Border Detection for Seamless Connection of Historical Cadastral Maps

Ladislav Lenc[1,2], Martin Prantl[1,2], Jiří Martínek[1,2], and Pavel Král[1,2]

[1] Dept. of Computer Science & Engineering
Faculty of Applied Sciences
University of West Bohemia
Plzeň, Czech Republic
[2] NTIS - New Technologies for the Information Society
Faculty of Applied Sciences
University of West Bohemia
Plzeň, Czech Republic
{llenc,perry,jimar,pkral}@kiv.zcu.cz

Abstract. This paper presents a set of methods for detection of important features in historical cadastral maps. The goal is to allow a seamless connection of the maps based on such features. The connection is very important so that the maps can be presented online and utilized easily. To the best of our knowledge, this is the first attempt to solve this task fully automatically. Compared to the manual annotation which is very time-consuming we can significantly reduce the costs and provide comparable or even better results.
We concentrate on the detection of cadastre borders and important points lying on them. Neighboring map sheets are connected according to the common border. However, the shape of the border may differ in some subtleties. The differences are caused by the fact that the maps are hand-drawn. We thus aim at detecting a representative set of corresponding points on both sheets that are used for transformation of the maps so that they can be neatly connected. Moreover, the border lines are important for masking the outside of the cadastre area.
The tasks are solved using a combination of fully convolutional networks and conservative computer vision techniques. The presented approaches are evaluated on a newly created dataset containing manually annotated ground-truths. The dataset is freely available for research purposes which is another contribution of this work.

Keywords: Historical Document Images · Cadastral Maps · Fully Convolutional Networks · FCN · Computer Vision

1 Introduction

In the recent years, there has been a growing interest in the field of historical document and map processing. Such materials already exist in digital form

(scanned digital images) and the efforts of current research lie in the automatic processing of such documents (e.g. information retrieval, OCR and full-text search but also computer vision tasks such as noise reduction, localization and segmentation of regions of interest and many others). All such tasks are very important for making the documents easily accessible and usable.

This paper deals with historical cadastral Austro-Hungarian maps from the 19th century. The maps are available as isolated map sheets covering particular cadastre areas. To be able to fully utilize such materials it is very important to create a continuous seamless map that can be easily presented online.

The map sheets are arranged in a rectangular grid [1]. Each cell of the grid can contain one or more map sheets (if the grid lies on the border of two or more cadastre areas). A preliminary step is to place the map sheets into corresponding cells according to textual content contained in the sheets. In this work, we concentrate on merging of sheets lying on the same position in the grid. The sheets have to be connected according to the common border line present in both sheets. However, the maps are hand-drawn and there might be tiny differences in the position and shape of corresponding border lines. We thus have to transform the sheets so that they can be neatly put together. Two map sheets belonging to the same cell in the grid are shown in Figure 1.

Our goal is twofold. First, we have to find a set of corresponding points in the two adjacent sheets that allow the transformation. The second goal is to find the border line which allows us to mask the regions outside the cadastre area.

The presented methods are developed as a part of a robust system for a seamless connection of map sheets which will require a minimum manual interaction of the user. This is the first attempt to solve this task fully automatically. We utilize machine learning approaches based on fully convolutional neural networks (FCN) complemented by analytic computer vision approaches. It is important to note that there is a potential of using the same system for processing of cadastre maps from neighboring countries that were also part of the Austro-Hungarian empire and have the same form (i.e. Slovakia, Hungary, Austria and parts of Ukraine, Serbia or Croatia).

An important contribution is the creation of an annotated dataset which is used for training of neural networks as well as for evaluation of the methods. This dataset is freely available for research purposes.

2 Related Work

To the best of our knowledge there are no studies utilizing deep learning for exactly the same task. We thus present relevant papers utilizing neural networks for map analysis and processing with a particular focus on segmentation methods based on neural networks.

A complete survey of historical cadastral maps digitization process is provided by Ignjatić et al. in [2]. The authors try to pave the way for employing deep neural networks in this field, highlighting potential challenges in map digitization.

