A Comparison of Approaches for Automated Text Extraction from Scholarly Figures

Falk Böschen[1] and Ansgar Scherp[1,2]

[1] Kiel University, Kiel, Germany
{fboe,asc}@informatik.uni-kiel.de
[2] ZBW - Leibniz Information Centre for Economics, Kiel, Germany
a.scherp@zbw.eu

Abstract. So far, there has not been a comparative evaluation of different approaches for text extraction from scholarly figures. In order to fill this gap, we have defined a generic pipeline for text extraction that abstracts from the existing approaches as documented in the literature. In this paper, we use this generic pipeline to systematically evaluate and compare 32 configurations for text extraction over four datasets of scholarly figures of different origin and characteristics. In total, our experiments have been run over more than 400 manually labeled figures. The experimental results show that the approach BS-4OS results in the best F-measure of 0.67 for the Text Location Detection and the best average Levenshtein Distance of 4.71 between the recognized text and the gold standard on all four datasets using the Ocropy OCR engine.

Keywords: Scholarly Figures · Text Extraction · Comparison

1 Introduction

Scholarly figures are data visualizations in scientific papers such as bar charts, line charts, and scatter plots [7]. Many researchers use a semi-supervised text extraction approach [6, 18]. However, semi-supervised approaches do not scale with the amount of scientific literature published today. Thus, unsupervised methods are needed to address the task of text extraction from scholarly figures. This task is challenging due to the heterogeneity in the appearances of the scholarly figures such as varying colors, font sizes, and text orientations. Nevertheless, extracting text from scholarly figures provides additional information that is not contained in the body text [4]. To the best of our knowledge, no comparison of the different approaches for text extraction from scholarly figures has been conducted so far.

Based on the related work, we have defined a generic pipeline of six sequential steps that abstracts from the various works on text extraction from scholarly figures. We have re-implemented and systematically evaluated the most relevant approaches for text extraction from scholarly figures as described in the literature. In total, 32 configurations of the generic pipeline have been investigated. Fig. 1 shows the pipeline and the investigated methods for each step. We assess each pipeline configuration with regard to the accuracy of the text location detection via precision, recall, and F1-measure. In addition, we evaluate

the text recognition quality using Levenshtein distance based on the evaluation methodology of the Born-Digital Image Track of the ICDAR Robust Reading Competition. In summary, the contributions of the paper are: (i) A systematic comparison of in total 32 configurations of a generic pipeline for text extraction from scholarly figures. Each configuration consists of a combination of six to nine methods from a total of 21 different methods that we have implemented and evaluated. (ii) We make available four manually labeled datasets of scholarly figures that allow reproducing and extending our results[3]. (iii) Furthermore, we make available the implementation of our generic pipeline which allows to use it on other datasets.

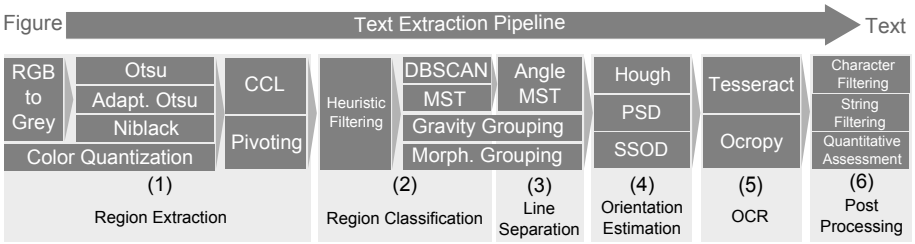

Fig. 1. Generic pipeline for text extraction from figures abstracted from the literature

2 Related Work

An early work on text extraction from scholarly figures is by Huang et al. [9]. They use connected component labeling and a series of filters to extract regions from the figure that represents text. Subsequently, text lines are found by using a derivation of Newton's formula for "Gravity" from classical physics and OCR is applied. Sas and Zolnierek [17] propose a three-stage approach for text extraction from figures. Their approach binarizes the figure, applies connected component labeling, and filters the extracted regions using pre-defined thresholds. Tesseract[4] is used for OCR and after a rotation of 90° the process is repeated. Finally, we have developed a pipeline called TX for unsupervised text extraction from scholarly figures [2, 3]. The pipeline combines an adaptive binarization method with connected component labeling and DBSCAN clustering to find text. A minimum spanning tree algorithm is used to estimate text lines followed by a Hough transformation for calculating the orientation, followed by OCR with Tesseract. A recent approach for semi-automatic text extraction from cartographic maps is proposed by Chiang et al. [6]. Cartographic maps use text

