (FO), *Franconia Vulgo* (FV), *Carte Topographique D'Allemagne* (CT), and *Nova Franconiae* (NF)

Experiment	FO1	FO2	FV1	FV2	CT1	CT2	NF1	NF2
Number of markers	536	549	851	868	369	374	906	925
Number of stray markers	—	13	—	17	—	5	—	19
Number of labels	536	538	851	851	369	374	906	907
Number of stray labels	—	2	—	0	—	5	—	1
Correct matches	526	517	851	848	350	342	881	871
Incorrect matches	10	20	0	3	19	29	25	35
Correctly unassigned markers	—	7	—	14	—	2	—	15
Correctly unassigned labels	—	0	—	0	—	0	—	1
Error measure	3.7%	7.0%	0.0%	0.7%	9.8%	14.1%	5.4%	7.4%
Runtime	0.6 s	0.6 s	0.8 s	0.8 s	0.6 s	0.6 s	1.0 s	1.0 s
Greedy error measure	35.6%	33.8%	14.2%	15.0%	27.2%	27.7%	28.4%	32.9%

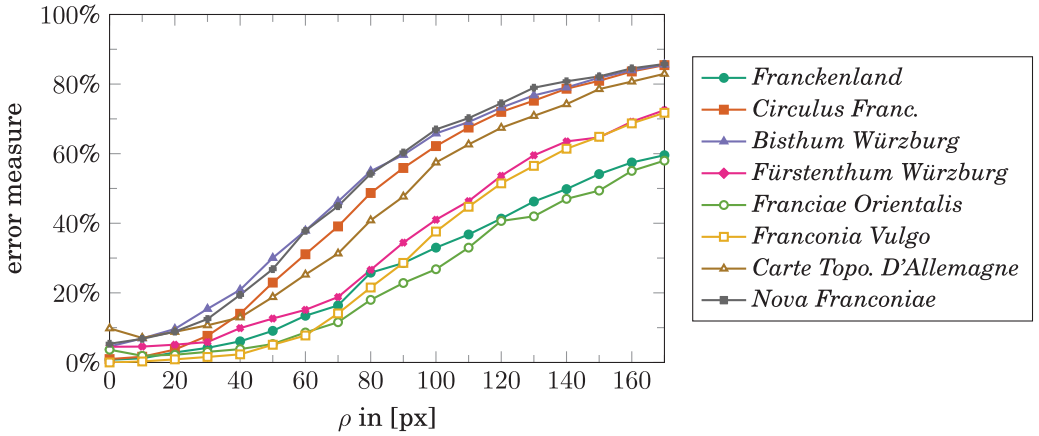


Fig. 6. Effect of ρ on the error measure. For $\rho \leq 30$ px, the graphs are nearly horizontal and below 15%. Every data point corresponds to the average error measure over 10 runs of the experiment.

iteratively adds a match with smallest distance to the matching. The greedy algorithm has its smallest error measure on the *Circulus Franconicus* map (9.7%) and its highest on the *Bisthum Würzburg* map (36.9%); its average error measure is 25.8%. We conclude that the greedy algorithm is unsuitable for this matching task and that the results of our algorithm are indeed nontrivial. Note that the error measure of our algorithm is similar on most maps, whereaas the greedy algorithm tends to perform particularly badly on some of the older maps (like *Franckenland* and *Franciae Orientalis*). This is likely due to the labeling style used in these maps: It requires some combinatorial inference, which our algorithm is able to do but the greedy algorithm is not.

Since our algorithm is intended for use as part of a semi-automatic digitization pipeline, we cannot assume the input to be absolutely accurate. This is especially true for the detection of labels (text), where some characters are easily missed by existing algorithms (see, e.g., Höhn [2013]). In the next set of tests, we take this into account by introducing *positional noise*. Based on the ground truth data, we shifted all labels by some offset, each label independently, uniformly at random from $[-\rho, \rho] \times [-\rho, \rho]$, for some real parameter ρ. We then run the algorithm repeatedly with different values of ρ (10 runs per value of ρ). In Figure 6, we observe that our algorithm copes well with positional noise when the distances by which labels are shifted are realistic. For

example, for $\rho \leq 30$ px, the error measure stays below 15% on all maps. This is the width of approximately one to three characters in an average label in the maps, which we consider a reasonable margin for imprecision of the input.

4.3. Imbalanced Case

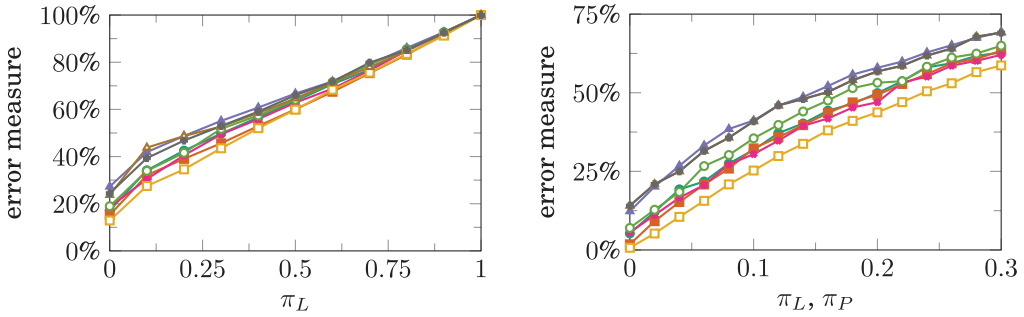
Historical maps often contain a small number of unlabeled place markers and stray labels; in fact, this holds for all eight maps we created ground truth for. Also, when integrating our approach into a (semi-)automated digitization process, the preceding steps might have missed some of the map elements. In the upcoming experiments, we assess the performance of our algorithm in such situations. Again, we use the manually created ground truth for the eight maps, but this time we do not filter P and L to have a one-to-one assignment.

Note that, in this setting, possible errors additionally include unlabeled markers and stray labels that are falsely assigned to another map element (instead of being left unmatched). Vice versa, map elements that do have a corresponding element may erroneously be left unmatched by the algorithm. By using the Jaccard distance, we take both of these error types into account when calculating the error measure.

