Vectorization of Historical Maps Using Deep Edge Filtering and Closed Shape Extraction*

Yizi Chen[1,2][0000−0003−1637−0092], Edwin Carlinet[1][0000−0001−5737−5266], Joseph Chazalon[1][0000−0002−3757−074X], Clément Mallet[2][0000−0002−2675−165X], Bertrand Duménieu[3][0000−0002−2517−2058], and Julien Perret[2,3][0000−0002−0685−0730]

[1] EPITA Research and Development Lab. (LRDE), EPITA, France
[2] Univ. Gustave Eiffel, IGN-ENSG, LaSTIG
[3] LaDéHiS, CRH, EHESS

Abstract. Maps have been a unique source of knowledge for centuries. Such historical documents provide invaluable information for analyzing the complex spatial transformation of landscapes over important time frames. This is particularly true for urban areas that encompass multiple interleaved research domains (social sciences, economy, etc.). The large amount and significant diversity of map sources call for automatic image processing techniques in order to extract the relevant objects under a vectorial shape. The complexity of maps (text, noise, digitization artifacts, etc.) has hindered the capacity of proposing a versatile and efficient raster-to-vector approaches for decades. We propose a learnable, reproducible, and reusable solution for the automatic transformation of raster maps into vector objects (building blocks, streets, rivers). It is built upon the complementary strength of mathematical morphology and convolutional neural networks through efficient edge filtering. Evenmore, we modify ConnNet and combine with deep edge filtering architecture to make use of pixel connectivity information and built an end-to-end system without requiring any post-processing techniques. In this paper, we focus on the comprehensive benchmark on various architectures on multiple datasets coupled with a novel vectorization step. Our experimental results on a new public dataset using COCO Panoptic metric exhibit very encouraging results confirmed by a qualitative analysis of the success and failure cases of our approach. Code, dataset, results and extra illustrations are freely available at https://github.com/soduco/ICDAR-2021-Vectorization.

Keywords: Image processing · Deep Learning · Maps · Vectorization · Semantics · Edge Filtering · Shape extraction.

* This work was partially funded by the French National Research Agency (ANR): Project SoDuCo, grant ANR-18-CE38-0013. We thank the City of Paris for granting us with the permission to use and reproduce the atlases used in this work.

Fig. 1: Example (cropped) of a raster map used as input (top) and output of our vectorization approach (bottom): a set of individual polygons in vector format ready for further annotation in any GIS software. Some shapes are still undetected.

1 Introduction

Historical maps are unique and powerful tools for understanding the transformations of the geographical space over unique time spans. They are invaluable inputs in historical sciences, architecture, and urban planning. The massive digitization of archival collection resources carried out by heritage institutions dramatically increases the amount of geospatial information available for certain areas of the world. In the Western world, the rapid development of geodesy and cartography from the 18^{th} century resulted in massive production of topographic maps at various scales. City maps are of utter interest. They contain rich, detailed, and often geometrically accurate representations of numerous geographical entities. Maps also document the distribution in space and the topological relationship of the depicted entities, while legends and text labels provide semantic information, in particular about their functions [22,10]. Recovering spatial and semantic information represented in old maps requires a so-called *vectorization* process.

Vectorizing maps consists in transforming rasterized graphical representations of geographic entities (often maps) into instanced geographic data (or vector data), that can be subsequently manipulated (using Geographic Information Systems, GIS). This is a key challenge today to better preserve, analyze and disseminate content for numerous spatial and spatio-temporal analysis purposes.

From an image processing and a document analysis perspective, vectorization can be cast by the following, often interleaved, problems: (i) isolate the map content on pictures of map sheets (leave out the legend, in particular);

(ii) detect and separate the various layers of graphical content: points, lines, and shape objects, as well as symbols and text; (iii) classify / recognize each graphical object of interest (including text), while ensuring a topologically and geometrically consistent result (often considered as an *instance segmentation* issue); (iv) georeference the extracted elements.

In this paper, we focus on closed shape detection in the 19^{th} and early 20^{th}-century historical map atlases of Paris (France). Currently, shape detection is usually manually performed, using GIS software. Such a costly and tedious process leads to heterogeneous data quality. The latest methodological developments in image processing leverage the ability to automatically build a significant number of geo-historical databases, which eventually benefit to multiple research areas. Unfortunately, unlike computer-generated maps, roughly following the same semiotic rules, historical maps steadily vary in terms of legend, level of generalization, type of geographic features, and text fonts. Even more, objects in historical maps exhibit very limited color and texture information. This creates ambiguities in interpretation, leading to the failure of main texture-based semantic [24,3,31]/instance segmentation [6,30,4] and raster-to-vector [23] approaches. Such an issue is exacerbated by the frequent overlap between map objects and occlusion with the carto-geodetic information (vertical and horizontal lines). Their instantiation become more complex, with often non continuous boundary retrieval. Last, ink+paper aging and damage in the historical image such as erased lines, bad contrast, noisy content, tearing, folding might cause gaps in the cartographic information and lead to poor object detection. To tackle those challenges, we are therefore interested in developing a versatile shape detection approach, loosely coupled with the style of a given map. We aim to accelerate the detection of core city structures (building blocks, rivers, street networks), as well as the georeferencing process while keeping both very accurate.

