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

Benedikt Budig
Universität Würzburg
info@benedikt-budig.de

Thomas C. van Dijk*
Chair for Computer Science I
Universität Würzburg
thomas.van.dijk@uni-wuerzburg.de

Alexander Wolff
Chair for Computer Science I
Universität Würzburg
alexander.wolff@uni-wuerzburg.de

ABSTRACT

In this paper 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 want to match the labels to the markers. This time-consuming step in the digitization process of historical maps is non-trivial even for humans, but provides valuable metadata (for example when subsequently georeferencing the map). In order 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 results.

We test the algorithm on a manually-extracted ground truth for two historical maps with a combined total of over 4000 markers and labels. The algorithm correctly matches 99% 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 assignments. We use this as the basis of an interactive system, where the user's effort is directed to checking the parts of the map where the algorithm is unsure; any corrections the user makes can be propagated by the algorithm. We discuss an early prototype of this system and statistically confirm that it successfully locates the areas on the map where the algorithm needs help.

Categories and Subject Descriptors

F.2.2 [**Analysis of algorithms and problem complexity**]: Nonnumerical Algorithms and Problems—*Computations on discrete structures, Geometrical problems and computations*; H.2.8 [**Information Systems**]: Database Applications—*Spatial databases and GIS*

General Terms

Algorithms, Experimentation

Keywords

Historical Maps, Algorithms, Interaction, Sensitivity analysis, Metadata extraction

*Corresponding author.

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. In order to make maps more easily retrievable, metadata describing the contained information need 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 upwards of 25 hours to georeference the 1000 to 4000 places contained in a typical map from their collection.[1]

This information extraction task 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; see Figure 1 for some examples. 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 historical map collections, however, the correctness of metadata is of paramount importance.

In this paper 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 of a certain marker. The question is then: which label belongs to which marker? This is in fact a non-trivial problem, even for humans. (Look ahead at Figure 7 for a tricky situation that we will discuss in more detail later on.)

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

Figure 1: Place markers and place labels on several historical maps. Note the variety of visual styles, both in the pictographs and the lettering.

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 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 [14] and Schöneberg et al. [12]. The latter propose a pipeline with separate tasks operating independently; our algorithm could serve as a module in their system.

After introducing our algorithm for matching labels and markers (Section 3), we present several experiments that show that the algorithm performs well on the two historical maps we have tested on (Section 4). Finally we present 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).

2. RELATED WORK

Since the digitization and analysis of historical maps is of increasing interest to libraries and collections, 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 (for example Fleet et al.'s *Georeferencer* [4] and the system by Simon et al. [15]). For the postprocessing of georeferenced maps, Jenny and Hurni [9] introduced a tool that is able to analyze the geometric and geodetic accuracy of historical maps and then visualize the identified distortions.

Some research has gone into the segmentation of bitmap images of (historical) maps. Höhn [6] introduced a method to detect arbitrarily rotated labels in historical maps; Mello et al. [11] dealt with the similar topic of identifying text in historical maps and floor plans. Both methods could be used to generate input data for the algorithm presented in this paper. We should note, however, that 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—that is, developed specifically to digitize a particular corpus. For example, Leyk et al. [10] describe a method to find forest cover in a specific set of 19^{th} century topographic maps. Arteaga [1] 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 19^{th} and early 20^{th} centuries. The effectiveness of these approaches is in part due to the homogeneity of these relatively recent maps. The tests in this paper are performed on much older maps (16^{th} and early-18^{th} century). For detecting and georeferencing places in such early maps, Höhn et al. [7] 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 paper we take a different approach to identify contained places: finding the corresponding text labels in the map.