The input data based on the *Franckenland* map now contains 539 markers and 524 labels; we determined manually that 22 markers and 7 labels do not have a counterpart. Our algorithm gives a matching of size 517, which contains 503 correct matches (experiment FL2). Of the 14 incorrect matches, four assign markers that are actually unlabeled and one assigns one stray label. The remaining nine incorrect assignments involve only regular place markers and labels. Conversely, 6 out of 7 stray labels and 18 out of 22 unlabeled markers are correctly left unassigned. Taking all errors into account, we get an error measure of 5.4%, with a runtime of 0.6 s. Doing the same for the *Circulus Franconicus* map, we have an input of 1,663 markers and 1,669 labels, with 19 unlabeled markers and 25 stray labels. The error measure in this experiment (experiment CF2) is 1.8%, with a runtime of 1.8 s. Note that the error measure on this map is considerably lower than on the *Franckenland* map. As stated in Section 4.2, the labeling in that map has a more complicated structure, which is further complicated by including the stray map elements. The experiments on the remaining maps show similar error measures (on average 6.8%), with the *Carte Topographique D'Allemagne* again being worst (14.1%). On all maps, error measures are higher than for the balanced case, but we still consider the results to be of high quality. The greedy algorithm performs poorly in this imbalanced setting as well, returning matchings with average error measures of 25.8%. Further statistics are provided in Tables I and II; the experiments referred to in this section have the suffix "2." Figure 7 shows an example of an incorrect assignment returned by our algorithm in this set of experiments.

The imbalanced case allows us to address imprecision in the input beyond positional noise: missing elements. We tested this scenario on artificial instances, starting with the ground truth and removing each element from P with probability π_P and each element from L with probability π_L. In several experiments, we varied values for both probabilities (see Figure 8). On all maps, error measures immediately increase with π_P and π_L: Even with low values for these probabilities, the errors are not mitigated by the algorithm. This is not surprising because the algorithm simply cannot match map elements that it does not know about, but it is still scored against the full ground truth. Assuming that a previous processing step has failed to detect such elements, this experiment reflects a realistic setting. This effect is in contrast to the positional noise, where some noise was tolerated. Still, the algorithm is able to correctly identify the *resulting* situation, where some labels and markers have become unmatchable. With increasing deletion probability, the number of correct matches decreases linearly,

Fig. 7. Example of an erroneous assignment in experiment CF2: The marker for *Mittelstrew* is matched to the label *Unsleben* and vice versa. The map image (left) visually suggests that *Obrenstrew*, *Mittelstrew*, and *Unsleben* are each labeled across the river. This assumption can be confirmed by modern map data, since the three villages still exist. Considering the manually created bounding boxes (center), the distances between markers and labels caused by the river become apparent. This, together with the crowdedness of the area, causes the algorithm to fail: It assigns labels vertically along the river rather than across it (right). Also note the unlabeled marker above the *Mittelstrew* label.



(a) Impact of π_L on the error measure (with $\pi_P = 0.1$)

(b) Impact of $\pi_P = \pi_L$ on the error measure

Fig. 8. Impact of randomly removing labels and markers on the error measure in different settings. For a legend to the plots, see Figure 6.

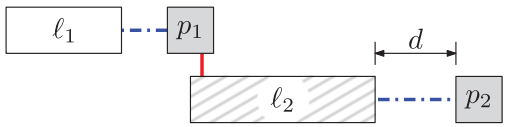


Fig. 9. When using parameter $r < d$, the marker p_2 is too far away from label ℓ_2 to be considered. This results in the matching $\{(p_1, \ell_2)\}$. With a higher value of r, the matching $\{(p_1, \ell_1), (p_2, \ell_2)\}$ is found, which is presumably correct.

but the error measure in the remaining instance is almost unaffected: Missing some elements does not significantly "confuse" the algorithm.

4.4. Parameter Choice

The quality of the matching relies to some extent on a reasonable choice of the parameter r: Our algorithm will not assign labels that belong to markers with distance greater than r. Due to the combinatorial nature of the problem, this can also influence the assignment of markers and labels that are less than r apart (see Figure 9). Picking a value of r that is too low can clearly be a problem in this way. Less intuitively, r can also be too high: One can construct instances such that our algorithm has an error measure of 0% for some r and 100% for higher r (see Figure 10 for an example). However, such instances are unlikely to occur in practice: An experiment on balanced input data with fixed positional noise $\rho = 40$ px shows that the error measure does not increase

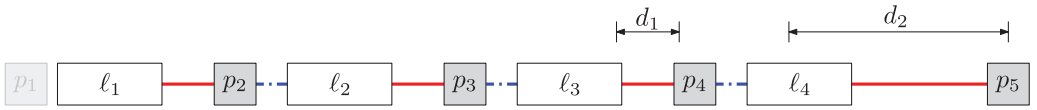


Fig. 10. A situation in which increasing r leads to an incorrect matching. Assume that p_1 exists on the map but was not correctly detected and is thus not part of the input. Also assume that p_5 is unlabeled on the actual map. For $r = d_1$, our algorithm returns the correct matching $\{(p_2, \ell_2), (p_3, \ell_3), (p_4, \ell_4)\}$ (dashed blue lines). In contrast, for $r = d_2$, our algorithms "flips" those matches and returns the entirely incorrect matching $\{(p_2, \ell_1), (p_3, \ell_2), (p_4, \ell_3), (p_5, \ell_4)\}$ (solid red lines).

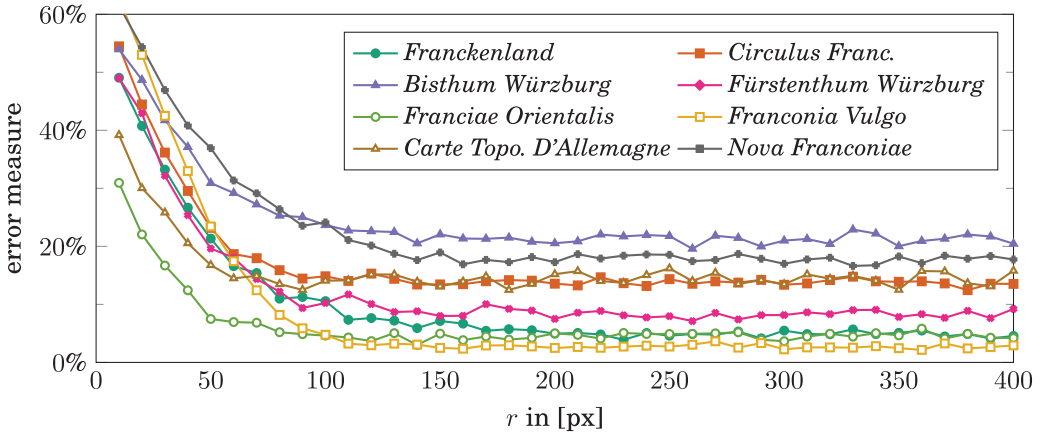


Fig. 11. Effect of r on the error measure ($\rho = 40$ px). Every data point corresponds to the average error measure over 10 runs of the experiment.

significantly for high values of r (Figure 11). In this experiment, even for $r = 1000$ px, the error measure stays at approximately the same level as for $r = 150$ px. In contrast, a small value of r (in this case, below 60 px) does lead to many errors.