We recently found that extracting closed shapes from historical maps [8] was feasible, using a deep learning architecture as an edge filter (rather than trying to predict instances directly), connected with mathematical morphology tools to extract closed shapes from an edge map. The current paper improves over these preliminary findings by exploring more combinations of deep edge filters, followed by closed shape extraction (Sec. 3), and exploring all these potential architectures in an extensive benchmark (Sec. 4).

2 Related work

Extracting the content of historical maps is an active topic in Geographic Information Sciences, in an effort to provide tools and methods to build large spatio-temporal datasets and historical gazetteers. This is a special case of the broader topic of digital map processing where most challenges are exacerbated by the higher complexity and heterogeneity of ancient maps compared to the more homogeneous modern maps [11,9]. Three main categories of extraction strategies can be distinguished: manual vectorization with GIS tools, automatic solutions, and hybrid frameworks. They either focus on structured geographical

entities (roads, buildings) or textual information (named places mainly).
Manually extracting the content of historical maps with GIS tools is still the main strategy in digital humanities. This solution is conceivable when small-size datasets are handled, in space and time, for peculiar case studies [26]. Collaborative approaches have been proposed to handle larger map corpora, ranging from a limited number of contributors [27] to large-scale crowdsourcing experiments [34,5]. Although manual extraction creates "data-intimacy" conducive to reflexive work, the process is tedious, time-consuming, and leads to non-reproducible results.
Automatic solutions are dedicated either to specific symbols and local features extraction [14] or to map vectorization, i.e., (semantic) shape retrieval using raster-to-vector techniques. Many image vectorization solutions exist [18] but cannot be applied to complex historical maps: design, multiplicity of objects, noise, limited graphical quality (no color nor texture) hinder their performance. Approaches dedicated to old maps rely on the following unsupervised workflow: color-based segmentation of objects, binary (shape) filtering, cleaning, and vectorization. They exhibit two main drawbacks: a priori knowledge on color and object shapes narrow down the versatility power of the methods [21], and no focus is made on extracting multiple closed shapes, under a learning paradigm. A suitable trade-off is found with hybrid methods that would target to learn shape extraction from crowdsourcing data. First attempts have shown their relevance [5] but closer interaction between extraction and annotation is desired to really benefit from both fields.

In Mathematical Morphology, the watershed transform is a *de facto* standard approach for image segmentation, and it has been widely studied in terms of topological properties [13,29] and computational efficiency [13,12]. While being efficient on rather clean images, the watershed is hard to apply in real, complex images. This hinders wider applications related to closed shape extractions.For natural images, Hanbury et al. [16] use a watershed algorithm to segment closed shapes from the learned gradient image extracted, while Arbelaez et al. [2] introduce oriented watershed to extract closed shapes in global probability boundaries created by several image features and cues. Unfortunately, there are very few applications that adopt watershed transform in spatial data, especially cartographic images. To our best knowledge, only one approach uses watershed transform to extract spatial semantic objects through color and geometrical features in color cartographic images [1]. However, this method is hard to apply in our historical map image, which has very limited colors and textual information.

Detecting edges from images is a widely studied subject. Firstly, it has been achieved with handcrafted features such as brightness, color, and textures [25]. After that, those features are efficiently grouped by mono-scale and multi-scale attributes retrieving micro-structures such as textons and salient examples [37]. This led to methods focusing on combining all existing features [2]. The weights are learned to efficiently combine the relevant cues such as gradients and textons in multiscale images [2] to estimate a probability of boundary. Then, an

ultrametric contour map (UCM) is created by using probability boundary as input to extract closed shapes to represent image objects, turning soft assignments into hard ones. Recently, Convolutional Neural Networks (CNNs) have proved their relevance to extract and combine meaningful image features to delineate semantic objects. A large amount of research has focused on semantic edge detection. The most famous deep edge detector is called holistically edge detector(HED) [35] which is an end-to-end deep learning multi-scale architecture, built upon a VGG-16 backbone network[33]. The novelty lies in adopting skip-connections to combine multiple levels of features, while different losses are measured in the intermediate outputs of VGG-16 to filter out useful edge features at each stage of the network. It allows to better learn and recover multi-scale representations of image features. Recently, He et al. [17] proposed a so-called Bi-directional cascade network (BDCN) by designing a scale-enhancement module (SEM) and adding to every intermediate output of the HED to learn diversity edge features and enhance spatial contexts: traditional convolution kernels are substituted by dilated convolution kernels [36]. Most of the deep edge detectors are encoder-decoder structures (U-Net fashion [31]), built based on VGG-16 architecture which can use ImageNet [32] pretrained features for limiting the discrimination task to a transfer learning process: convergence is accelerated and accuracy is increased.