3. MATCHING MARKERS AND LABELS

In our algorithm, both markers and labels are represented by an axis-aligned bounding rectangle. This is a reasonable simplification on many maps, but could easily be generalized if needed. 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 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 $l \in L$, and every $l \in L$ is assigned to at most one $p \in P$. However, we do not assume that there is a one-to-one correspondence: indeed, both maps we have tested contain unlabeled markers and 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. For a marker $p \in P$ and a label $l \in L$, we define the distance $d(p, l)$ as the Euclidean distance between the rectangles (that is, the smallest distance between a point in p and a point in l). 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. This parameter r has to be chosen somewhat carefully because an insufficiently large value might disallow the correct assignment. In practice, a suitable value is easily found; see Section 4.3 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.[2] Our goal is now to find a matching M that: (C1) matches many labels, (C2) generally matches labels to nearby markers, and (C3) has no match that is very far away.

A first attempt might be to use a greedy algorithm that simply matches the closest label-marker pair and repeats, but this does not work well at all (see Section 4). A natural way to combine these criteria into a proper optimization objective is as follows: any pair $(p, l) \in M$ gives us some fixed benefit r (criterion C1) but also has a cost $d(p, l)$ (criterion C2):

$$\textbf{Maximize } f(M) \;=\; \sum_{(p,l)\in M} r - d(p, l)$$

We define the Label Assignment problem as maximizing

[2]This is the graph-theoretical concept of a matching in the complete bipartite graph between P and L.

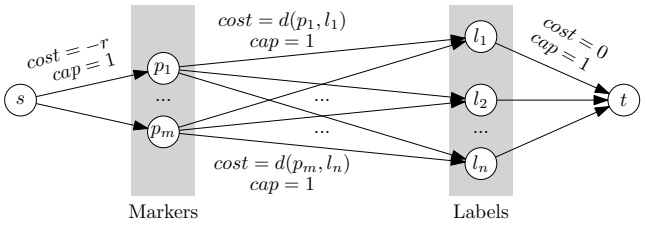

Figure 2: Flow network G.

f subject to M being a matching. Note that criterion (C3) will always hold in an optimal solution since any pair (p, l) with $d(p, l) > r$ 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 rise in order to match an additional label.

We solve the Label Assignment problem using the flow-based approach illustrated in Figure 2. 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 [2]. In order to apply the standard concept of *minimum* cost flows, we translate our maximization problem into a minimization problem and define arc weights accordingly. Arcs (s, p) leaving s have $cost(s, p) = -r$: this corresponds to a benefit of r for establishing a match. For each arc (p, l) we set $cost(p, l) = d(p, l)$: the distance-based cost. Each remaining arc (l, t) has $cost(l, t) = 0$. Then a flow in G corresponds precisely to a solution M of Label Assignment, where the flow cost equals $-f(M)$.

Finding a flow of minimum cost over all admissible flows in G gives an optimal solution for the model introduced above: 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. [5]). Since our instances have integer capacities, we can also use fractional linear programming to find an optimal solution: this is correct because the fractional flow LP is integer when the capacities are integer [13]. This allows us to use standard LP algorithms (e.g. simplex) to efficiently solve the problem. We have used this approach in our implementation.

4. EXPERIMENTS

The experiments presented in this section have been run on a laptop PC with an Intel® Core™ i5-3427U CPU at 1.80 GHz; memory was no issue. We have used CPLEX v12.5.1 for solving linear programs.

We have 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 the 19th century, many featuring several thousand place markers. For this paper we have manually extracted all markers and labels contained in two historical maps from this collection, the *Franckenland* map[4] from 1533 and the *Circulus Franconicus* map[5] from 1706. For both these maps, we have matched markers and labels by hand and use this as a ground truth for our experiments. Unless otherwise noted, we have used a fixed value of $r = 150$ px on both maps. (We discuss this value in Section 4.3; for now, see Figure 6 for an indication of scale.)

[3] `http://franconica.uni-wuerzburg.de/`

[4] Sebastian von Rotenhan. *Das FranckenLandt = Chorographi Franciae Orien[talis]*, 1533.

	F1	F2	C1	C2
number of labels	517	524	1644	1669
incorrect matches	2	14	8	20
error rate	0.4%	3.5%	0.5%	1.3%
runtime	0.9 s	1.0 s	2.1 s	2.2 s
greedy error rate	17.8%	17.8%	5.4%	5.9%

Table 1: Statistics of our experimental results.