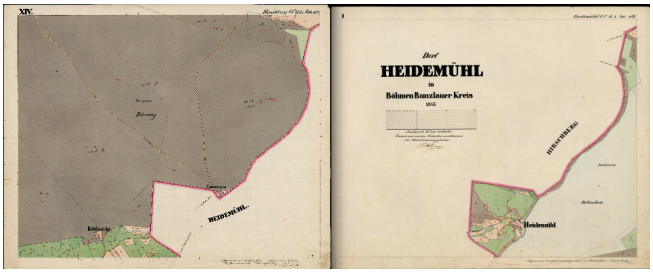


Fig. 1: Example of map sheets belonging to the same cell; The map areas are complements to each other; The outer area of one sheet is the inner area of the other one.

Convolutional neural networks are used for the analysis of aerial maps by Timilsina et al. in [3]. The objective of this work is to identify tree coverage of cadastre parcels.

Illegal building detection from satellite images is investigated by Ostankovich and Afanasyev in [4]. Their approach integrates various computer vision techniques with the *GoogLeNet* model and obtains reasonable and acceptable results that are verified against cadastral maps.

A method for cadastre border detection utilizing FCNs was presented by Xia et. al in [5]. They were able to effectively extract cadastral boundaries, especially when a large proportion of cadastral boundaries is visible. Another example of cadastre border extraction has been explored by Fetai et. al [6].

Nyandwi et. al [7] present detection and extraction method of visible cadastral boundaries from very high-resolution satellite images. They compared their approach with human analysis and they were able to obtain good results on rural parcels. On the other hand, the methods for the urban area extractions have significant room for improvement.

Kestur et. al [8] presented an approach for road detection from images acquired by sensors carried out by an unmanned aerial vehicle (UAV). They obtained promising results with a novel U-shaped FCN (UFCN) model that is able to take inputs of arbitrary size.

A combination of neural networks and mathematical morphology for historical map segmentation was proposed in [9]. The approach first utilizes CNN based network for edge detection. Follows morphological processing which helps to close the shapes and extract final object contours.

To sum up the related work, we can say that variants of CNN, especially the FCN architecture are suitable for our task of detection and segmentation of the important map features.

3 Historical Cadastral Maps

This section is intended to facilitate a basic understanding of the maps we work with and the terminology used in following sections.

The map sheets are arranged to a rectangular grid. The position (cell) of the sheet in the grid is described in so called "nomenclature". Each map sheet contains a map frame which defines the area of the cell that the sheet covers. The resolution of the scanned sheets is circa 8400×6850 pixels and the dimension of the map frame is usually around 7750×6200 pixels. The area inside the map frame can be divided into the map itself and the blank area outside the map. The map area is delimited by cadastre border which is marked as a black line with map symbols (dots, triangles, crosses etc.) determining the type of the border.

Our main concern is finding map sheets from neighboring cadastre areas that belong to the same cell. Such sheets are the most problematic ones for the creation of the seamless map. We focus on detection of landmarks ("Grenzsteine" from German) that are marked as black/red dots lying on the cadastre border line. These points are important because their positions are physically marked in the terrain and they are present on both map sheets covering a particular cell.

Another type of important points are significant direction changes on the border line. We will denote such points as "break points". These points are important mainly for map sheets containing no landmark positions. Both types of points can be used for transformation and connection of the map sheets.

The outside of the border line is usually accompanied by a colored (mostly red) area denoted as "edge-line". The border line, landmarks and edge-line are illustrated in Figure 2.

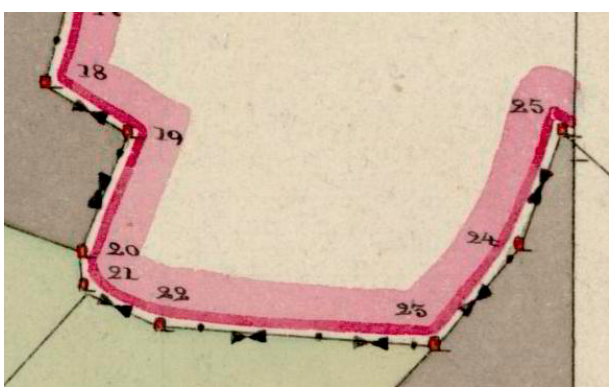


Fig. 2: Detail of a cadastre border with landmarks (red/black symbols) and red edge-line on the outside of the border