Very few papers are fully dedicated to historical map processing. Petitpierre et al. [28] use U-Net [31] and SegNet [3] architectures to perform semantic segmentation on historical maps of various kinds and locations. They focus on few classes and do not propose a vectorization step, required to extract semantized instances. To tackle this issue, we provide a complete, trainable, raster-to-vector solution to extract closed shapes from historical maps with little to no texture or color information by combining the power of deep edge detector as a filtering tool with strong guarantees of closed shape extraction using mathematical morphology. To our best knowledge, to automate this process as far, we are among the only approaches with a strong commitment to being reproducible and reusable; all our code and data is open source and public.

3 Raster-to-Vector Pipeline

Our process for the raster-to-vector conversion of historical map images is divided into 3 main stages, as illustrated by Fig. 2. We introduce several possibilities for each stage, detailed hereafter. Stage 1 (Deep Edge Filtering) can be composed of either HED, BDCN, U-Net, or ConnNet standard architectures. It consists in producing an Edge Probability Map (EPM) from the raster input. Stage 2 (Closed Shape Extraction) can be composed of either a connected component (CC) labeling or a watershed (WS). We generate a Label Map (LM), which assigns a shape identifier to each pixel. Stage 3 (Vectorization) is performed using an off-the-shelf vectorization tool of the GDAL open-source project [15] as a proof of concept, leading to results like the one illustrated in Fig. 1.

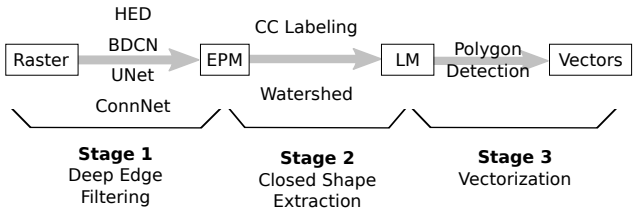

Fig. 2: Overview of our raster-to-vector pipeline and of the evaluated alternative techniques for each stage. Stage 1 produces an Edge Probability Map (EPM), which estimates how likely a pixel belongs to a semantic edge, and Stage 2 produces a Label Map (LM), where each pixel receives the label of the shape it belongs to.

3.1 Deep edge detectors

HED The Holistically edge detector (HED) is a state-of-the-art deep edge detection architecture, with a so-called *skip-layer network training* procedure: different levels of intermediate features and merge into the final stage of the output layer [35]. HED is built based on VGG-16 architecture. Intermediate outputs are chosen from VGG-16 to select multi-level features from several scales. Those outputs are upsampled and concatenated, followed by 1×1 convolution to fuse multiple channel outputs into an EPM image. We will consider both a complete training and a fine-tuning from weights pre-trained using natural images (ImageNet).

BDCN BDCN is another reference network, built on VGG-16, with some modifications compared to HED. Scale enhancement module (SEM) [17] modules are added to the intermediate outputs. The SEM modules are created by using dilated convolution [7], which enlarges the receptive field of kernels without downgrading the resolution of the feature maps [17]. In the end, all the intermediate output are similarly upsampled, concatenated, and fused into one EPM image. We will consider both a complete training and a fine-tuning from weights pre-trained using natural images (ImageNet).

U-Net The U-Net architecture [31] is a U-shape architecture inspired by Fully convolution networks [24]. It contains two different paths to combine image features. The first path is called the contracting path, which is using downsampling to get deep and semantic features. The other one is the expansive path, which concatenates high resolution features with spatial information from downsampling, using the outputs of the up-convolution features. U-Net has a symmetrical architecture that can retain well spatial information of pixels leading to very accurate localizations. This is an advantage on boundary detection task.

ConnNet ConnNet [19] architecture is invented by using pixel connectivity information to predict salient objects. In this paper, we decided to encode and predict pixel connectivity information for every pixel in the output. It is a learnable way to binarize EPM, making use of pixel-based spatial information at training time. The original ConnNet exhibits a significant amount of parameters and may not be efficient in our data. We modify the original ConnNet architecture by simply concatenating two networks, a deep edge detector (BDCN in this paper but any deep edge detector can be replaced here) with an encoder-decoder architecture (U-Net, but, again any full convolution architecture is relevant). The output of ConnNet is a 4- or 8-channel probability map which predicts for each pixel its 4- or 8-connectivity with each of its neighbors.

3.2 Training HED, BDCN and U-Net

Our input image is $x \in \mathbb{R}^{H \cdot W}$ and the ground truth label us $y \in 0, 1^{H \cdot W}$. The output of the predicted image is $\hat{y} = f(x, w) \in 0, 1^{H \cdot W}$ and every element of $\hat{y}$ is interpreted as the probability of pixel i having label 1: $\hat{y} \equiv p(Y_i = 1|x, w)$. Binary cross entropy loss is used as loss function, measuring the difference between edge predictions and labels. Due to the highly imbalanced nature between non-edge (97.5% in training and validation sets) and edge pixels (only 2.5%), α, β are used to re-balance the binary cross entropy loss. It is formulated as:

$$\mathcal{L}_{BCE} = -\alpha \sum_{j \in Y_-} log(1 - \hat{y_j}) - \beta \sum_{j \in Y_+} log(\hat{y_j}), \tag{1}$$

where Y_+ is the set of indices of edge pixels, Y_- is the set of indices of non-edge pixels, $\alpha = (\lambda \cdot |Y_-|/(|Y_+| + |Y_-|))$ is the percentage of edge pixels in each batch of the historical map image, and $\beta = (|Y_+|/(|Y_+| + |Y_-|)))$ is percentage of non-edge pixels. An extra $\lambda = 1.1$ factor is used to enhance the percentage of edge pixels in order to give extra weights for edge responses.