[3] `http://www.kd.informatik.uni-kiel.de/en/research/software/text-extraction`

[4] `https://github.com/tesseract-ocr/`

elements for city and street names, regions, and landmarks. In contrast to other approaches, Chiang et al. apply a color quantization algorithm and separate the text in the map from the rest using a semi-automatic extraction that requires a positive and a negative example for each text color. Text lines are detected using morphological operators and recognized using the commercial OCR engine AbbyyFineReader[5]. A text detection algorithm for biomedical images was proposed by Xu and Krauthammer [19] as part of the Yale Image Finder. Their pivoting algorithm uses vertical and horizontal histogram projection analysis to recursively split the image while classifying each region into text or non-text. Lu et al. [14] developed a retrieval engine for scholarly figures in chemistry. Their system works only on 2D plots and uses connected component labeling and fuzzy rules. Another approach for text extraction from figures by Jayant et al. [10] uses classic connected component labeling, support vector machines, minimum spanning trees, Adobe Photoshop (for preprocessing) as well as commercial OCR engines. They extract figures from books and their approach makes the assumptions that these figures have the same style throughout a book.

3 Pipeline Structure

Based on the discussion of the related work, we derived a generic pipeline for text extraction from scholarly figures as shown in Fig. 1. The pipeline consists of six steps and can be implemented through different methods, which are described below. This allows to create different configurations of the pipeline and to conduct a fair comparison of these configurations.

The first step of the pipeline takes a scholarly figure (color raster image) as input. The figure is converted into a binary image using either Color Quantization [5], by reducing the number of colors in an image and taking each resulting color channel as a separate binary image, or by converting the image to greyscale using the formula $Y = 0.2126R + 0.7152G + 0.0722B$ and subsequently applying a binarization method. For binarization, we use Otsu's Method [15], which finds the binarization threshold by maximizing the intra-class variance, Niblack's Method [13] which is often used for document image binarization, and Adaptive Otsu Binarization [3], which hierarchically applies Otsu's Method to adapt to local inhomogenities. The output of the first step is a set of regions, where each region is a set of connected pixels. They can be extracted using classic Connected Component Labeling (CCL) [16], which iterates over the pixel of an image and connects adjacent foreground pixel into regions. Another option is the Pivoting Histogram Projection method [19], which iteratively splits the binary image by analyzing the horizontal and vertical projection profiles. The second step takes these regions as input and computes a feature vector for each region, consisting of coordinates of the center of mass, dimension, and area occupation, to classify them into text or graphics. Heuristic Filtering [17] can be applied, prior to more complex algorithms, to preprocess the set of regions and remove outliers. The

[5] `http://www.abbyy.com/ocr-sdk/`

classification of the remaining regions can be achieved using clustering methods like DBSCAN [3], since text should be more dense in the feature space, or Minimum Spanning Tree (MST) clustering [10]. Other approaches are Grouping Rules based on Newtons Gravity Formula [9] from classical physics or the Morphological Method [6], which uses morphological operators to merge regions on pixel level. Subsequently, the generated sets of regions that are classified as text are fed into the third pipeline step to determine individual lines of text if necessary. For this step, we have only found one method in the literature, the Angle-Based MST clustering [3]. It computes a MST on the centers of mass of the regions and removes those edges that are not inside a predefined range of 60° around the main orientation. The fourth step of the pipeline computes the orientation for each text line using one of the following methods: The Hough Transformation [3] can be used on the centers of mass of a text line's regions to transform them into Hough space, where the maximal value determines the orientation. A different option is to minimize the Perpendicular Squared Distance of the bounding box of a text line to identify its orientation [10]. The third option is the Single String Orientation Detection algorithm [6] which determines the text line orientation using morphological operators. In the fifth step, existing OCR engines are used to recognize the horizontal text lines. We have evaluated the Tesseract OCR engine and Ocropy[6], since both are freely available, frequently updated, and allow to reproduce our results without limitations. We used the English language models that are provided by the OCR engines and we deactivated any kind of layout analysis. The recognized text is post-processed in the sixth and last step of the pipeline. Here, we apply either Special Character Filtering that removes all special characters from the text, since they often appear when text was incorrectly recognized, Special Character Filtering per String [17] that removes complete text lines, if they contain too many special characters, or Quantitative OCR Assessment [6]. The latter analyzes the difference between the number of characters (regions) that went into the OCR process and the number of recognized characters in order to decide whether to discard a text line.