Arcs in the flow network with cost larger than r cannot be part of an optimal solution and can therefore be excluded when running the algorithm. In this way, a low value of r leads to a lower runtime since the flow network G is smaller. With $r = 150$ px, our algorithm runs experiment CF1 in 2.1 seconds; with $r = 1000$ px, this increases to 11.9 seconds, even though the algorithm finds the exact same matching. This illustrates that r should not be set arbitrarily high.

A reasonable value for r can usually be determined visually by a user of the system before running the algorithm. For example, we arrived at $r = 150$ px by observing that the distance between a label and the corresponding marker in our test maps is typically limited to 2 or 3 times the average text height (Figure 12) and picking r to be larger by a significant margin. The dense placement of elements in some maps (e.g., *Circulus Franconicus*) would also allow a lower value of r without affecting the returned matching; for example, 100 px.

5. INTERACTIVE POSTPROCESSING

In general, historical maps have situations where it is unclear (even to a human reader without domain knowledge) how the markers and labels belong together. This also occurs on most of the maps in our experiments. Changing a single match in such situations can propagate to other matches. Figure 13 shows an example where three

Fig. 12. Value of r on *Franckenland* (left) and *Circulus Franconicus* (right). The red boundary marks a distance of 150 px from the blue bounding box.

Fig. 13. A difficult case: Without geographic or historical context, it is hard to tell which one of these three matchings is correct.

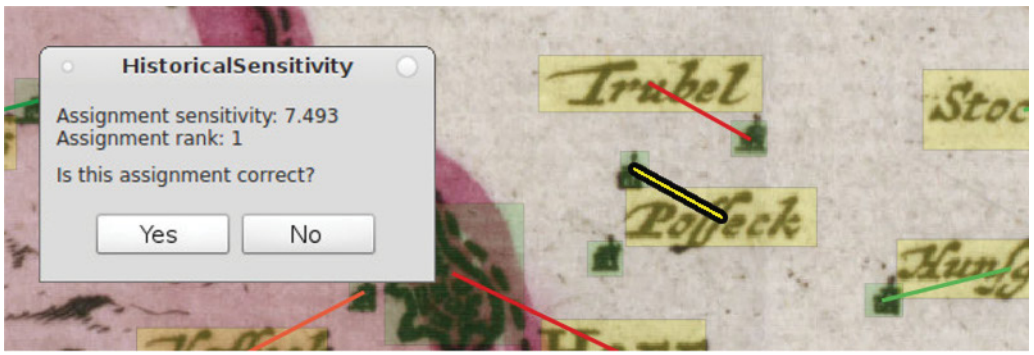


Fig. 14. Screenshot of our prototype for interactive postprocessing. The yellow matching in the map is presented to the user for verification.

distinct matchings seem reasonable: The correct matching is unclear without additional topographic or historical information.

Figure 14 shows a screenshot of our prototype plug-in for the open-source geographic information system QGIS.[13] Our tool automatically presents areas of the map that the algorithm is most unsure about and asks the user for confirmation. (How these areas are picked is described after the example.) The user's confirmation or correction can then be taken into account for a new run of the algorithm. Note that the indicated matching in Figure 14 is indeed unclear. There are three markers near the label *Posseck,* and one clearly belongs to the label *Trubel*: The algorithm assigns this correctly. For lack of another reasonable label, however, one of the two remaining markers must remain unlabeled. Purely from the image, it is unclear which one, so a user with domain knowledge must get involved.

[13]http://www.qgis.org/.

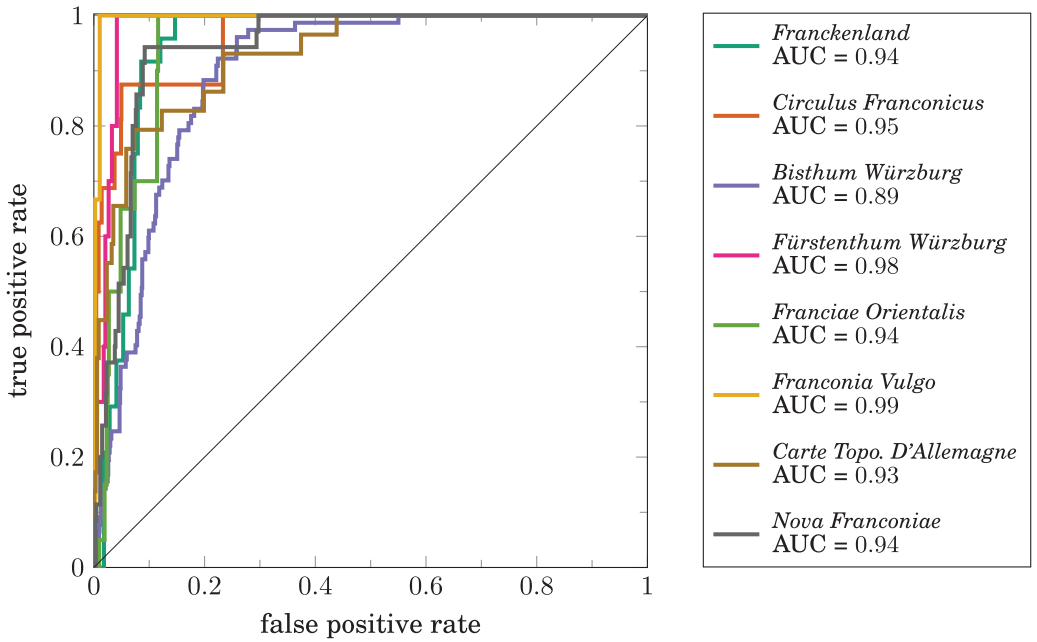


Fig. 15. ROC curves for classifying matches by sensitivity.