3.3 Training ConnNet

We do detail the ConnNet training procedure. Please refer to [19]. Here we only mention the process of measuring the loss function of ConnNet, which is a combination of an edge loss and a connectivity loss:

$$\mathcal{L}_{Connnet} = \mathcal{L}_{edge} + \mathcal{L}_{pixcon}. \tag{2}$$

$\mathcal{L}_{edge}$ is the binary cross entropy loss from Equ. 1 for edge detection, and $\mathcal{L}_{pixcon}$ is the pixel connectivity loss. L_{pixcon} is measured by the binary cross-entropy loss between connectivity labels and predictions [19]. The connectivity label y^c is created by encoding surrounding location information of the pixel. It results in eight channels, every channel represents the connectivity information in a specific direction. Then, the pixel connectivity loss L_{pixcon} is measured in Equ. 3 by using the binary cross entropy loss (Equ. 1) between connectivity labels and

predictions [19]. In Equ. 3, $\hat{y}^c$ is the probability of the predicted connectivity, y^c represents the connectivity labels, N is the number of pixels in images, and C is the number of connected pixels, respectively. $\frac{1}{(N \times C)}$ is the normalizing term for the loss.

$$\mathcal{L}_{pixcon} = \frac{1}{N \times C} \sum_{c=1}^{C} \sum_{i=1}^{N} L_{BCE}(\hat{y}_i^c, y_i^c). \tag{3}$$

3.4 Closed Shape Extraction

Closed shapes can be extracted from EPMs either through connected component labeling or watershed transform.

Connected component labeling requires a binarized edge map as input and subsequently a binarization threshold θ in order to provide a label map.

The watershed transform directly necessitates the EPM as input and returns 1-pixel thin edges and a label map. It offers a strong guarantee of closed shape extraction with efficient implementation. The strength of the watershed is to recover the boundaries of objects even on weak edge responses that would be lost by EPM thresholding. The filtering parameters (dynamic δ and minimum area σ) are important to control the trade-off between the fact we want to recover small/leaking regions (somewhat related to the recall) and the false-detection of boundaries (somewhat related to the precision). The h-minima characterize the importance of each local minimum through their **dynamic**. When flooding a basin, it refers to the water elevation required to merge with another basin. The calibration of this parameter depends on the intensity of the edges detected by the deep edge detector, and needs to be tuned accordingly. The other parameter is the **minimum area** of the components which can be easily known as prior knowledge in our historical maps, for example, objects with a size that below 100 m^2 do not appear in our map. Even if the edge response is low (i.e., weak gradient), the watershed can consider this weak response and closes the contour of the region.

4 Benchmark

Here, we conduct an extensive comparison of the performance of the different modules for Stages 1 (6 variants) and 2 (2 variants) of our pipeline.

4.1 Datasets and training data generation

The dataset used for training and testing the networks is one of the collections of Paris atlases. Two particular sheets in the years 1898 and 1925 are selected, focusing on the central area of the city, which exhibits high landscape diversity. These large maps were digitized with high resolution resulting in also large image sizes. Those two sheets share similar graphical content but have content, contrast, and preservation differences.

We annotated all objects in the map image by creating line vector information for each boundary to represent and label every object of interest in the map. From this vector information, we created the target outputs for each of our processing stages. We rasterized the borders of the polygons to produce a binary edge image used as the target for the Edge Probability Map deep edge filters of Stage 1 must produce. Borders were dilated to 3 pixels to match the average border thickness in the original images. Then, we created a raster label map where an integer label is assigned to every pixel of the image depending on the shapes it falls into. We assigned a special zero label to the parts of the image which did not contain map content (borders and titles); this was used to filter the outputs of Stages 1 and 2 to ensure only relevant areas were processed.

Finally, the 1925 map sheet was split into two distinct parts and used as a training and validation set, and the 1898 map sheet was used as a test set. The resulting training image has a 4,500×9,000 pixel size, with 3,343 objects, while the validation image has a size of 3,000×9,000 pixels with 2,183 shapes, and the test image has a size of 6,000×5,500 pixels with 2,836 shapes.