4 Proposed Detection Methods

The scheme of the overall map sheet processing pipeline is depicted in Figure 3. We assume that the input map sheet is already cropped according to the bounding box given by the map frame. Our task is to find a set of points that can be used for connection with the corresponding (complementing) map sheet. We utilize four separate algorithms, namely *Landmark Detector*, *Border Line Detector*, *Break Point Detector* and *Edge-line Detector*, to extract all necessary information. We first search for explicitly drawn landmarks and also identify the border line. If the amount of landmark points is low or we find no landmarks at all, the border line is further processed by the *Break Point Detector* that seeks significant direction changes in the border line and such points are used together with the landmark points to create a set of *Map Border Key-points* that are used for map transformation and connection. The *Edge-line Detector* is useful for removing false positive landmark points. Moreover, the relative edge-line position to the border line is crucial for determining the inner and outer area of the map. According to the border line and edge-line we can construct a map area mask which is subsequently utilized for map area masking.

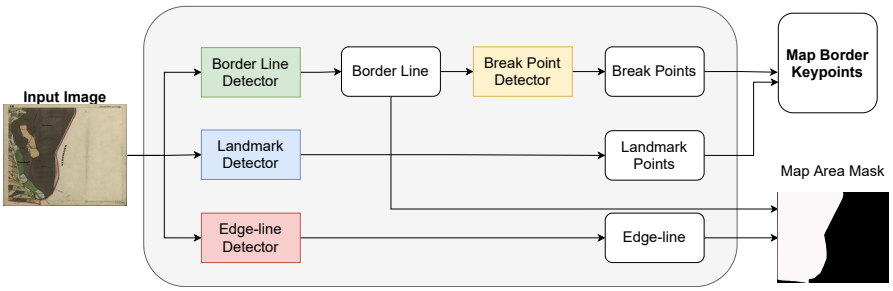


Fig. 3: Overall scheme of the map sheet processing pipeline

4.1 Landmark Detector

The landmark detection algorithm utilizes an FCN trained to predict landmark positions. We use the architecture proposed by Wick and Puppe [10]. In our preliminary experiments we compared this architecture with U-Net [11], which is one of the first and most popular FCN architectures, and obtained comparable results. Moreover, U-Net has ten times more parameters and is computationally much more demanding.

Because of the large size of the input images and the need of preserving relatively small details, we chose a patch-based approach instead of processing the whole (resized) map sheet at once. The training is performed on rectangular image crops that were extracted randomly from annotated training examples. We

only ensure that each extracted patch contains at least one landmark if there are any in the map sheet. See Section 5.1 for examples of annotated ground-truths.

In the prediction process, we divide the map sheet into a set of overlapping patches and predict each of them by the network. The prediction result is a composition of individual patches predictions. It is a black and white image where white points represent the predicted landmarks.

We then apply a post-processing step based on the connected components analysis. After that we perform a reduction of false positives. We reduce components with size smaller than a specified area threshold (50 px). The centres of remaining components comprise the resulting set of landmark positions.

The further filtration of the resulting points is done according to the detected edge-line (see Sec. 4.4). We keep only the points that are sufficiently close (defined by distance threshold 20 px) to the edge-line.

4.2 Border Line Detector

The border line detector uses the same neural network architecture as the landmark detector. The training is also performed using patches randomly extracted from the training images. We take only patches coinciding with the border line. The trained network is used for patch-wise prediction of a map sheet. The final prediction result contains the predicted border line mask.

4.3 Break Point Detector

The input of this algorithm is the detected border line. Our goal is to find significant direction changes on the line. The predicted border line may have varying width and it is also disconnected in some cases. We therefore apply morphological closing with a circular structuring element (size 10 pixels) in order to fill small holes in the border. Then we compute the skeleton [12] to get a one-pixel wide representation. The skeleton is further simplified using an implementation of *Douglas-Peucker* algorithm [13]. This way we obtain a line (or several line segments) represented by a series of points.

The final step is the reduction of non-significant angles. We inspect all triplets of consecutive points (P_1, P_2 and P_3) and calculate difference in direction of vectors (P_1,P_2) and (P_2,P_3). If this direction difference α is lower than specified angle threshold (we use a value of 10 degrees), we discard point P_2. Figure 4 illustrates the process of direction difference computation.

4.4 Edge-line Detector

We use traditional image-processing techniques (edge detection, flood fill, morphology etc.) for this task. We divide the pipeline into the following major steps and describe each of them separately.