Since a typical map contains hundreds to thousands of map elements, we do not want to show everything to the user for verification. We perform a *sensitivity analysis* of each match (p, ℓ) and develop a classifier to determine which matches warrant user inspection. Our classifier simply sorts the matches in order of sensitivity and presents the t most sensitive ones to the user for inspection, for some parameter t. This parameter (number of matches, rather than an objective-value cutoff) is a reasonable measure for the amount of user effort we wish to expend and fits well to the standard ROC-curve analysis that we will perform on this classifier.

We use the following sensitivity analysis. Starting with the optimal matching M computed by our algorithm, we calculate for each $(p, \ell) \in M$ how much more expensive we could make that arc without changing the optimum. Equivalently: How much worse does the objective value get if we were not allowed to use (p, ℓ)? If this difference is large, then we can have some confidence in this particular match: All matchings that do not contain this match are much worse. If, on the other hand, the difference is small, then there are alternative matchings that the algorithm would consider almost as good as M: If the input were just slightly perturbed, perhaps one of those other matchings would be considered best. Then we call the match (p, ℓ) *sensitive* and decide that it is best to get a judgment from the user. We want to classify the assigned pairs in M into "show to user" and "don't show to user." The latter should all be correctly assigned (so all mistakes get caught), and the former should all be wrong (so as to not waste the user's time). Recall that our classifier simply picks the t most sensitive matches to be shown; this value t is known as the *discrimination threshold*.

We have run this classifier on all maps from Section 4 (with imbalanced input). To evaluate its performance, we calculated the *Receiver Operating Characteristic* (ROC) curve using the ground truth data; see Figure 15. The concept of ROC curves is often used to evaluate the performance of classifiers by showing false- and true-positive rates while varying the discrimination threshold t. Following standard methods in ROC

analysis [Fawcett 2006], we calculated the *Area Under the Curve* (AUC). Generally, an AUC value between 0.8 and 0.9 can be considered excellent, whereas values over 0.9 are outstanding [Hosmer Jr. and Lemeshow 2004]. The AUC in our experiments lies between 0.89 for the *Bisthum Würzburg* map and 0.99 for the *Franconica Vulgo* map: The classifier successfully finds problematic areas.

In our implementation, we use CPLEX's *warm start* feature to speed up the computation of the sensitivities. This uses partial results from the computation of M when determining how the optimal matching changes when a certain match is disallowed.[14] This speeds up the process since the new matching is probably close to M. (Disallowing a single match is likely to have local effects only.) The runtime for calculating the sensitivities and running the classifier was 1.2 seconds for *Fürstenthum Würzburg* and 46.3 seconds for *Circulus Franconicus*; without a warm start, the latter takes almost an hour. For the other maps, runtimes were between these two values.

Next, we introduce errors to our test input by randomly removing map elements as introduced in Section 4.3. As an example, we conduct this experiment on the *Franckenland* and the *Circulus Franconicus* maps. Removing 10% of the markers and up to 30% of the labels, the AUC value of our classifier stays above 0.8 on both maps. Based on balanced input but with introduced positional noise, the AUC is above 0.8 with ρ up to 70 px. For an extreme value of $\rho = 150$ px, the AUC value is still above 0.7 for the *Franckenland* map and above 0.6 for the *Circulus Franconicus* map. These experiments show that our classifier is sufficiently robust against erroneous input data to be of practical value.

By running the sensitivity analysis, we obtain for each match $(p, \ell) \in M$ an alternative matching; that is, a matching M' that is optimal subject to $(p, \ell) \notin M'$. When presenting an unclear match (p, ℓ) to the user, we can immediately show this matching: The best alternative matching if (p, ℓ) is indeed incorrect. Figure 16 shows an example of how the sensitivity analysis could be presented to the user. The depicted map contains the unclear situation from Figure 13, with the sensitivity values color coded from red to green. These matches have indeed been identified by the classifier and are displayed as uncertain by our sensitivity analysis. The figure also shows how we could instantly preview the next-best matching to the user in case he or she considers rejecting a match (here, the match under the mouse pointer). In the depicted situation, the next-best matching would only differ on three edges (dashed blue lines). Also note that the values obtained from the sensitivity analysis can be stored and used in later steps as an "accuracy estimate" as requested by Chiang [2015].

6. MATCHING MARKERS WITH SETS OF LABEL FRAGMENTS

In the preceding section, we presented a way to improve the quality of matching results by including user feedback. Next, we discuss a different approach to improve matching results, one that is based on refining our optimization model. Based on the output of the label detection approaches by Höhn [2013] and some preliminary experiments with other approaches, we observe that parts of text that belong to a single label often will be detected as separate labels.

See the map in Figure 17 for an example from our own experiments. Some of these label fragments clearly belong together and should have been detected as one (e.g., ℓ_1, ℓ_2, and ℓ_3): This is a mistake by the algorithm. On the other hand, it actually happens on historical maps that multiple fragments of text together form a single logical label.

[14]Using a previously computed solution to quickly solve similar instances is a common technique when using the simplex algorithm. This is widely supported by linear programming software.