4.1 Balanced Case

First, we have 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 filtered a small amount of unlabeled markers and stray labels.

The filtered input data for the *Franckenland* map thus consists of 517 markers and labels. Our algorithm assigns 515 labels correctly and makes 2 incorrect assignments (experiment F1). This took 0.9 seconds of runtime. On the larger *Circulus Franconicus* data set the algorithm correctly assigns 1636 out of 1644 labels, with the remaining 8 labels matched to incorrect markers (experiment C1). Here the runtime was 2.1 seconds. Table 1 contains these and further statistics.

In both experiments, the error rate is below 1%, which we consider a good result given the dense and sometimes inconsistent placement of map elements. For comparison, we have also implemented the greedy algorithm discussed before, which iteratively adds an assignment with smallest distance to the matching. It has error rate 17.8% on the *Franckenland* map and 5.4% on the *Circulus Franconicus* map. We conclude that the greedy algorithm is unsuitable for this matching task, and that the results of our algorithm are indeed non-trivial. Note that the error rate of our algorithm is similar on both maps, but that the greedy algorithm performs significantly worse on the *Franckenland* map. This is likely due to the labeling style used in this map (it predates the other map by 170 years): 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. [6]). In the next set of tests, we take this into account by introducing *positional*

[5] 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, Wertheim, Holach, Reinec, Papenheim, Erpach, Schwartzenberg, et Castel, Baronatus Sensheim et Territorium Norinbergense*, 1706.

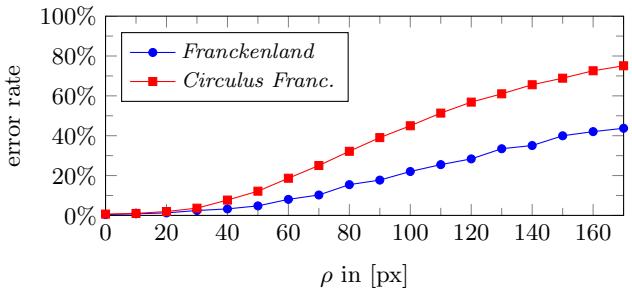

Figure 3: Effect of ρ on the error rate. For $\rho \leq 30$ px the graphs are nearly horizontal and below 4%.

noise. Based on the ground truth data, we have shifted all labels by some offset, each label independently, uniformly at random from $[-\rho, \rho] \times [-\rho, \rho]$, for some real parameter ρ. We have then run the algorithm repeatedly with different values of ρ. In Figure 3 we observe that our algorithm copes well with positional noise as long as the distances by which labels are shifted are not too high. In particular, for $\rho \leq 30$ px the error rate stays below 4% on both maps. This is the width of approximately one to three characters in an average label in these maps, which is a reasonable margin for imprecision of the input. The error rate increases faster for the *Circulus Franconicus* map because the placement of its map elements is denser than in the *Franckenland* map.

4.2 Imbalanced Case

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

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 F2). Taking into account that the algorithm also matched some map elements that should in truth have remained unmatched, we get an error rate of 3.5%, with a runtime of 1.0 s. Doing the same for the *Circulus Franconicus* map, we have an input of 1663 markers and 1669 labels, with 19 unlabeled markers and 25 stray labels. The error rate in this experiment (experiment C2) is 1.3%, with a runtime of 2.2 s. Note that the error rate on this map is considerably lower than on the *Franckenland* map. As stated in Section 4.1, the labeling in the that map has a more complicated structure, which is further complicated by including the stray map elements. Both error rates 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 error rates 17.8% and 5.9%, respectively.

The imbalanced case allows us to address imprecision in the input beyond positional noise: missing elements. This is particularly relevant considering that our algorithm is intended for use as a module in a longer pipeline: a preceding step might not have detected all map elements. We tested

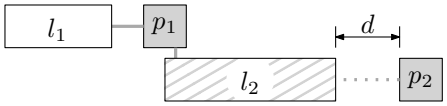

Figure 4: When using parameter $r < d$, the marker p_2 is too far away from label l_2 to be considered. This results in the matching $\{(p_1, l_2)\}$. With a higher value of r, the matching $\{(p_1, l_1), (p_2, l_2)\}$ is found, which is presumably correct.