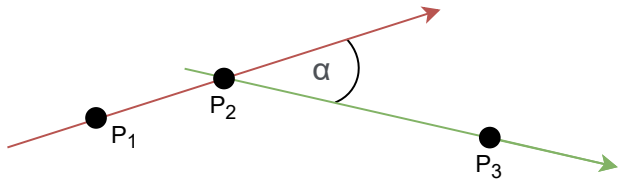


Fig. 4: Illustration of direction change at three consecutive border-line points

Edge Detection The goal of this step is to create a binary edge map. To partially suppress noise in the input, median filter (with kernel size 9) is used. Edges are then detected by Sobel operator [14] for each RGB channel separately. A single-channel version is created by taking maximal difference among the channels. To create the binary edge map, we use Otsu's thresholding [15].

Components Detection In this step, we identify areas (components) with similar colors. We first transform the RGB color space into HSV that is more suitable for this task.

Then we detect initial components by iterating the image pixel by pixel. For every pixel P of the image that is not already part of any component or is not an edge pixel, we start the flood filling algorithm [16]. The newly processed pixel Q is added to the current component only if it satisfies the following conditions:

1. Q is not an edge pixel - its value in edge map corresponds to the background;
2. Differences between pixels P and Q in H and S channels are lower than a specified threshold (pixels are considered to have a similar color).

The result of the initial components detection can be seen in the left part of Figure 5. It can be observed that there are many small, noise-like components. We first filter "holes" formed by edge detection by merging them with the neighboring component (if there is just one).

Second, we iteratively search for small components that have only one unique surrounding neighbor. This process utilizes a max-heap implementation. For the remaining components, we apply morphological opening which removes fine details and jagged edges. Thus we improve stability and performance. The result of the filtering can be seen in the right part of Figure 5.

Components Filtering We further filter the remaining components. We focus on long and narrow components that should represent edge-line candidates. To obtain the width of the components, we use signed distance field (SDF) [17]. The length of the component is obtained from its skeleton [12]. Components with sufficient length and width (according to the specific threshold) are marked as edge-line candidates.

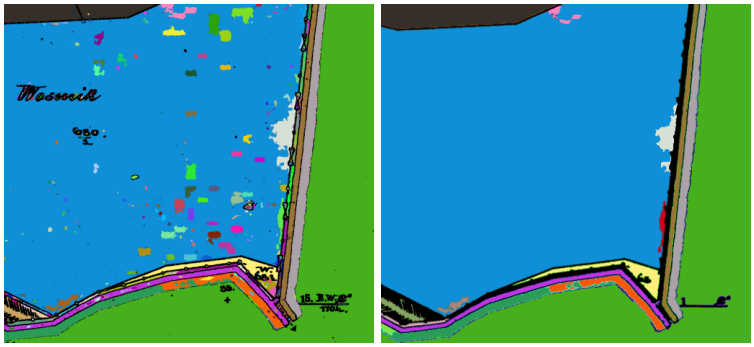


Fig. 5: Component detection example. Each component has a different color, the edges are black. Left: Initial components; Right: After filtering

Nonetheless, after the previous step, we still have many false positives (e.g. roads, rivers). To remove them, we exploit the knowledge that there is a border line with map symbols (mostly "dots") in the immediate vicinity of the edge-line as shown in the middle part of Figure 6.

The "dots" are detected using the V channel of the HSV color space.

We apply morphological opening to eliminate thin lines and texts that can cause false detections. Then we apply *Harris corner detector* [18] followed by another morphological opening for noise removal. The result is depicted in the right part of Figure 6.

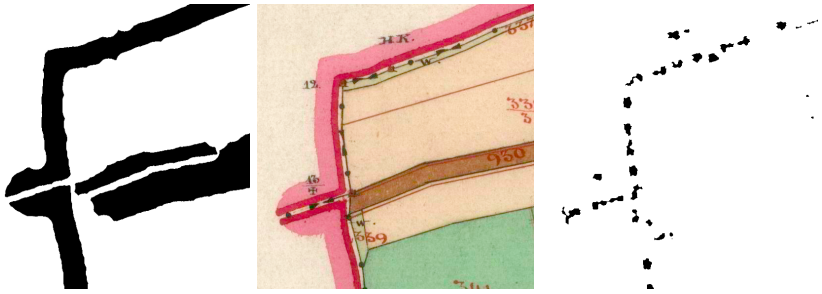


Fig. 6: Left: all detected edge-line candidates; Middle: original image; Right: "dots" on the border line

To filter out false positives, we use following conditions:

1. Majority of "dots" is placed around one side of the candidate;

2. The distance between neighboring "dots" is in a specific range based on the input images. In our experiments it was 20 – 100 px;
3. The distance between each "dot" and skeleton of the edge-line is in a specific range (10 – 40 px).