4 Pipeline Configurations

From the methods defined in the previous section, one can create various pipeline configurations. Some methods are restricted in how they can be combined as illustrated in Fig. 1.

Configurations based on the discussion of the state-of-the-art in Section 2: Each of the seven configurations is identified by (x), an acronym created from the contributing author(s). The first configuration (SZ13) is inspired by the work of Sas and Zolnierek [17]. It uses Otsu's method for binarization, followed by CCL. Subsequently, it applies heuristic filtering similar to the original approach. The decision tree used by Sas and Zolnierek is replaced by the line generation approach based on MST. Since the original work by Sas and Zolnierek does not include a method for orientation estimation, we do not use any replacement in

[6] `https://github.com/tmbdev/ocropy`

step 4. Tesseract is used as OCR engine, since it was also used in the original paper. In the post processing step, all strings are removed that contain too many special characters.

The second configuration (Hu05) is based on the work of Huang et al. [9]. After region extraction using Otsu binarization and CCL, the Heuristic Filter method is applied, and the regions are grouped using the Gravity method. Finally, the grouped regions are processed with Tesseract.

Based on the work of Jayant et al. [10], the configuration (Ja07) starts with Otsu's method and CCL. Subsequently, it clusters the regions using a MST and approximates the orientation by minimizing the perpendicular squared distance. Text recognition is achieved by applying Tesseract.

Different from the previous configurations, the fourth configuration (CK15) – inspired by Chiang et al. [6] – uses Color Quantization to generate multiple binary images, followed by a CCL. Subsequently, it applies heuristic filtering and Morphological Clustering on the regions. This step differs from the original paper, where the relevant color levels were manually selected. Thus, we assess all extracted binary images. The orientation of each cluster is estimated using the SSOD method, followed by Tesseract OCR, and quantitative post-processing.

Similar to the previous pipeline configuration, the fifth configuration (Fr15), inspired by Fraz et al. [8], starts with Color Quantization and CCL. The original approach uses a supervised SVM to form words, which we replaced with unsupervised methods from our methods set. The extracted regions are filtered and DBSCAN is applied, followed by a MST clustering into text lines. The orientation of each text line is calculated using Hough method and the text is recognized using Tesseract.

All configurations so far use CCL to extract regions. The sixth configuration (XK10), motivated by Xu and Krauthammer [19], uses the pivoting algorithm after binarization with adaptive Otsu. The regions are filtered using heuristics and grouped into lines using DBSCAN and MST. This differs from the original work, which only applied heuristic filtering to remove the graphic regions. The reason behind this is that the authors only aimed at finding text regions and not to recognize the text. Thus, we filled the rest of the pipeline steps with suitable methods. The orientation of each line is estimated via Hough and OCR is conducted with Tesseract.

Finally, configuration (BS15) resembles our own work [3]. It uses adaptive Otsu for binarization and CCL for region extraction. Heuristic Filtering is applied on the regions and DBSCAN groups them into text elements. Text lines are generated using the angle-based MST approach and the orientation of each line is estimated via Hough transformation, before applying Tesseract's OCR.

Influence of individual methods: In order to evaluate the influence of the individual methods, we chose the pipeline configuration (BS15) as basis for systematical modification, since our evaluation showed that it produces the best results, as reported in Section 6. The systematic modifications are organized along the six steps of the generic pipeline in Fig. 1. Each of the systematic configurations has an identifier (BS-XYZ) based on the original configuration,