4.2 Protocol

Our evaluation protocol aims to identify the best deep edge detector for Stage 1 (first experiment), as well as to validate the relevance of the watershed for Stage 2 under optimal parameters settings (second experiment). The final polygon vectorization step is not used in this quantitative analysis; it was used as a pre-defined post-processing step to demonstrate that our approach effectively provides a suitable solution in the challenging sections of the raster-to-vector conversion of historical maps. The first experiment consists in comparing the performance of each deep edge detector, either trained from scratch or fine-tuned from pre-trained weights. For each of them, we check their performance on the validation set using the final metric for each training epoch. To this end, for each epoch of each detector, we compute the set of detected shapes using a connected component labeling computed on a binarized edge probability map (EPM) with a fixed threshold of 0.5. This creates a small bias in favor of our baseline for the second stage (CC labeling module). For each deep edge filter, we retain the model with the highest metric value to produce the EPMs for the second experiment. The second experiment consists in comparing the performance of the watershed module against a simpler baseline (connected component extraction) under the best set of parameters of each of them, given the best possible input from the previous stage. Using the EPMs produced by the best performing model for each variant in the first stage, we proceed in two steps. First, we calibrate the parameters for each closed shape extractor using the performance measured on the validation set. Secondly, we evaluate on the test set the performance of each combination of the best model for Stage 1 with the best calibration for each variant of Stage 2. The qualitative results illustrated in Fig. 1 are produced by vectorizing the output of the best performing combination on the test image.

Deep edge filtering variants and parameters. We consider six network variants: HED trained from scratch (**HED-scratch**), and fine-tuned using ImageNet pre-trained weights (**HED-pretrain**), BDCN trained from scratch (**BDCN-scratch**) and fine-tuned using ImageNet pre-trained weights (**BDCN-pretrain**), U-Net trained from scratch (**U-Net scratch**), and ConnNet trained from scratch (**ConnNet scratch**). As historical maps are scanned documents that have different image features compared to natural images, assessing the benefits of transfer learning is an important question. This explains why we test both HED and BDCN with and without a pre-training model on natural images. Such a pre-training was not available for U-Net and ConnNet and was discarded. During the training phase, we use the same settings for all variants: ADAM optimizer with an initial learning rate of $5e^{-5}$, a momentum of 0.9, and a weight decay of 0.002. We use an early-stopping scheme that limits the number of total epochs to consider, setting an upper limit to 60 epochs for each network.

Closed shape extraction variants and parameters. We consider two variants for the second stage of our pipeline: A connected component labeling (**CC labeling**) module used as a baseline, and a watershed (**Watershed**) module. As the CC labeling requires a binary image as input, it is necessary to binarize the Edge Probability Map produced beforehand, except in the case of ConnNet which already generates a binary image. We perform a grid search on 10 different thresholds ranging from 0.1 to 0.9 on the EPM prior to the labeling procedure. The watershed module is tuned through two parameters: a threshold on the EPM dynamic (δ) and a minimum area threshold (σ). We perform a grid search on 10 different thresholds for δ ranging from 0 to 0.04 (based on a study of the distribution of predicted edge probabilities) and with the following minimum areas for σ: 50, 100, 200, 300, 400, 500 pixels (roughly corresponding to objects with a real surface between 25 and 250 square meters — this could be set manually). The watershed does not bring any advantage over connected component labeling in the case of a binary EPM. Therefore, we do not report results for the **ConnNet scratch + Watershed** combination. Finally, large-image processing both for validation and test sets requires a tilling step: we first reconstruct the global images before running closed shape extraction to preserve the topological properties of the image and leverage a much larger spatial context than the one used by the deep edge detectors.

4.3 Metrics

To assess the performance of our system, we report the Panoptic Quality (PQ) values for each variant under test. The PQ metric [20] was introduced to simultaneously measure the detection, segmentation, and classification (hence the *panoptic* term) performance of a set of systems and rank them in the context COCO Panoptic challenge. In this work, we consider only one class of object. Wee do not make use of the multi-class capabilities of the metric and focus the joint detection and segmentation evaluation. The COCO Panoptic PQ indicator is computed as follows. First, for each pair of shapes (t_i, p_j) in the target

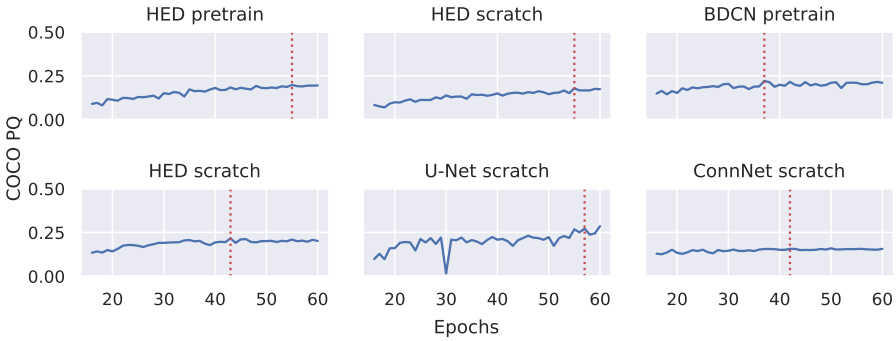

Fig. 3: COCO-PQ score of the networks under test for Stage 1 (Deep Edge Filtering), computed on the validation set, for each training epoch. The red vertical bar indicates the best model we retained to compute the results on the test set.