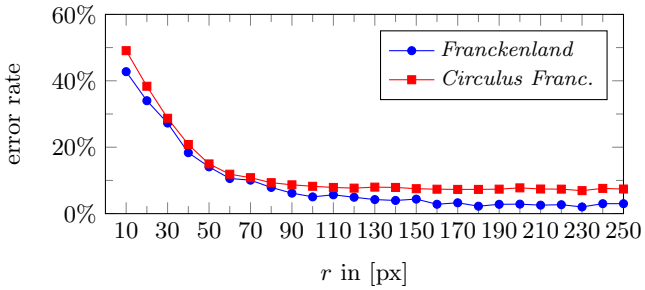

Figure 5: Effect of r on the error rate ($\rho = 40$ px)

this scenario on artificial instances, starting with the ground truth and removing each element from P with some probability and each element from L with some other probability. We want the algorithm to correctly identify the *resulting* situation, where some labels and markers have become unmatched. With higher deletion probability, the number of correct matches goes down of course, but the error rate in the remaining instance is almost uninfluenced (statistics have been omitted due to a lack of space): missing some elements does not significantly "confuse" the algorithm.

4.3 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 4). 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 balanced instances such that our algorithm has an error rate of 0% for some r and 100% for higher r. However, such instances are unlikely in practice; indeed, in our experiments on real maps, picking r exceedingly high is not detrimental to the quality of the result. An experiment on balanced input data with fixed positional noise $\rho = 40$ px shows that the error rate does not increase significantly for high values of r (Figure 5). In this experiment, even for $r = 1000$ px, the error rate stays at 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 C1 in 2.1 seconds; with $r = 1000$ px, this increases to 11.9 seconds, even though the algorithm finds the exact same matching.

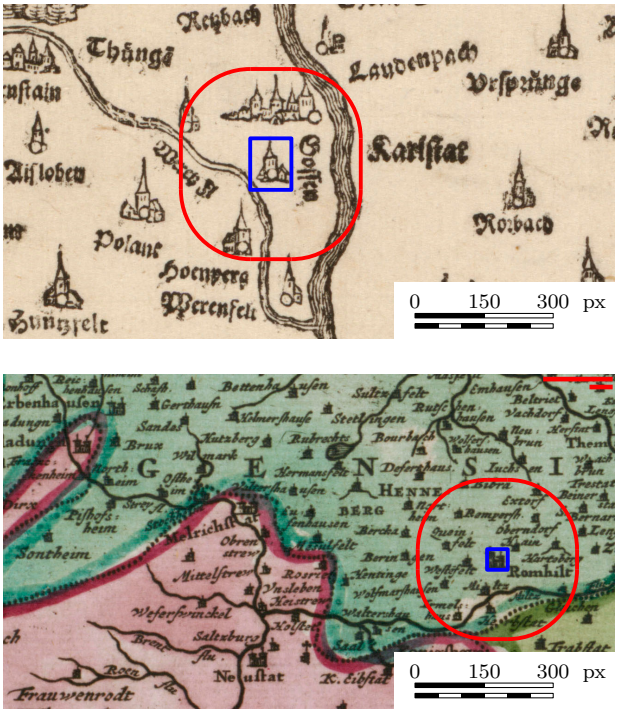

Figure 6: Value of r on *Franckenland* (left) and *Circulus Franconicus* (right). The red boundary marks a distance of 150 px from the blue bounding box.

For this reason, 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 6).

5. INTERACTIVE POSTPROCESSING

Both maps in our experiments have parts where it is unclear (even to a human reader without domain knowledge) how the markers and labels belong together. Changing a single assignment in such situations can propagate to other assignments. Figure 7 shows an example where three matchings seem reasonable: the correct assignment is unclear without additional topographic or historical information.

Figure 8 shows a screenshot of our prototype plug-in for QGIS.[6] The 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 assignment in Figure 8 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.