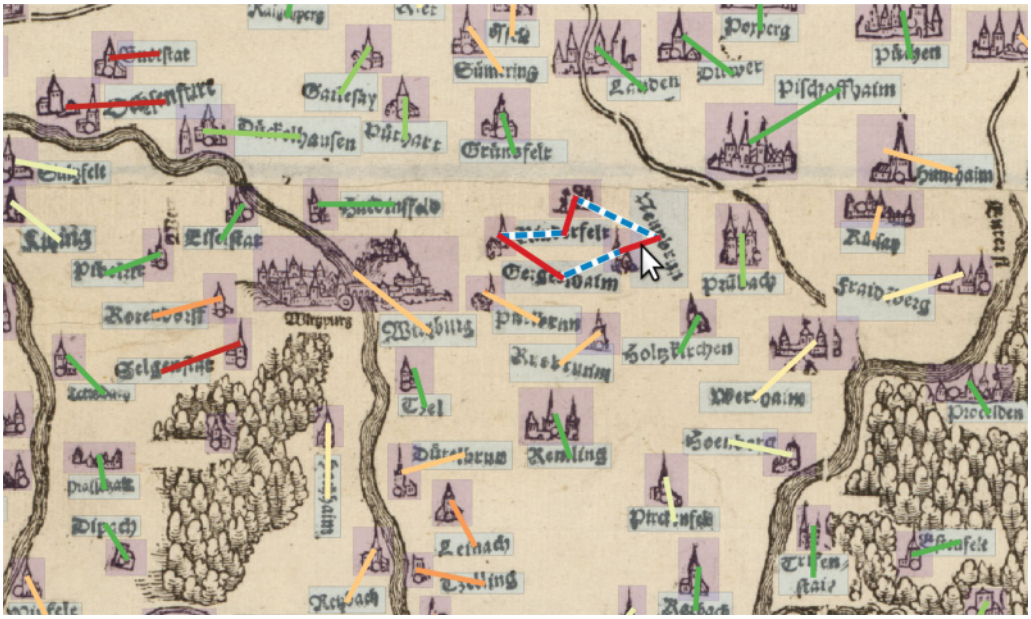
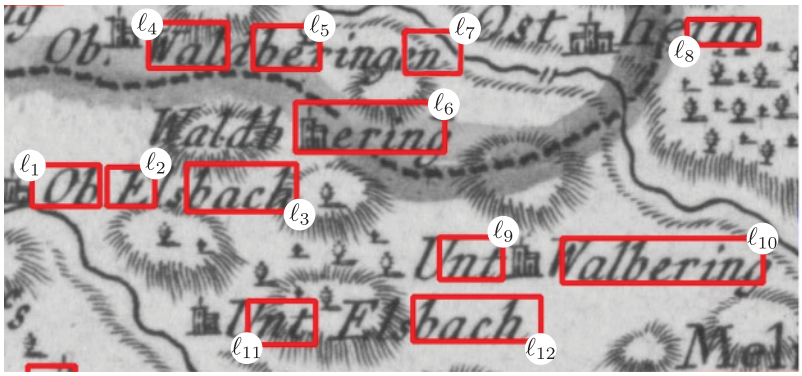


Fig. 16. Color-coded visualization of our system's confidence in each match. An alternative is presented for the match under the mouse pointer. Note that the unclear situation from Figure 13 has been identified and highlighted in red.



Family $\mathcal{F}$ contains

$\{\ell_1, \ell_2, \ell_3\}$
$\{\ell_4, \ell_5, \ell_7\}$
$\{\ell_6\}$
$\{\ell_8\}$
$\{\ell_{11}, \ell_{12}\}$
$\{\ell_9, \ell_{10}\}$
$\{\ell_9, \ell_{12}\}$

Fig. 17. Label fragments detected with a prototype label-detection algorithm. Note that many labels have (incorrectly) been detected as several separate fragments. On the right, we give an example family of sets that indicates which fragments might belong together. Note that it includes both $\{\ell_9, \ell_{12}\}$ and $\{\ell_{11}, \ell_{12}\}$.

Consider the fragments ℓ_9 and ℓ_{10}. They are geometrically separated by a place marker, but *together* form its label "*Unt. Walbering*." Figure 1 has many such split labels, even breaking them vertically into multiple lines, such as "*Hai-*" and "*delfelt*" in the middle left. Still, separately detected label fragments that form one logical label are located relatively close to each other. We use this property to improve our matching results and adapt the optimization problem introduced in Section 3.

In order to identify detected label fragments that potentially form a single label in the map, we propose using a heuristic that constructs a family of sets containing fragments that possibly belong together. For instance, an appropriate heuristic would be to put label fragments that are located within a certain distance from each other

into one set. On maps that contain mostly horizontal text, one might want to restrict the elements of a set to fragments that are aligned horizontally. The list on the right of Figure 17 shows an example of a family $\mathcal{F}$ of sets that a heuristic could return in the given map situation. Note that we allow fragments to be contained in more than one set to deal with unclear situations, such as, for example, the constellation of ℓ_9, ℓ_{10}, and ℓ_{12} in the lower right. Label fragments that do not have other fragments in their immediate vicinity could as well be the only element in a set, such as ℓ_8.

6.1. Optimization Problem

Based on a family of sets $\mathcal{F}$, we assume that all elements ℓ in a set $S \in \mathcal{F}$ could, together, plausibly form a single label. If they do, this entire set should be matched to a single place marker. Recall the three goals introduced in Section 3:

(C1) The size of the matching M is large.
(C2) The sum over $d(p, \ell)$ for all $(p, \ell) \in M$ is small.
(C3) No match $(p, \ell) \in M$ has distance $d(p, \ell) > r$.

Since we assume that the label fragments within a set S actually form a single label in the map, we do not want to split S by matching more than one fragment in S to a place marker. We add a fourth goal which takes this into account:

(C4) At most one label fragment ℓ from each $S \in \mathcal{F}$ should be matched in M.

This is not a hard constraint, but instead something we try to avoid through optimization. We combine these four goals into a new objective function by introducing a penalty term for any sets that get split:

$$g_{\text{obj}}(M) = \sum_{(p,\ell)\in M} (r - d(p, \ell)) - \sum_{S\in\mathcal{F}} c(S, M), \tag{1}$$

where $c : \mathcal{F} \times 2^L \to \mathbb{R}_{\geq 0}$ is the function that penalizes each match beyond the first from each set S. We define

$$c(S, M) = \begin{cases} 0 & \text{if } S \cap M = \emptyset, \\ \varphi \cdot (|S \cap M| - 1) & \text{otherwise} \end{cases} \tag{2}$$

and want to maximize g_{obj} under the constraint that M is a matching. By choosing a positive value for the penalty φ, each match $(p, \ell) \in M$ with $\ell \in S$ lowers the matching value if M contains at least one other match (p', ℓ') with $\ell' \in S$. We call minimizing this new objective g_{obj} the Fragment Assignment problem.

Solving the Fragment Assignment problem does not immediately give a matching between *all* label fragments and place markers. See Figure 18 for three example situations: In situation (a), we have the optimal matching $\{(p_1, \ell_1), (p_2, \ell_3)\}$. Since this matching splits S_1, it is uncertain which of the two place markers the remaining fragment ℓ_2 belongs to. In situation (b), the optimal matching is $\{(p_1, \ell_1), (p_2, \ell_4), (p_3, \ell_8)\}$. This only assigns the label fragments ℓ_1, ℓ_4, and ℓ_8. Based purely on the sets in $\mathcal{F}$, it is ambiguous whether ℓ_6 is part of the label for p_1 or p_2 (and whether ℓ_7 belongs to p_2 or p_3). Situation (c) shows that our optimization goal actually makes sense: p_2 is closer to ℓ_2, but is assigned to ℓ_3 to avoid (the costs of) splitting S_1.