The final detected edge-lines together with associated "dots" are presented in Figure 7.

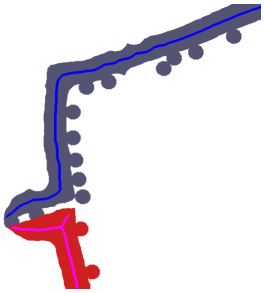


Fig. 7: The final detected edge-lines and their corresponding "dots"; The central line is a vectorized representation of the edge-line.

5 Experiments

5.1 Dataset

In order to obtain the training data for proper FCN training, we have created a dataset containing 100 map sheets. Annotated ground-truths also allow us to evaluate the presented methods.

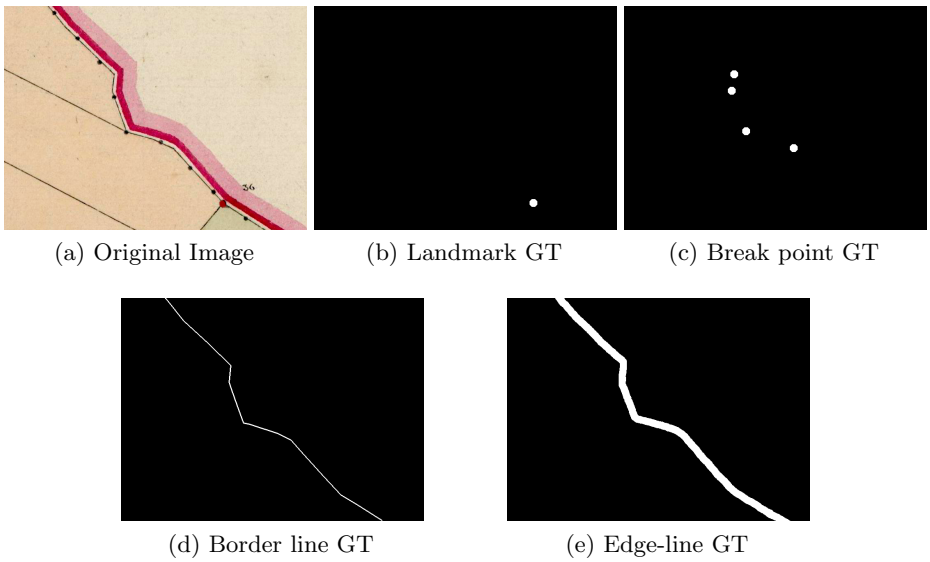
The dataset is split into training (70 images), validation (10 images) and testing (20 images) parts. For each of the map sheets we have manually annotated a set of ground-truth images for landmarks, border line, break points and edge-line. The ground-truths for break points and edge-line were created only for the testing part because they are not used for network training.

All ground-truth types are stored as binary images with black background and the objects of interest are marked with white color. Landmarks are marked as white circles with diameter corresponding approximately to the real size of the drawn landmarks.

Border lines are drawn as a white line, trying to have a similar width as the original border line. Break point ground-truths were annotated in the same way as landmarks – white dots represent significant direction changes in the border line. Edge-line ground-truths contain white mask for the red edge-line. The latter two types of ground-truths are used only for evaluation purposes. All

types of ground truths are shown in Figure 8. Note that the crop contains just one explicitly drawn landmark symbol. However, the break-point ground-truth contains 4 points in border line direction changes.

The dataset is freely available for research and education purposes at[3].



(a) Original Image (b) Landmark GT (c) Break point GT

(d) Border line GT (e) Edge-line GT

Fig. 8: Example of a map sheet fragment with corresponding ground-truths

5.2 Evaluation Criteria

For the evaluation of landmark and break point detectors, we present the standard precision (P), recall (R) and F1 score ($F1$). The edge-line is represented as a set of consecutive points and thus we use the same evaluation criteria. We first count true positives (TP), false positives (FP) and false negatives (FN) as follows.

For each ground-truth point, we identify the closest predicted point. If their distance is smaller than a certain distance threshold T the point is predicted correctly (TP). Otherwise, it is counted as either FP (present only in prediction) or FN (present only in ground-truth).