[6] `http://www.qgis.org/`

Figure 7: A difficult case: without geographic or historical context it is hard to tell which of these three assignments is correct.

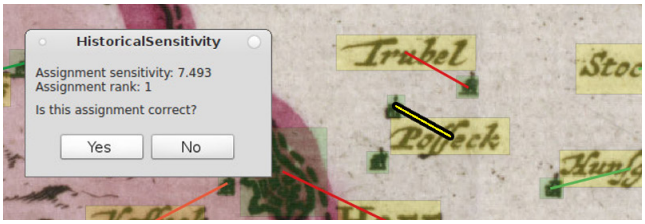

Figure 8: Screenshot of our prototype for interactive postprocessing.

Since a typical map contains hundreds to thousands of map elements, we do not want to show everything to the user for verification. We therefore develop a classifier to determine which assignments require user inspection, based on a *sensitivity analysis* of each assignment (p, l) in the matching. Our classifier simply sorts the assignments in order of sensitivity value and presents the t most sensitive assignments to the user for inspection, for some parameter t.

We use the following sensitivity analysis. Starting with the optimal matching M computed by our algorithm, we calculate for each $(p, l) \in M$ how much more expensive we could make that arc without changing the optimum—equivalently: how much worse does the objective value get if we weren't allowed to use (p, l)? If this difference is large, then we can have some confidence in this particular assignment: all matchings that do not contain this assignment 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 assignment (p, l) *sensitive* and it may be best to get a judgment call 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 assignments to be shown; this value t is known as the *discrimination threshold*.

We have run this classifier on both 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 9. 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 [3], we calculated the *area under the curve* (AUC). Generally an AUC value between 0.8 and 0.9 can be considered excellent, while values over 0.9 are outstanding [8]. The AUC in our experiments is 0.94 for the *Franckenland* map and 0.88 for the *Circulus Franconicus*

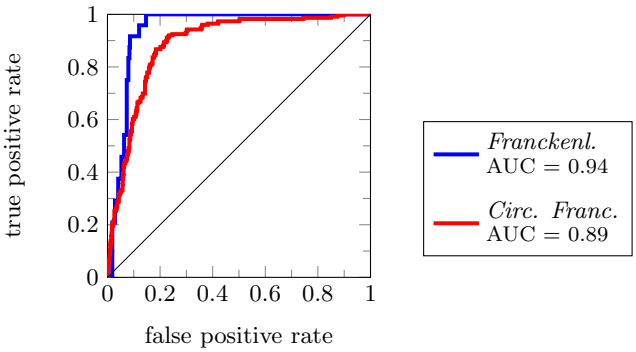

Figure 9: ROC curves for classifying assignments by sensitivity.

map: the classifier successfully finds trouble 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 assignment is disallowed.[7] The runtime for calculating the sensitivities and running the classifier was 3.7 seconds for *Franckenland* and 78.5 seconds for *Circulus Franconicus*; without a warm start the latter takes almost an hour.

Next, we introduce errors to our test input by dropping map elements as introduced in Section 4.2. Dropping 10% of the markers and up to 30% of the labels, the AUC value of our classifier stays above 0.8. 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 assignment $(p, l) \in M$ an alternative matching, that is, an assignment M' that is optimal subject to $(p, l) \notin M'$. When presenting an unclear assignment (p, l) to the user, we can immediately show this matching: the best alternative matching if (p, l) is indeed incorrect. Figure 10 shows an example of how the sensitivity analysis could be presented to the user. The depicted map contains the unclear situation from Figure 7, with the sensitivity values color coded from red to green. These assignments 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 an assignment (here the assignment under the mouse pointer). In the depicted situation, the next-best matching would only differ on three edges (dashed blue).

6. CONCLUSIONS AND FUTURE WORK

We have introduced an algorithm that assigns labels to the corresponding markers in historical maps, given the positions of these map elements. We have experimentally demonstrated that the algorithm has low error rate when run on perfect input (a manually-extracted ground truth) and also copes well with a reasonable amount of noise. To satisfy the high quality demands in the digitization of historical documents, we propose a system that allows interactive post-processing of the assignments calculated by our algorithm. The system calculates a measure of confidence in the assignments and presents unclear situations to a user for verification. This classifier performs well in identifying parts of the map that need careful human attention or even research.