and in the predicted segmentation, the Jaccard index (or IoU) is computed as $IoU(t_i, p_j) = \frac{t_i \cap p_j}{t_i \cup p_j}$. Then, the set of uniquely matching pairs (or *true positives*) TP is defined as the set all pairs (t_i, p_j) such as $IoU(t_i, p_j) > 0.5$, leading to the definition of PQ:

$$PQ = \frac{\sum_{(t_i,p_j)\in TP} IoU(t_i, p_j)}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}, \tag{4}$$

where FP is the set of *false positives* (the set of *predicted* shapes which do not belong to any pair TP), and FN is the set of *false negatives* (the set of *target* shapes which do not belong to any pair in TP). While this measure summarizes the segmentation and the detection quality into a single indicator, two additional metrics provide additional insights: the Segmentation Quality (SQ) and the Recognition Quality (RQ) defined such as $PQ = SQ \times RQ$ where:

$$SQ = \frac{\sum_{(t_i,p_j)\in TP} IoU(t_i, p_j)}{|TP|}, \quad RQ = \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}. \tag{5}$$

4.4 Results and Discussion

The validation of the performance of each deep edge filter using the shape detection metric on the validation set enables the selection of the best model for each of the variants. Indeed, instead of selected the best model based on the validation loss used for the training, we adopt a more complex evaluation procedure, more representative of the final goal. We report in Fig. 3 the profile of the COCO PQ indicator during the training for each variant, and identify the best model retained. It is worth noting that BDCN and ConnNet seem to plateau quite rapidly (best model obtained around epoch 40), while HED and U-Net keep progressing (best model obtained close to the limit of 60 epochs).

	Stage 2: Connected Component Labeling						
	Parameters	Validation			Test		
Stage 1	θ	PQ	SQ	RQ	PQ	SQ	RQ
HED pretrain	0.7	27.6	77.6	35.6	**16.2**	76.1	21.3
HED scratch	0.3	23.2	76.5	30.3	14.0	74.8	18.8
BDCN pretrain	0.8	27.6	82.1	33.7	8.9	82.8	10.7
BDCN scratch	0.9	27.7	80.6	34.4	10.2	80.4	12.7
U-Net scratch	0.9	**34.8**	80.5	43.3	8.1	78.2	10.4
ConNet scratch	*none*				14.2	73.6	19.3

	Stage 2: Watershed							
	Parameters		Validation			Test		
Stage 1	δ	σ	PQ	SQ	RQ	PQ	SQ	RQ
HED pretrain	0.031	300	52.8	87.6	60.3	**38.4**	85.5	44.9
HED scratch	0.040	200	50.5	87.2	57.9	35.6	84.6	42.0
BDCN pretrain	0.027	300	53.0	88.1	60.1	37.8	86.4	43.8
BDCN scratch	0.031	200	52.5	87.8	59.8	34.9	85.8	40.6
U-Net scratch	0.000	50	**56.6**	87.7	64.5	18.3	85.2	21.4

Table 1: Global COCO Panoptic results (in %) of our evaluation, for each combination of deep edge detector (Stage 1) and closed shape extractor (Stage 2). Best results on validation and test sets are indicated **in bold**.

Then, using those best models for deep edge filtering, we are able to perform a grid search on the set of parameters for the baseline (CC labeling) and our proposed watershed module for closed shape extraction. The best parameters obtained are then used to compute the final performance of each combination of a deep edge filter with a closed shape extractor on the test set. Table 1 reports these results and the value of the best set of parameters for each combination. From these results we can draw several observations.

The first observation is that **the watershed constantly improves shape detection.** The joint ability of the watershed to filter out small noisy shapes, to recover weak edges, and to produce thin edges of one pixel explain the systematic gain obtained, no matter which deep edge filter is used. This very simple, yet effective technique, is a great complement to convolutional neural network and a key enabler in our application. This confirms the potential of the global pipeline we propose, i.e., combining a deep edge filter with a shape extraction using Mathematical Morphology tools.

The second observation is that **pre-trained HED is the best deep edge filter.** The Holistically Edge Detector outperforms the other architectures both when used with a naive closed shape extraction or with a more advanced one such as our watershed module. This seems to be mainly due to the lower complexity of this model compared to BDCN, U-Net and ConnNet; given a limited training

set, it generalizes better, and produces less strict edges (as seen in Fig. 4), which are easier to recover in the second stage.

This is highly related to the observation that **deep edge filters tend to overfit**. This is rather obvious that deep networks can easily overfit, but in our application, where annotation comes with a great cost (annotation took approximately 200 hours here), the resilience of HED is very valuable.

A further observation is that **transfer learning can improve results**: HED and BDCN benefit from features learned on natural images (ImageNet), when used with the watershed. Surprisingly, the performance is degraded for BDCN with plain component labeling for the pre-trained version.

This is unfortunate that no pre-training is available in the case of **U-Net which seems prone to overfit**. Despite reaching the best performance on the validation set, this filter performs poorly on the test set. The study of the output Edge Probability Map reveals very strong edges and equally strong cuts in them, as illustrated in Fig. 4: edges too clear and many gaps prevent a proper shape detection. Such edges cannot be recovered by the watershed, even with a dynamic threshold set to zero.

Finally, it should be noted that **the ConnNet architecture we introduced here for deep edge detection produces encouraging results.** Despite its heavier architecture, it reaches the second rank for the baseline shape detection while integrating the threshold calibration directly in its training process. Further work will be needed to enable its compatibility with the watershed.