Visualization of the evaluation process is depicted in Figure 9. It shows a fragment of a processed map sheet with marked TP, FP and FN points. Green circles denote TP points, yellow circles are used for FN and yellow crosses for FP points. The colored circles show the distance threshold T used for computation of the scores.

[3] hidden_during_review

Fig. 9: Visualized TP (green circles), FP (yellow crosses) and FN (yellow circle) points

5.3 Landmark Detection

The crucial part of the landmark detector is the network trained for landmark prediction. We performed an experiment with different network configurations in order to find suitable parameters. We utilized several sizes of the training patches. The patch size determines the context the network takes into consideration and can thus influence the overall result. We also must consider that larger patches are computationally more demanding.

We also varied the number of patches extracted from one training map sheet. The main reason was to find a minimum amount of patches that provides enough variability to learn the features.

We use the cross-entropy loss for network training and as an optimizer we utilize Adam [19] with initial learning rate set to 0.001. We apply early-stopping based on validation loss to prevent overfitting as well. Results for different patch sizes and different numbers of training patches extracted from one image are presented in Table 1. The best performing configuration is typeset in bold.

We report the scores for two values of distance threshold T. Smaller threshold causes lower scores, however, it ensures that the predicted points are really close to the ground-truth ones. The ground-truth images contain 33 break points in average. The presented recalls are thus sufficient for our goal ($5 - 10$ detected landmarks are usually enough for the transformation and connection).

When using $T = 5$, the average distance of the corresponding points is 1.3 px. Using $T = 2$ we reduce this value to 0.8 px. We have performed a comparative experiment with landmarks annotated by two human annotators. On three randomly selected map sheets we obtained average distance of 1.5 px between

the annotations. We thus can state that the presented approach performs better than a human annotator and the resulting set of detected landmarks is usable for map sheets connection.

Table 1: Evaluation of the landmark detection algorithm with different sizes and numbers of the training patches extracted from one map sheet

		Threshold 5			**Threshold 2**		
	Patch Size ($w \times h$)	**P**	**R**	**F1**	**P**	**R**	**F1**
10 patches	320 × 240	37.6	34.7	36.1	22.2	19.8	20.9
	640 × 480	65.0	56.8	60.6	31.5	27.9	29.6
	960 × 720	85.5	76.2	80.6	62.2	49.7	55.2
	1280 × 960	86.5	81.3	83.8	66.0	57.0	61.2
25 patches	320 × 240	53.9	37.9	44.5	33.3	21.0	25.8
	640 × 480	74.6	36.4	48.9	54.5	25.1	34.4
	960 × 720	84.5	80.6	82.5	61.7	53.4	57.2
	1280 × 960	82.2	31.7	45.7	63.9	21.1	31.7
50 patches	320 × 240	79.3	76.2	77.7	54.8	48.7	51.6
	640 × 480	85.2	84.7	84.9	62.7	58.3	60.4
	960 × 720	86.0	71.7	78.2	65.2	50.6	57.0
	1280 × 960	84.5	85.3	84.9	61.2	57.6	59.4
75 patches	320 × 240	74.2	90.5	81.5	56.0	64.9	60.1
	640 × 480	78.9	84.7	81.7	58.6	59.7	59.2
	960 × 720	**88.3**	**82.0**	**85.1**	**67.3**	**58.2**	**62.4**
	1280 × 960	88.4	38.7	53.9	71.7	28.0	40.3
100 patches	320 × 240	78.7	89.2	83.6	59.0	59.3	59.1
	640 × 480	76.5	37.7	50.5	53.9	24.6	33.7
	960 × 720	86.5	80.0	83.2	59.6	50.9	54.9
	1280 × 960	87.1	81.6	84.2	65.2	56.1	60.3

5.4 Border Line and Break Points Detection

In this section, we experiment with neural network configurations in the same way as for the landmark detection. We do not evaluate the predicted border line. Instead of that, the evaluation is performed for the detected break points using the same criteria as in the case of landmarks. Also the settings and hyper-parameters of the network are the same.

The results are summarized in Table 2. We report the scores for two values of distance threshold, namely $T = 5$ and $T = 2$. We can observe that the scores are significantly lower compared to the landmark detection. However, the recall exceeding 60 % is still sufficient for finding enough points usable for the map sheet connection. It is also important to note that the break point detector is applied only on map sheets containing insufficient amount of landmarks which happens only in a minority of cases.