In this paper, we have 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.

[7] 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.

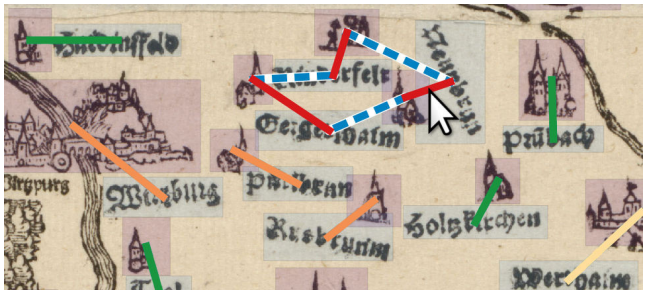

Figure 10: Color-coded visualization of our system's confidence in each assignment. An alternative is presented for the assignment under the mouse pointer.

Acknowledgments

We thank Dr. H.-G. Schmidt for fruitful discussion and for providing the scans used in the experiments.

7. REFERENCES

[1] M. G. Arteaga. Historical Map Polygon and Feature Extractor. In *Proceedings of the 1st ACM SIGSPATIAL International Workshop on MapInteraction*, MapInteract '13, pages 66–71, 2013.

[2] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms*. The MIT Press, 3rd edition, 2009.

[3] T. Fawcett. An introduction to ROC analysis. *Pattern recognition letters*, 27(8):861–874, 2006.

[4] C. Fleet, K. C. Kowal, and P. Pridal. Georeferencer: Crowdsourced Georeferencing for Map Library Collections. *D-Lib Magazine*, 18(11/12), 2012.

[5] A. V. Goldberg. An Efficient Implementation of a Scaling Minimum-Cost Flow Algorithm. *Journal of Algorithms*, 22:1–29, 1997.

[6] W. Höhn. Detecting Arbitrarily Oriented Text Labels in Early Maps. In *Proceedings of the 6th Iberian Conference on Pattern Recognition and Image Analysis*, volume 7887 of *LNCS*, pages 424–432. Springer, 2013.

[7] W. Höhn, H.-G. Schmidt, and H. Schöneberg. Semiautomatic Recognition and Georeferencing of

Places in Early Maps. In *Proceedings of the 13th ACM/IEEE-CS Joint Conference on Digital Libraries*, JCDL '13, pages 335–338, 2013.

[8] D. W. Hosmer Jr and S. Lemeshow. *Applied Logistic Regression*. John Wiley & Sons, 2004.

[9] B. Jenny and L. Hurni. Cultural Heritage: Studying Cartographic Heritage: Analysis and Visualization of Geometric Distortions. *Computers & Graphics*, 35(2):402–411, 2011.

[10] S. Leyk, R. Boesch, and R. Weibel. Saliency and semantic processing: Extracting forest cover from historical topographic maps. *Pattern Recognition*, 39(5):953–968, 2006.

[11] C. A. B. Mello, D. C. Costa, and T. J. dos Santos. Automatic Image Segmentation of Old Topographic Maps and Floor Plans. In *Proceedings of the 2012 IEEE International Conference on Systems, Man, and Cybernetics*, SMC '12, pages 132–137, 2012.

[12] H. Schöneberg, H.-G. Schmidt, and W. Höhn. 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, pages 359–362, 2013.

[13] A. Schrijver. *Combinatorial Optimization: Polyhedra and Efficiency*. Springer, 2003.

[14] T. Shaw and P. Bajcsy. Automation of Digital Historical Map Analyses. In *Proceedings of the IS&T/SPIE Electronic Imaging 2011*, volume 7869, 2011.

[15] R. Simon, B. Haslhofer, W. Robitza, and E. Momeni. Semantically Augmented Annotations in Digitized Map Collections. In *Proceedings of the 11th Annual International ACM/IEEE Joint Conference on Digital Libraries*, JCDL '11, pages 199–202, 2011.