Although we do not necessarily find a matching of all fragments, we still obtain useful information: a set of *reasonable* assignment options for each fragment. This could, for example, be used in a user interaction or as the input of a postprocessing heuristic. Considering situation (b) again, fragment ℓ_6 could be presented to the user with the suggestion that it belongs to either p_2 or p_1. For fragment ℓ_3, the only reasonable options are matching it to p_1 or leaving it unmatched.

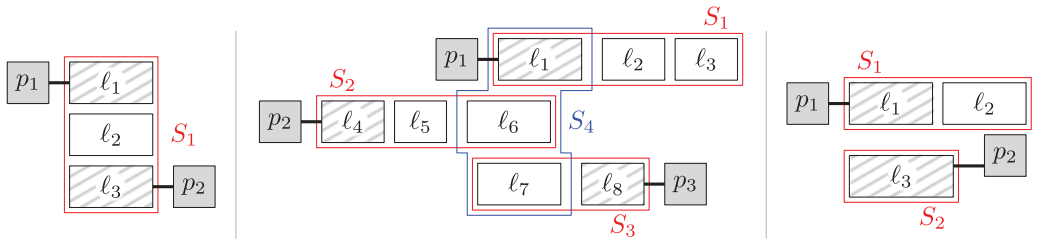


Fig. 18. Three different situations and the corresponding solutions to the Fragment Assignment problem. In situation (a), the assignment of label fragment ℓ_2 remains ambiguous; likewise in (b), the assignment of ℓ_6 and ℓ_7 is ambiguous (consider S_4). The solution in situation (c) yields a unique assignment and avoids splitting S_1.

6.2. Integer Linear Program

The preceding optimization problem is NP-complete (see Section 6.3). Here, we show that optimal solutions can be computed using an Integer Linear Program (ILP) formulation. The ILP has practical relevance because it can be used to test small instances and to gain deeper insight into the formulated optimization problem.

Let $x_{p,\ell} \in \{0, 1\}$ be a decision variable that indicates whether (p, ℓ) is taken into the matching or not; that is, we let $M = \{(p, \ell) \mid x_{p,\ell} = 1\}$. Additionally, for each $S \in \mathcal{F}$, we introduce an auxiliary variable $y_S \in \mathbb{Z}_{\geq 0}$ to track if multiple elements of S are part of the matching M: We use this to apply the proper splitting penalties to the objective function. For notational convenience, let $w(p, \ell) = r - d(p, \ell)$. Note that $w(p, \ell)$ and φ are constants. We can now use the objective function from Equation (1) to derive the following integer linear program:

$$\text{Maximize} \quad \sum_{p \in P} \sum_{\ell \in L} x_{p,\ell} \cdot w(p, \ell) - \sum_{S \in \mathcal{F}} y_S \cdot \varphi \tag{3}$$

$$\text{Subject to} \quad \sum_{p \in P} x_{p,\ell} \leq 1 \qquad \forall \ell \in L \tag{4}$$

$$\sum_{\ell \in L} x_{p,\ell} \leq 1 \qquad \forall p \in P \tag{5}$$

$$\sum_{\ell \in S} \sum_{p \in P} x_{p,\ell} \leq 1 + y_S \qquad \forall S \in \mathcal{F} \tag{6}$$

$$x_{p,\ell} \in \{0, 1\} \qquad \forall p \in P,\ \ell \in L \tag{7}$$

$$y_S \in \mathbb{Z}_{\geq 0} \qquad \forall S \in \mathcal{F} \tag{8}$$

Constraints (4) and (5) ensure that each marker and each label fragment is assigned at most once. Together, this guarantees that M is a matching. Constraint (6) forces y_S to be at least the number of matches within S minus one, thus applying the penalty to the objective value: In particular, if S is not split, then y_S can be set to 0 in order to get no penalty.

6.3. NP-Hardness

We now show that optimizing g_{obj} is NP-hard. Our proof uses a polynomial-time reduction from the NP-complete Set Packing problem to a decision variant of the Fragment Assignment problem; the following definition is taken from Karp [1972]:

	Set Packing
Instance:	A family of sets $\{S_j\}$.
	A positive integer k.
Question:	Does $\{S_j\}$ contain k mutually disjoint sets?

We take the following natural decision variant of our problem to reduce to:

	Fragment Assignment
Instance:	A set P of place markers.
	A set L of label fragments with $\lvert L\rvert \geq \lvert P\rvert$.
	A weight function $w(p, \ell)$: $P \times L \to \mathbb{R}_{\geq 0}$.
	A family $\mathcal{F} \subseteq 2^L$.
	A cost function $c(S, M)$: $\mathcal{F} \times 2^L \to \mathbb{R}_{\geq 0}$.
	A threshold τ.
Question:	Does there exist a matching M of place markers and label fragments such that $g_{\text{obj}}(M) \geq \tau$?

Theorem 6.1. *The* Fragment Assignment *problem is NP-complete.*

Proof. First, we note that Fragment Assignment is clearly in NP. We show hardness by reducing the classic Set Packing problem to Fragment Assignment.

Reduction. We construct the following Fragment Assignment instance in polynomial time. Let the set of label fragments L be $\bigcup_j S_j$. Let $\mathcal{F}$ be the family of sets $\{S_j\}$. We introduce k place markers (which form the set P) and let $w(p, \ell) = 1$ for all $p \in P$ and $\ell \in L$. Let $\tau = k$. For $c(S, M)$, we use the function defined in Equation (2) with $\varphi = 1$.

Equivalence. Assume $\{S_j\}$ contains k mutually disjoint sets. Then there are also k mutually disjoint sets of label fragments in $\mathcal{F}$. Matching each of the k place markers to an arbitrary label fragment from a different disjoint set in $\mathcal{F}$ yields a solution M for Fragment Assignment with $g_{\text{obj}}(M) = k$: Since at most one element from each set in $\mathcal{F}$ was matched, no penalties were applied.