5.5 Edge-line Detection

To validate the results of the proposed solution, we have compared our detected edge-lines against the ground-truth data.

Table 2: Evaluation of the break point detection algorithm with different sizes and numbers of training patches

		Threshold 5			Threshold 2		
	Patch Size ($w \times h$)	**P**	**R**	**F1**	**P**	**R**	**F1**
10 patches	320 × 240	13.6	33.6	19.3	4.0	9.4	5.6
	640 × 480	10.8	34.7	16.5	3.1	7.2	4.4
	960 × 720	14.9	38.3	21.5	3.6	9.2	5.2
	1280 × 960	25.1	58.7	35.2	7.1	18.1	10.2
25 patches	320 × 240	6.4	42.0	11.1	0.9	6.2	1.6
	640 × 480	19.6	50.8	28.3	6.6	16.5	9.4
	960 × 720	20.7	46.6	28.7	7.3	14.5	9.7
	1280 × 960	34.0	45.9	39.1	7.6	17.0	10.5
50 patches	320 × 240	14.2	54.6	22.6	4.7	16.9	7.4
	640 × 480	26.0	63.7	36.9	7.5	18.3	10.7
	960 × 720	28.3	52.8	36.8	9.5	17.5	12.3
	1280 × 960	32.3	50.2	39.3	8.6	19.3	12.0
75 patches	320 × 240	15.9	50.6	24.1	6.4	20.5	9.8
	640 × 480	20.3	49.2	28.8	6.4	15.8	9.2
	960 × 720	35.1	46.5	40.0	9.7	17.6	12.5
	1280 × 960	37.5	52.6	43.8	6.3	12.8	8.5
100 patches	320 × 240	14.5	57.1	23.2	5.1	17.1	7.9
	640 × 480	30.7	53.9	39.2	4.1	9.0	5.7
	960 × 720	32.6	58.5	41.9	**12.9**	**24.9**	**17.0**
	1280 × 960	**33.9**	**65.5**	**44.7**	8.9	18.4	12.0

To overcome problems with the detected edge-lines that may contain noise, we have vectorized the edge-lines in both datasets. The lines are represented as sets of equally distant points. Therefore, instead of pixel-based comparisons, we use vector-based calculations that are more robust. The downside is that the points from ground-truth and the results may not be aligned. For comparison, we have used the solution with a threshold defined in Section 5.2. We have tested three different threshold values T selected to be near the median distance between points in the ground-truth dataset (median distance is 27.3 px). We present the results of the edge-line detection in Table 3.

Table 3: Evaluation of edge-line detection algorithm

Threshold	**P**	**R**	**F1**
20	73.6	75.4	74.5
30	80.1	81.5	81.2
40	81.5	86.4	83.9

This table shows clearly, that the best results are obtained for the threshold value of 40. This configuration gives very high F1 score (F1 = 83.9%). This quality

of detection is sufficient for our needs. We do not require detected edge-line to be pixel accurate. Our goal is to have a correct orientation against the border line.

5.6 Final Results

In this final experiment, we used the above described methods on the whole task of the seamless map sheet connection and visualize the results.

Fig. 10: Example of connected maps (left) with the detail of the connection in the complicated areas (right)

Figure 10 shows one example of the resulting connection of two neighboring map sheets. The corresponding detected break points on these sheets are marked by the different blue cross symbols. Based on the manual analysis of the small sample containing 100 of resulting images, we can claim that the proposed approaches are sufficient to be integrated into the final system.

6 Conclusions and Future Work

We have presented a set of methods for the detection of important features in historical cadastral maps. The methods represent the fundamental part of a system for seamless connection of the individual map sheets. We have focused on finding important points lying on cadastre border usable for seamless connection of neighboring map sheets. Another task is the detection of the cadastre border which allows masking of the area outside the map. The presented methods are built upon FCN networks in combination with traditional computer vision techniques. Only the task of edge-line detection is solved using solely traditional computer vision techniques. The reason is a relatively more complicated annotation of ground-truths for this task and also the sufficient results obtained by the presented solution.

The methods are evaluated on a newly created dataset containing ground-truths for all four solved tasks. We have compared several configurations of the

algorithms and searched for the best performing ones. We have further visually demonstrated that the best implemented methods have sufficient accuracy to be integrated into the final system.