where X is a number that refers to the associated pipeline step and YZ uniquely identifies the method. The systematically modified configurations are described below. *Modifications of Step (1):* The binarization and region extraction is evaluated with the following configurations: (BS-1NC) differs from (BS15) by using Niblack instead of adaptive Otsu for binarization. Configuration (BS-1OC) uses the third option for binarization, Otsu's method. Color quantization is combined with the pivoting region extraction in (BS-1QP). *Modification over Steps (2) and (3):* The next step is the region classification and generation of text lines. Configuration (BS-2nF) differs from the base configuration by not applying the optional heuristic filtering method. Configuration (BS-2CG) uses the Gravity Grouping instead of DBSCAN and MST. Configuration (BS-2CM) applies MST to cluster regions and create text lines. Morphological text line generation is used in configuration (BS-23M). *Modifications of Step (4):* The following two configurations assess the methods for estimating the orientation of a text line: Configuration (BS-4OP) uses the Perpendicular Squared Distance method and configuration (BS-4OS) uses the Single String Orientation Detection method to estimate the orientation. *Modifications of Step (5):* For all configurations, both OCR engines are used to generate the results. The identifier of a configuration is extended to (BS-XYZ-T) or (BS-XYZ-O), when referencing the configurations that use Tesseract or Ocropy, respectively. Furthermore, we assess the direct impact of the OCR engine on the recognition results with configuration (BS15-O), which only differs with respect to the OCR method from the base configuration by using the Ocropy OCR engine instead of Tesseract. *Modifications of Step (6):* The last step of the pipeline is the post-processing. We use three configurations to evaluate the different post-processing methods: Configuration (BS-6PC) uses the Special Character Filter method for post-processing. Configuration (BS-6PS) uses the String Filter method for post-processing. Configuration (BS-6PQ) uses the Quantitative Assessment method for post-processing.

5 Evaluation

Datasets: We have used four datasets of varying origin and characteristics with in total 441 figures in our evaluation. We have created the **EconBiz** dataset, a corpus of 121 scholarly figures from the economics domain. We obtained these figures from a corpus of 288,000 open access publications from EconBiz[7] by extracting all images, filtering them by size and other constraints, and randomly selecting the subset of 121 figures. The dataset resembles a wide variety of scholarly figures from bar charts to maps. The figures were manually labeled to create the necessary gold standard information. We manually labeled the **DeGruyter** dataset as well, which comprises scholarly figures from books provided by DeGruyter[8] under a creative commons license[9]. We selected ten books, mostly from the chemistry domain, which contain figures with English text and selected 120

[7] `https://www.econbiz.de/`
[8] `http://www.degruyter.com/`
[9] `http://www.degruyter.com/dg/page/open-access-policy`

figures randomly from these books. The gold standard for these figures was created using the same tool which has been used for the creation of the EconBiz dataset. The Chart Image Dataset[10] consists of two subsets. The CHIME-R dataset comprises 115 real images that were collected on the Internet or scanned from paper. It has mostly bar charts and few pie charts and line charts. The gold standard was created by Yang Li [20]. The CHIME-S dataset consists of 85 synthetically generated images. This set mainly contains line charts and pie charts and few bar charts. The gold standard was created by Zhao Jiuzhou [11].

We have also looked at ImageNet, TREC, ImageClef and ICDAR datasets. But none of them can be used to evaluate the specific challenges of scholarly figures. They either do not have the necessary ground truth information about the contained text or the dataset does not consist of scholarly figures. But we adopted the evaluation scheme of the Born-Digital Images track of the ICDAR Robust Reading Competition (RRC) [12], which is described below.

Procedure: We have selected three measures to evaluate the pipeline configurations and compare their results. Our gold standard consists of text elements which represent single lines of text taken from a scholarly figure. Each text line consists of one or multiple words which are separated by blank space. Each word may consist of any combination of characters and numbers. Every text line is defined by a specific position, size, and orientation. Each pipeline configuration generates a set of text line elements as well. These text lines need to be matched to the gold standard. Since we do not have pixel information per character, we match the extraction results with the gold standard by using the bounding boxes. This is based on the first evaluation task of the ICDAR RRC and evaluates the text localization on text line level. We iterate over all text lines in the gold standard and take all matches that are above the so-called intersection threshold. Our matching procedure calculates the intersection area between all pairs of the pipeline output and gold standard text lines. If the intersection comprises at least ten percent of the combined area of both text elements, than it is considered a match. This reduces the error introduced through elements which are an incorrect match and only have a small overlap with the gold standard. But it still allows to handle text lines that are broken into multiple parts. We look at each gold standard element and take all elements from the pipeline as matches that are above the intersection threshold. Thus, a gold standard element can have multiple matching elements and an element from the pipeline can be assigned to multiple elements from the gold standard if it fulfills the matching constraint for each match. We have defined three measures to assess these matches. The first two measures analyze the text localization. The third measure compares the recognized text, similar to the word recognition task of the ICDAR RRC, although we compare text lines and not individual words. First, we evaluate how accurate the configurations are at the Text Location Detection.