For the other direction, suppose there is a solution M to the Fragment Assignment instance such that $g_{\text{obj}}(M) \geq \tau = k = \lvert P\rvert$. Then every marker is matched and no sets are split: $\{S_j\}$ contains k mutually disjoint sets. □

6.4. Polynomial-Time Algorithm for a Restricted Problem

Since solving Fragment Assignment on large instances is not feasible, we focus on a restricted version of the problem. Recall that in the general version of the problem as stated earlier, we allow label fragments to be elements of more than one set. If we instead require that each fragment is contained in only one set – that is, that $\mathcal{F}$ is a partition of L – we can give a polynomial-time algorithm for this restricted problem, which we call Fragment Assignment (disjoint sets).

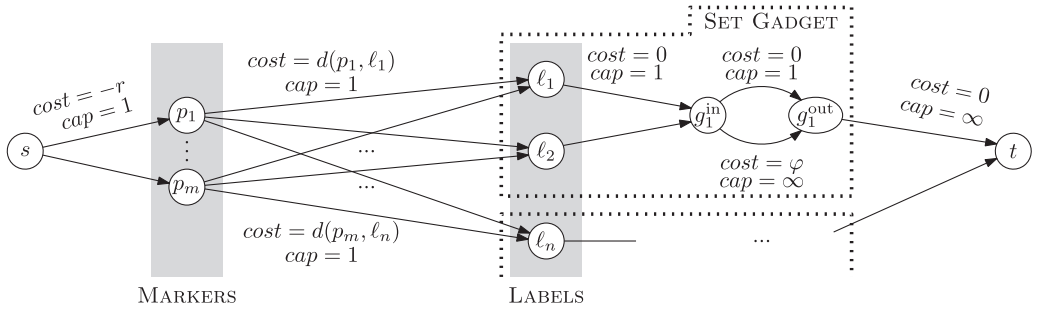


Fig. 19. The flow network from Figure 4, augmented with *set gadgets* (dashed boxes).

	Fragment Assignment (disjoint sets)
Instance:	A set P of place markers.
	A set L of label fragments with $\lvert L\rvert \geq \lvert P\rvert$.
	A weight function $w(p, \ell)$: $P \times L \to \mathbb{R}_{\geq 0}$.
	A partition $\mathcal{F}$ of L.
	A cost function $c(S, M)$: $\mathcal{F} \times 2^L \to \mathbb{R}_{\geq 0}$.
Objective:	Find a matching M of place markers and label fragments such that $g_{\text{obj}}(M)$ is maximized.

Requiring a partitioning of the label fragments is reasonable in many situations. As an example, in the situation in Figure 17, we can easily find a partition of detected fragments that are likely to belong together. This is especially the case under the assumption that detected text areas belong together horizontally; for example, if a label has been split by another map element (like a river or a place marker). We can meet this new requirement by applying a different heuristic to the label fragments of the input, which clusters them into disjoint sets. This can, for example, be done by applying standard clustering algorithms such as DBSCAN [Ester et al. 1996].

We can solve the Fragment Assignment (disjoint sets) problem efficiently by augmenting the flow network introduced for solving the Assign Labels problem (Section 3). For every set $S \in \mathcal{F}$, we introduce a *set gadget* to the flow network; Figure 19 shows its structure. The construction guarantees that one unit of flow can pass the gadget without increasing costs, whereas each additional unit of flow increases total costs by φ. The number of flow units that enter each gadget is equal to the number of label fragments in S that were matched. This corresponds directly to the behavior of the cost function $c(S, M)$ defined in Equation (2). The augmented flow network thus correctly models the Fragment Assignment (disjoint sets) problem.

The gadget for set i consists of two additional vertices, g_i^{in} and g_i^{out}. The entrance vertex g_i^{in} can be reached from all label fragments that belong to the set corresponding to the gadget by directed edges E_{in}. For every edge $e \in E_{\text{in}}$, we set $cost(e) = 0$ and $cap(e) = 1$. This capacity constraint guarantees that every label fragment is matched at most once. From g_i^{in} to g_i^{out}, there are two directed edges e_{free} and e_{penalty}, where $cost(e_{\text{free}}) = 0$ and $cap(e_{\text{free}}) = 1$, whereas $cost(e_{\text{penalty}}) = \varphi$ and $cap(e_{\text{penalty}}) = \infty$. The exit vertex g_i^{out} is connected to sink t with a directed edges e_{out} of $cost(e_{\text{out}}) = 0$ and $cap(e_{\text{out}}) = \infty$.

As in Section 3, we calculate a minimum cost flow in the constructed network. This can be done in polynomial time using standard methods, thereby solving Fragment

ASSIGNMENT (DISJOINT SETS). This shows that our extended optimization model, although NP-hard in general, can be solved efficiently in a realistic case.

7. CONCLUSION AND OUTLOOK

In this article, we considered the problem of determining the correct matching between labels and markers in historical maps. We assume that the bounding boxes of these map elements are given. We developed several optimization models based on such input and gave either efficient algorithms or hardness proofs for each.

We experimentally demonstrated that the algorithm for our main model has low error measure when run on accurate input (i.e., a manually extracted ground truth): It has an error measure in the low single digits. Additionally, we show that it copes well with a reasonable amount of noise. We did this by realistically perturbing the ground truth and evaluating how this influences the output of the algorithm.

In order to address the high quality demands in the digitization of historical documents, we proposed several improvements. First, we presented a system that allows interactive postprocessing of the matches calculated by our algorithm. The system calculates a measure of confidence in the matches and presents unclear situations to a user for verification. This selection procedure performs well in identifying parts of the map that need careful human attention or even research. In addition, we explored ways to extend our problem definition to deal with fragments of labels. While this problem is NP-hard in general, we gave an efficient algorithm for a reasonable restricted version.

Here, we have only previewed an early prototype of a user interface for our system. In future work, we intend to do a proper interaction design: The current user interface is very primitive from a human-computer interaction point of view, but a proper design was beyond the scope of the current work. In the context of interaction, our contribution has instead been the automatic detection of parts of the map that *need* interaction.

For a better user interface, we consider displaying a modern-day map next to the historical map for reference. Adding geographic context might help the user to deal with ambiguity. On the algorithmic front, we will engage in finding a method to quickly recompute sensitivity values once the matching changes due to user feedback. This is not trivial (recall that it takes more than a minute to perform our sensitivity analysis on the *Circulus Franconicus* map from scratch) but is crucial for a real-time interactive postprocessing system. Since matching markers with label fragments is hard, the task of linking label fragments should possibly be addressed independently in a preprocessing step.