5 Conclusion

We presented a learnable framework which enables the automatic vectorization of closed shaped from challenging historical maps. To our knowledge, this is the first approach which can accelerate manual annotation on such data. We leverage powerful convolutional neural networks as deep edge filters combined with watershed to effectively and efficiently detect closed shapes. We performed an extensive comparison on large images of several deep architectures which revealed the superiority of HED, and the relevance of the watershed transform we designed. The introduction of ConnNet architecture as a deep edge detector also opens some promising directions. Finally, our code, dataset, results and extra illustrations are freely available at https://github.com/soduco/ICDAR-2021-Vectorization.

References

1. Angulo, J., Serra, J.: Mathematical morphology in color spaces applied to the analysis of cartographic images. Proceedings of GEOPRO **3**, 59–66 (2003)
2. Arbelaez, P., Maire, M., Fowlkes, C., Malik, J.: Contour detection and hierarchical image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **33**(5), 898–916 (2010)
3. Badrinarayanan, V., Kendall, A., Cipolla, R.: Segnet: A deep convolutional encoder-decoder architecture for image segmentation. IEEE Trans. Pattern Anal. Mach. Intell. **39**(12), 2481–2495 (2017)

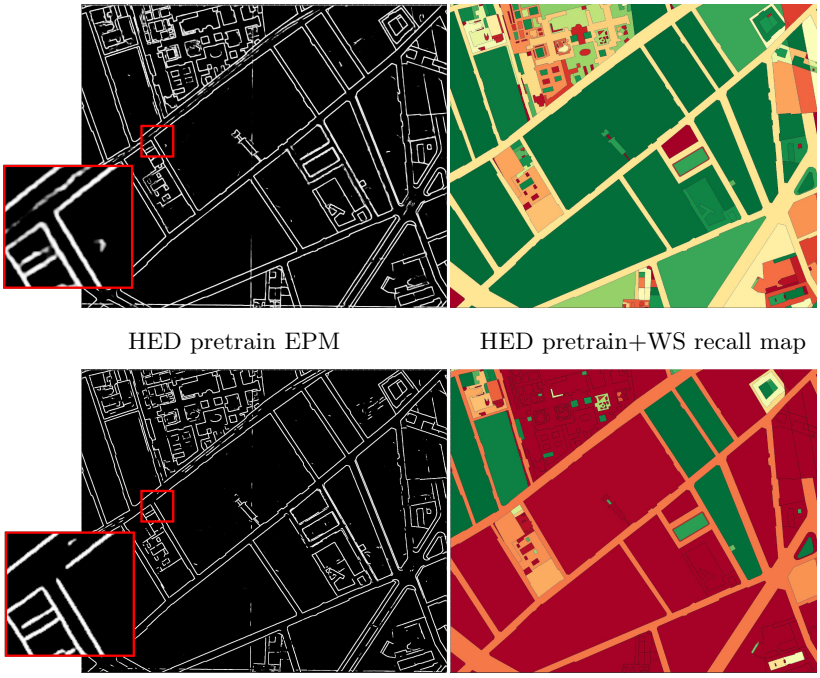

Fig. 4: Comparison of intermediate (EPM, *left*) and final shape prediction (recall map, *right*) results for pre-trained HED (best model, *top*) and U-Net. We can see on the left that many discontinuities in U-Net EPM prevent the detection of closed shaped, as indicated by zoomed part (red box). The recall map on the right indicates the quality of the segmentation of each shape – green shapes are correctly detected shapes and the red one are missed.

4. Bai, M., Urtasun, R.: Deep watershed transform for instance segmentation. In: Proc. of Conf. on Computer Vision and Pattern Recognition. pp. 5221–5229 (2017)
5. Budig, B., van Dijk, T.C., Feitsch, F., Arteaga, M.G.: Polygon consensus: smart crowdsourcing for extracting building footprints from historical maps. In: Proceedings of the 24th ACM SIGSPATIAL international conference on advances in geographic information systems. pp. 1–4 (2016)
6. Chen, K., Pang, J., Wang, J., Xiong, Y., Li, X., Sun, S., Feng, W., Liu, Z., Shi, J., Ouyang, W., et al.: Hybrid task cascade for instance segmentation. In: Proc. of Conf. on Computer Vision and Pattern Recognition. pp. 4974–4983 (2019)
7. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 834–848 (2017)
8. Chen, Y., Carlinet, E., Chazalon, J., Mallet, C., Duménieu, B., Perret, J.: Combining deep learning and mathematical morphology for historical map segmentation. In: Proc. of Int. Conf. on Discrete Geometry and Mathematical Morphology (DGMM) (2021), (accepted paper)