If at least one match is found for an element from the gold standard set, it counts as a true positive, regardless of what text was recognized. If no match was found, it is considered as false negative. A false positive is an element from the

[10] `https://www.comp.nus.edu.sg/~tancl/ChartImageDataset.htm`

pipeline output which has no match. From these values, we compute precision, recall, and F1-measure. This measure is a binary evaluation and assesses only whether a match to an element exists or not. In addition, we report the Element Ratio (ER) which is the number of elements recognized by the pipeline divided by the number of elements in the gold standard and the Matched Element Ratio (MER) which is the number of matched items from the pipeline divided by the number of elements of the gold standard. These ratios give an idea whether gold standard elements get matched by multiple elements and whether the configuration tends to find more elements or less elements than it actually should find.

Second, we investigate the matching in more detail by assessing the Text Element Coverage. For each gold standard text element, we take the pixel of the bounding boxes and compute their overlap to calculate precision, recall, and F1-measure over all of its matches. The true positives in this case are the overlapping pixel and the false positives are those pixel from the text elements from the pipeline which are not overlapping. The false negatives are the pixels of the gold standard element which were not covered by a text element from the pipeline. The values are averaged over all gold standard text elements in a figure.

Third, we assess the Text Recognition Quality by computing the Levenshtein distance between the extracted text and the gold standard. We calculate the distance for each match and report the average for the whole figure. Since multiple text elements from the pipeline can be matched to a gold standard text line, we have to combine their text into one string. We combine the elements using their position information. Besides a (local) Levenshtein Distance per match, we also compute a global Levenshtein distance over all extracted text. This means that for each figure, we combine all characters from the text elements of the gold standard and add them to one string. Likewise, we create a string from the text elements extracted by the pipeline. The characters in both strings are sorted alphabetically and we compute the Levenshtein Distance between these strings. This approximates the overall number of operations needed to match the strings without considering position information. Since the global Levenshtein Distance depends on the number of characters inside a figure, we normalize it to an operations per character (OPC) score, which is computed by dividing the global Levenshtein Distance by the number of characters in the gold standard. This makes the results comparable across scholarly figures with different amounts of characters.

6 Results

We have executed all configurations listed in Section 4 over the datasets described in Section 5. For reasons of simplicity, we are only reporting the average values for Text Location Detection, Text Element Coverage, and Text Recognition Quality over all datasets. The detailed results per dataset can be found in our Technical Report [1]. We compute the average Precision/Recall/F1-measure

Table 1. Average Precision (Pr), Recall (Re), and F1 values for Text Location Detection and Text Element Coverage, Element Ratio (ER), and Matched Element Ratio (MER) over all datasets for configurations from the literature

	Text Location Detection					Text Element Coverage		
Config.	Pr	Re	$F1\ (SD)$	ER	MER	Pr	Re	$F1\ (SD)$
SZ13	0.63	0.47	0.54 (0.23)	0.80	0.59	0.52	**0.59**	0.47 (0.21)
Hu05	0.61	0.43	0.48 (0.28)	0.77	0.57	**0.79**	0.54	**0.57 (0.20)**
Ja07	0.59	0.45	0.49 (0.28)	0.83	0.51	0.41	0.32	0.32 (0.21)
BS15	0.66	**0.55**	**0.58 (0.25)**	**1.04**	0.69	0.60	0.49	0.50 (0.24)
CK15	0.52	0.50	0.53 (0.23)	1.37	0.60	0.53	0.41	0.42 (0.21)
Fr15	0.55	0.51	0.54 (0.25)	1.44	**0.72**	0.65	0.54	0.54 (0.23)
XK10	**0.73**	0.35	0.45 (0.26)	0.43	0.39	0.33	0.34	0.30 (0.22)

Table 2. Average local Levenshtein (L), global Levenshtein (G), and Operations Per Character (OPC) over all datasets for the configurations from the literature using Tesseract