We have utilised a standard FCN architecture and there is surely a room for further improvements using different architectures. We can also concentrate on strategies of networks training, e.g. balance between positive and negative training samples etc.

Acknowledgement

This work has been partly supported by Grant No. SGS-2019-018 Processing of heterogeneous data and its specialized applications.

References

1. G. Timár, G. Molnár, B. Székely, S. Biszak, J. Varga, and A. Jankó, *Digitized maps of the Habsburg Empire The map sheets of the second military survey and their georeferenced version*, 01 2006. 2
2. J. Ignjatić, B. Nikolić, and A. Rikalović, "Deep learning for historical cadastral maps digitization: overview, challenges and potential," 2018. 2
3. S. Timilsina, S. Sharma, and J. Aryal, "Mapping urban trees within cadastral parcels using an object-based convolutional neural network," *ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. 4, pp. 111–117, 2019. 3
4. V. Ostankovich and I. Afanasyev, "Illegal buildings detection from satellite images using googlenet and cadastral map," in *2018 International Conference on Intelligent Systems (IS)*. IEEE, 2018, pp. 616–623. 3
5. X. Xia, C. Persello, and M. Koeva, "Deep fully convolutional networks for cadastral boundary detection from UAV images," *Remote sensing*, vol. 11, no. 14, p. 1725, 2019. 3
6. B. Fetai, K. Oštir, M. Kosmatin Fras, and A. Lisec, "Extraction of visible boundaries for cadastral mapping based on UAV imagery," *Remote sensing*, vol. 11, no. 13, p. 1510, 2019. 3
7. E. Nyandwi, M. Koeva, D. Kohli, and R. Bennett, "Comparing human versus machine-driven cadastral boundary feature extraction," *Remote sensing*, vol. 11, no. 14, p. 1662, 2019. 3
8. R. Kestur, S. Farooq, R. Abdal, E. Mehraj, O. S. Narasipura, and M. Mudigere, "Ufcn: A fully convolutional neural network for road extraction in rgb imagery acquired by remote sensing from an unmanned aerial vehicle," *Journal of Applied Remote Sensing*, vol. 12, no. 1, p. 016020, 2018. 3
9. Y. Chen, E. Carlinet, J. Chazalon, C. Mallet, B. Duménieu, and J. Perret, "Combining deep learning and mathematical morphology for historical map segmentation," *arXiv preprint arXiv:2101.02144*, 2021. 3
10. C. Wick and F. Puppe, "Fully convolutional neural networks for page segmentation of historical document images," in *2018 13th IAPR International Workshop on Document Analysis Systems (DAS)*. IEEE, 2018, pp. 287–292. 5
11. O. Ronneberger, P. Fischer, and T. Brox, "U-Net: Convolutional Networks for Biomedical Image Segmentation," *Miccai*, pp. 234–241, 2015. 5

12. T. Y. Zhang and C. Y. Suen, "A fast parallel algorithm for thinning digital patterns," *Commun. ACM*, vol. 27, no. 3, p. 236239, Mar. 1984. [Online]. Available: https://doi.org/10.1145/357994.358023 6, 7
13. D. H. Douglas and T. K. Peucker, "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature," *Cartographica: The International Journal for Geographic Information and Geovisualization*, vol. 10, no. 2, pp. 112–122, 1973. 6
14. N. Kanopoulos, N. Vasanthavada, and R. L. Baker, "Design of an image edge detection filter using the sobel operator," *IEEE Journal of Solid-State Circuits*, vol. 23, no. 2, pp. 358–367, 1988. 7
15. N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979. 7
16. S. Burtsev and Y. Kuzmin, "An efficient flood-filling algorithm," *Computers & Graphics*, vol. 17, no. 5, pp. 549 – 561, 1993. [Online]. Available: http://www.sciencedirect.com/science/article/pii/009784939390006U 7
17. P. F. Felzenszwalb and D. P. Huttenlocher, "Distance transforms of sampled functions," *Theory of Computing*, vol. 8, no. 19, pp. 415–428, 2012. [Online]. Available: http://www.theoryofcomputing.org/articles/v008a019 7
18. C. Harris and M. Stephens, "A combined corner and edge detector," in *Proceedings of the 4th Alvey Vision Conference*, 1988, pp. 147–151. 8
19. D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.