9. Chiang, Y.Y., Duan, W., Leyk, S., Uhl, J.H., Knoblock, C.A.: Historical map applications and processing technologies. In: Using Historical Maps in Scientific Studies, pp. 9–36. Springer (2020)
10. Chiang, Y.Y., Leyk, S., Knoblock, C.A.: Efficient and robust graphics recognition from historical maps. In: Intl. Workshop on Graphics Recognition. pp. 25–35. Springer (2011)
11. Chiang, Y.Y., Leyk, S., Knoblock, C.A.: A survey of digital map processing techniques. ACM Computing Surveys (CSUR) **47**(1), 1–44 (2014)
12. Couprie, M., Najman, L., Bertrand, G.: Quasi-linear algorithms for the topological watershed. J. Math. Imaging and Vision **22**(2), 231–249 (2005)
13. Cousty, J., Bertrand, G., Najman, L., Couprie, M.: Watershed cuts: Thinnings, shortest path forests, and topological watersheds. IEEE Trans. Pattern Anal. Mach. Intell. **32**(5), 925–939 (2009)
14. Dhar, D., Chanda, B.: Extraction and recognition of geographical features from paper maps. Int. J. Doc. Anal. Recogn. **8**, 890–904 (2006)
15. GDAL/OGR contributors: GDAL/OGR Geospatial Data Abstraction software Library. Open Source Geospatial Foundation (2021), https://gdal.org
16. Hanbury, A., Marcotegui, B.: Morphological segmentation on learned boundaries. Image Vision Comput. **27**(4), 480–488 (2009)
17. He, J., Zhang, S., Yang, M., Shan, Y., Huang, T.: BDCN: Bi-directional cascade network for perceptual edge detection. IEEE Trans. Pattern Anal. Mach. Intell. (2020)
18. Hilaire, X., Tombre, K.: Robust and accurate vectorization of line drawings. IEEE Trans. Pattern Anal. Mach. Intell. **28**(6), 890–904 (2006)
19. Kampffmeyer, M., Dong, N., Liang, X., Zhang, Y., Xing, E.P.: ConnNet: A long-range relation-aware pixel-connectivity network for salient segmentation. IEEE Trans. on Image Process. **28**(5), 2518–2529 (2018)
20. Kirillov, A., He, K., Girshick, R., Rother, C., Dollár, P.: Panoptic segmentation. In: Proc. of Conf. on Computer Vision and Pattern Recognition. pp. 9404–9413 (2019)
21. Leyk, S., Boesch, R.: Colors of the past: color image segmentation in historical topographic maps based on homogeneity. GeoInformatica **14**(1), 953–968 (2010)
22. Leyk, S., Boesch, R., Weibel, R.: Saliency and semantic processing: Extracting forest cover from historical topographic maps. Pattern Recognit. **39**(5), 953–968 (2006)
23. Liu, C., Wu, J., Kohli, P., Furukawa, Y.: Raster-to-vector: Revisiting floorplan transformation. In: Proc. of Int. Conf. of Computer Vision (ICCV) (2017)
24. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proc. of Conf. on Computer Vision and Pattern Recognition. pp. 3431–3440 (2015)
25. Martin, D., Fowlkes, C., Tal, D., Malik, J.: A database of human segmented natural images and its application to evaluating segmentation algorithms and measuring ecological statistics. In: Proc. of Int. Conf. of Computer Vision (ICCV). vol. 2, pp. 416–423. IEEE (2001)
26. Ostafin, K., Kaim, D., Siwek, T., Miklar, A.: Historical dataset of administrative units with social-economic attributes for austrian silesia 1837–1910. Scientific data **7**(1), 1–14 (2020)
27. Perret, J., Gribaudi, M., Barthelemy, M.: Roads and cities of 18th century france. Sci. Data **2**(1), 1–7 (2015)

28. Petitpierre, R.: Neural networks for semantic segmentation of historical city maps: Cross-cultural performance and the impact of figurative diversity. arXiv preprint arXiv:2101.12478 (2021)
29. Roerdink, J.B., Meijster, A.: The watershed transform: Definitions, algorithms and parallelization strategies. Fundamenta informaticae **41**(1, 2), 187–228 (2000)
30. Romera-Paredes, B., Torr, P.H.S.: Recurrent instance segmentation. In: Proc. of Eur. Conf. on Computer Vision (ECCV). pp. 312–329. Springer (2016)
31. Ronneberger, O., Fischer, P., Brox, T.: U-net: Convolutional networks for biomedical image segmentation. In: Proc. of Medical Image Computing and Computer Assisted Intervention (MICCAI). pp. 234–241 (2015)
32. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., et al.: Imagenet large scale visual recognition challenge. Int. J. Comput. Vision **115**(3), 211–252 (2015)
33. Simonyan, K., Zisserman, A.: Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556 (2014)
34. Southall, H., Aucott, P., Fleet, C., Pert, T., Stoner, M.: Gb1900: Engaging the public in very large scale gazetteer construction from the ordnance survey "county series" 1: 10,560 mapping of great britain. Journal of Map & Geography Libraries **13**(1), 7–28 (2017)
35. Xie, S., Tu, Z.: Holistically-nested edge detection. In: Proc. of Int. Conf. of Computer Vision (ICCV). pp. 1395–1403 (2015)
36. Yu, F., Koltun, V.: Multi-scale context aggregation by dilated convolutions. arXiv preprint arXiv:1511.07122 (2015)
37. Zhu, S.C., Guo, C.E., Wang, Y., Xu, Z.: What are textons? Int. J. Comput. Vision **62**(1), 121–143 (2005)