Config.	$AVG_L(SD)$	$AVG_G(SD)$	OPC
SZ13	6.67 (4.82)	122.28 (141.03)	0.70
Hu05	6.65 (5.41)	126.35 (138.95)	0.71
Ja07	7.92 (5.56)	150.25 (140.59)	1.13
BS15	6.23 (4.93)	**108.81 (108.53)**	**0.67**
CK15	**6.07 (5.08)**	120.12 (125.87)	0.71
Fr15	6.72 (6.02)	135.64 (201.31)	0.85
XK10	7.06 (5.41)	125.45 (134.88)	0.74

over the elements of each figure. We report the average Precision/Recall/F1-measure in terms of mean and standard deviation over all figures. The local Levenshtein distance is reported as the average of the mean values per figure and the average standard deviation. The global Levenshtein distance is defined by the mean and standard deviation over all figures and the average of the normalized OPC score.

First, we report the results of the configurations from the literature. Subsequently, we present the results for the systematically modified configurations. The Text Location Detection and Text Element Coverage results for the configurations from the literature computed over all datasets are reported in Table 1. The best result, based on the F1-measure, is achieved by configuration (BS15) with a F1-measure of 0.58. The coverage assessment in Table 1 shows the best precision of 0.79 for (Hu05), the best recall of 0.59 for (SZ13), and the best F1-measure of 0.57 for (Hu05). The text recognition quality is presented in Table 2. We obtain the best results with (BS15) with 0.67 operations per character (OPC), an average global Levenshtein of 108.81, and an average local Levenshtein of 6.23. The best local Levenshtein of 6.07 is achieved by configuration (CK15). For the systematically modified configurations, Table 3 shows the

Table 3. Systematically modified configurations: Average Precision (Pr), Recall (Re), and F1 values for Text Location Detection and Text Element Coverage, Element Ratio (ER), and Matched Element Ratio (MER) over all datasets

	Text Location Detection					Text Element Coverage		
Config.	Pr	Re	$F1$ (SD)	ER	MER	Pr	Re	$F1$ (SD)
BS15	0.66	0.55	0.58 (0.25)	1.04	0.69	0.60	0.49	0.50 (0.24)
BS-1NC	0.64	0.52	0.57 (0.25)	0.96	0.64	0.59	0.44	0.47 (0.24)
BS-1OC	0.67	0.40	0.49 (0.26)	0.74	0.53	0.46	0.40	0.38 (0.26)
BS-1QP	0.61	0.44	0.48 (0.25)	0.96	0.75	0.41	0.57	0.42 (0.23)
BS-2nF	0.60	0.46	0.51 (0.23)	0.86	0.52	0.59	0.54	0.50 (0.21)
BS-2CG	0.62	0.50	0.55 (0.27)	0.90	0.64	0.76	0.54	0.57 (0.20)
BS-2CM	0.61	0.54	0.59 (0.25)	1.19	0.74	0.57	0.47	0.47 (0.24)
BS-23M	0.67	0.55	0.62 (0.23)	1.08	0.65	0.60	0.47	0.48 (0.22)
BS-4OP	0.62	0.53	0.57 (0.24)	**1.01**	0.66	0.49	0.40	0.41 (0.20)
BS-4OS	0.67	**0.63**	**0.67 (0.22)**	1.27	**0.88**	**0.77**	**0.63**	**0.65 (0.17)**
BS-6PC	**0.69**	0.54	0.59 (0.25)	0.97	0.70	0.59	0.49	0.49 (0.24)
BS-6PS	0.67	0.55	0.60 (0.25)	**1.01**	0.69	0.59	0.49	0.49 (0.24)
BS-6PQ	0.66	0.38	0.48 (0.25)	0.60	0.43	0.39	0.29	0.31 (0.21)

Text Location Detection results and the Text Element Coverage. Table 4 shows the Text Recognition Quality. The best location detection F1-measure of 0.67 is achieved by (BS-4OS), which is also supported by the coverage assessment with the highest F1-measure of 0.65. Configuration (BS-4OS-O) also produces the best text recognition results with an average local Levenshtein of 4.71 and an OPC of 0.53. In addition, configuration (BS-4OS-O) shows the best results of 95.49 for the average global Levenshtein Distance. Comparing the different, systematically modified configurations per step of the pipeline shows that the only major improvement is achieved by (BS-4OS). Please note, a performance analysis of the different configurations can be found in our Technical Report [1].