Another direction for future work is designing a user interface for the set disambiguation problem of Figure 18. Once a user provides information on the correct assignment of an ambiguous label fragment, this information can be propagated, possibly enabling the automatic assignment of other fragments. This should have propagation properties similar to the matching sensitivity.

ACKNOWLEDGMENTS

We thank Dr. H.-G. Schmidt of the Würzburg University Library for fruitful discussion and for providing the scans used in the experiments. We thank Julian Behringer for creating the ground truth data for our experiments. Furthermore we thank M. Chlechowitz, J. Kauer, A. Löffler, and F. Wisheckel for their work on the preliminary tests shown in Figure 17. We thank the anonymous reviewers for useful remarks and questions.

REFERENCES

Mauricio Giraldo Arteaga. 2013. Historical map polygon and feature extractor. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on MapInteraction (MapInteract'13)*. ACM, 66–71.

Benedikt Budig and Thomas C. van Dijk. 2015. Active learning for classifying template matches in historical maps. In *Proceedings of the 18th International Conference on Discovery Science (DS'15) (Lecture Notes in Computer Science)*, Vol. 9356. Springer, 33–47.

Yao-Yi Chiang. 2015. Querying historical maps as a unified, structured, and linked spatiotemporal source. In *Proceedings of the 23rd ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (ACM SIGSPATIAL'15)*. ACM, 270–273.

Yao-Yi Chiang and Craig A. Knoblock. 2015. Recognizing text in raster maps. *GeoInformatica* 19, 1 (2015), 1–27.

Yao-Yi Chiang, Stefan Leyk, and Craig A. Knoblock. 2014. A survey of digital map processing techniques. *ACM Computing Surveys (CSUR)* 47, 1, Article 1 (2014), 44 pages.

Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. 2009. *Introduction to Algorithms* (3rd ed.). MIT Press.

Boris Epshtein, Eyal Ofek, and Yonatan Wexler. 2010. Detecting text in natural scenes with stroke width transform. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR'10)*. IEEE, 2963–2970.

Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. 1996. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the 2nd International Conference on Knowledge Discovery and Data Mining (KDD-96)*. The AAAI Press, Menlo Park, California, 226–231.

Tom Fawcett. 2006. An introduction to ROC analysis. *Pattern Recognition Letters* 27, 8 (2006), 861–874.

Christopher Fleet, Kimberly C. Kowal, and Petr Pridal. 2012. Georeferencer: Crowdsourced georeferencing for map library collections. *D-Lib Magazine* 18, 11/12 (2012).

Andrew V. Goldberg. 1997. An efficient implementation of a scaling minimum-cost flow algorithm. *Journal of Algorithms* 22 (1997), 1–29.

Winfried Höhn. 2013. Detecting arbitrarily oriented text labels in early maps. In *Proceedings of the 6th Iberian Conference on Pattern Recognition and Image Analysis (LNCS)*, Vol. 7887. Springer, 424–432.

Winfried Höhn, Hans-Günter Schmidt, and Hendrik Schöneberg. 2013. Semiautomatic recognition and georeferencing of places in early maps. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'13)*. ACM, 335–338.

David W. Hosmer Jr. and Stanley Lemeshow. 2004. *Applied Logistic Regression*. John Wiley & Sons.

Paul Jaccard. 1912. The distribution of the flora in the alpine zone. *New Phytologist* 11, 2 (1912), 37–50.

Bernhard Jenny and Lorenz Hurni. 2011. Cultural heritage: Studying cartographic heritage: Analysis and visualization of geometric distortions. *Computers & Graphics* 35, 2 (2011), 402–411.

Richard M. Karp. 1972. Reducibility among combinatorial problems. In *Complexity of Computer Computations*. Springer, 85–103.

Stefan Leyk, Ruedi Boesch, and Robert Weibel. 2006. Saliency and semantic processing: Extracting forest cover from historical topographic maps. *Pattern Recognition* 39, 5 (2006), 953–968.

Carlos A. B. Mello, Diogo C. Costa, and T. J. dos Santos. 2012. Automatic image segmentation of old topographic maps and floor plans. In *Proceedings of the IEEE International Conference on Systems, Man, and Cybernetics (SMC'12)*. IEEE, 132–137.

Reza Farrahi Moghaddam and Mohamed Cheriet. 2009. Application of multi-level classifiers and clustering for automatic word spotting in historical document images. In *Proceedings of the 10th International Conference on Document Analysis and Recognition (ICDAR'09)*. IEEE, 511–515.

Toni M. Rath and Raghavan Manmatha. 2003. Word image matching using dynamic time warping. In *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'03)*, Vol. 2. IEEE, II–521–II–527.

Toni M. Rath and Raghavan Manmatha. 2007. Word spotting for historical documents. *International Journal of Document Analysis and Recognition (IJDAR'07)* 9, 2–4 (2007), 139–152.

Hendrik Schöneberg, Hans-Günter Schmidt, and Winfried Höhn. 2013. A scalable, distributed and dynamic workflow system for digitization processes. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries (JCDL'13)*. ACM New York, NY, USA, 359–362.

Alexander Schrijver. 2003. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer.

Tenzing Shaw and Peter Bajcsy. 2011. Automation of digital historical map analyses. In *Proceedings of the IS&T/SPIE Electronic Imaging 2011*, Vol. 7869. SPIE.

Rainer Simon, Elton Barker, Leif Isaksen, and Pau de Soto Cañamares. 2015. Linking early geospatial documents, one place at a time: Annotation of geographic documents with recogito. *e-Perimetron* 10, 2 (2015), 49–59.

Rainer Simon, Bernhard Haslhofer, Werner Robitza, and Elaheh Momeni. 2011. Semantically augmented annotations in digitized map collections. In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries (JCDL'11)*. ACM New York, NY, USA, 199–202.

Rainer Simon, Peter Pilgerstorfer, Leif Isaksen, and Elton Barker. 2014. Towards semi-automatic annotation of toponyms on old maps. *e-Perimetron* 9, 3 (2014), 105–112.

Kai Wang, Boris Babenko, and Serge Belongie. 2011. End-to-end scene text recognition. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV'11)*. IEEE, 1457–1464.

Jerod Weinman. 2013. Toponym recognition in historical maps by gazetteer alignment. In *Proceedings of the 12th International Conference on Document Analysis and Recognition (ICDAR'13)*. IEEE, 1044–1048.

Received December 2015; revised June 2016; accepted September 2016