7 Discussion

Comparing the different configurations from the literature shows that the best performing configuration is (BS15). A possible reason is that our pipeline does not make many assumptions about the figures, e. g. figure type, font, or color. Thus performing better on the heterogeneous datasets. In the following, we will discuss the results for the individual pipeline steps based on the results from the systematically modified configurations. Comparing the configurations for the first pipeline step leads to the conclusion that the adaptive binarization works best, because it can adapt to local variations of the appearance in a figure. Otsu's method is too simple and Niblack's method is more suited for document images which have fewer color variations. The lower results for the pivoting algorithm can be explained with the larger regions and the possibility that a region can be a mixture of text and graphic elements due to the only horizontal and vertical

Table 4. Average local Levenshtein (L), global Levenshtein (G), and Operations Per Character (OPC) over all datasets for the systematic configurations

	Tesseract			Ocropy		
Config.	$AVG_L(SD)$	$AVG_G(SD)$	OPC	$AVG_L(SD)$	$AVG_G(SD)$	OPC
BS15	6.23 (4.93)	108.81 (108.53)	0.67	5.47 (4.98)	108.55 (106.64)	0.64
BS-1NC	6.27 (4.95)	117.58 (124.23)	0.69	5.70 (5.09)	117.46 (128.73)	0.66
BS-1OC	6.55 (5.06)	131.58 (142.74)	0.75	6.16 (5.21)	131.39 (143.16)	0.73
BS-1QP	8.31 (6.14)	154.54 (168.10)	1.09	7.06 (5.62)	136.40 (132.05)	0.82
BS-2nF	6.55 (4.94)	111.30 (105.13)	0.75	6.29 (5.50)	120.71 (109.18)	0.76
BS-2CG	6.68 (5.65)	108.86 (102.93)	0.66	6.22 (5.75)	130.21 (127.87)	0.69
BS-2CM	6.30 (5.29)	115.43 (113.79)	0.69	5.85 (5.34)	110.74 (107.23)	0.67
BS-23M	6.15 (5.12)	104.61 (105.97)	0.63	5.52 (5.10)	106.71 (104.05)	0.64
BS-4OP	8.30 (5.59)	147.91 (129.55)	1.04	7.23 (5.60)	135.21 (122.48)	0.85
BS-4OS	**5.47 (4.39)**	**96.29 (99.44)**	**0.58**	**4.71 (4.66)**	**95.49 (94.80)**	**0.53**
BS-6PC	5.96 (4.88)	105.50 (107.16)	0.61	5.46 (5.00)	109.07 (104.57)	0.63
BS-6PS	6.20 (4.90)	108.06 (109.38)	0.64	5.45 (4.96)	106.38 (103.29)	0.63
BS-6PQ	6.07 (5.03)	120.78 (122.44)	0.67	5.79 (4.97)	126.92 (124.06)	0.71

subdivision. Looking at step 2 and 3 of the pipeline, only the morphological clustering shows slightly better results than the DBSCAN-MST combination, most likely due to its processing on pixel level. The overall best results, when also considering the systematic configurations, are achieved by (BS-4OS). This can be explained by the fact that the orientation estimation via Hough works on the centers of mass of character regions, which is an aggregated region representation, while the SSOD in (BS-4OS) computes the orientation on the original pixels. Thus, it avoids a possible error, which could be induced by the pixel aggregation. When comparing the OCR engines from step 5, Ocropy generally produces better results than Tesseract. Ocropy seems to be more conservative, having built in much more restrictions about what input to accept and when to execute the OCR. Furthermore, each OCR engine comes with its own English language model and we did not evaluate their influence. The methods for post-processing do not improve the results. One reason might be the simplicity of methods. Thus, some more advanced techniques may be developed in the future. Overall, there are many more options for the different pipeline steps, e. g., other binarization methods, different clustering algorithms, or post-processing methods that could be used. However, we made a selection of relevant approaches and methods to limit the combinatorial complexity. On the other hand, as stated in the introduction, we provide the datasets and the implementation of the generic pipeline that was used in our experiment to the public. This allows for integrating and comparing new methods as well as the reproduction of our results.

Acknowledgement This research was co-financed by the EU H2020 project MOVING (`http://www.moving-project.eu/`) under contract no 693092.

References

1. Böschen, F., Scherp, A.: A Systematic Comparison of Different Approaches for Unsupervised Extraction of Text from Scholarly Figures [Extended Report]. Tech. Rep. 1607, Christian-Albrechts-Universität zu Kiel (2016), http://www.uni-kiel.de/journals/receive/jportal_jparticle_00000290
2. Böschen, F., Scherp, A.: Formalization and preliminary evaluation of a pipeline for text extraction from infographics. In: Bergmann, R., Görg, S., Müller, G. (eds.) LWA 2015 Workshop: KDML. pp. 20–31. CEUR (2015)
3. Böschen, F., Scherp, A.: Multi-oriented text extraction from information graphics. In: DocEng. pp. 35–38. ACM (2015)
4. Carberry, S., Elzer, S., Demir, S.: Information graphics: an untapped resource for digital libraries. In: SIGIR. pp. 581–588. ACM (2006)
5. Chiang, Y., Knoblock, C.A.: A general approach for extracting road vector data from raster maps. IJDAR 16(1), 55–81 (2013)
6. Chiang, Y., Knoblock, C.A.: Recognizing text in raster maps. GeoInformatica 19(1), 1–27 (2015)
7. Choudhury, S.R., Giles, C.L.: An architecture for information extraction from figures in digital libraries. In: WWW. pp. 667–672 (2015)
8. Fraz, M., Sarfraz, M.S., Edirisinghe, E.A.: Exploiting colour information for better scene text detection and recognition. IJDAR 18(2), 153–167 (2015)
9. Huang, W., Tan, C.L., Leow, W.K.: Associating text and graphics for scientific chart understanding. In: ICDAR. pp. 580–584. IEEE Computer Society (2005)
10. Jayant, C., Renzelmann, M., Wen, D., Krisnandi, S., Ladner, R.E., Comden, D.: Automated tactile graphics translation: in the field. In: ASSETS. pp. 75–82 (2007)
11. Jiuzhou, Z.: Creation of synthetic chart image database with ground truth. Honors year project report, National University of Singapore (2006), https://www.comp.nus.edu.sg/~tancl/ChartImageDatabase/Report_Zhaojiuzhou.pdf
12. Karatzas, D., Gomez-Bigorda, L., Nicolaou, A., Ghosh, S.K., Bagdanov, A.D., Iwamura, M., Matas, J., Neumann, L., Chandrasekhar, V.R., Lu, S., Shafait, F., Uchida, S., Valveny, E.: ICDAR 2015 competition on robust reading. In: ICDAR, August 23-26, 2015. pp. 1156–1160. IEEE Computer Society (2015)
13. Khurshid, K., Siddiqi, I., Faure, C., Vincent, N.: Comparison of Niblack inspired binarization methods for ancient documents. In: Document Recognition and Retrieval (DRR). pp. 1–10. SPIE (2009)
14. Lu, X., Kataria, S., Brouwer, W.J., Wang, J.Z., Mitra, P., Giles, C.L.: Automated analysis of images in documents for intelligent document search. IJDAR 12(2), 65–81 (2009)
15. Otsu, N.: A threshold selection method from gray-level histograms. TSMC 9(1), 62–66 (1979)
16. Samet, H., Tamminen, M.: Efficient component labeling of images of arbitrary dimension represented by linear bintrees. IEEE TPAMI 10(4), 579–586 (1988)
17. Sas, J., Zolnierek, A.: Three-Stage Method of Text Region Extraction from Diagram Raster Images. In: CORES. pp. 527–538. Springer (2013)
18. Savva, M., Kong, N., Chhajta, A., Fei-Fei, L., Agrawala, M., Heer, J.: ReVision: Automated Classification, Analysis and Redesign of Chart Images. In: UIST. pp. 393–402. ACM (2011)
19. Xu, S., Krauthammer, M.: A new pivoting and iterative text detection algorithm for biomedical images. Journal of Biomedical Informatics 43, 924–931 (2010)
20. Yang, L., Huang, W., Tan, C.L.: Semi-automatic ground truth generation for chart image recognition. In: DAS. pp. 324–335. LNCS, Springer (2006)