Ruling Analysis and Classification of Torn Documents

Markus Diem
Computer Vision Lab
Vienna University of Technology
Vienna, Austria
diem@caa.tuwien.ac.at

Florian Kleber
Computer Vision Lab
Vienna University of Technology
Vienna, Austria
kleber@caa.tuwien.ac.at

Robert Sablatnig
Computer Vision Lab
Vienna University of Technology
Vienna, Austria
sab@caa.tuwien.ac.at

ABSTRACT

A ruling classification is presented in this paper. In contrast to state-of-the-art methods which focus on ruling line removal, ruling lines are analyzed for document clustering in the context of document snippet reassembling. First, a background patch is extracted from a snippet at a position which minimizes the inscribed content. A novel Fourier feature is then computed on the image patch. The classification into *void*, *lined* and *checked* is carried out using Support Vector Machines. Finally, an accurate line localization is performed by means of projection profiles and robust line fitting. The ruling classification achieves an F-score of 0.987 evaluated on a dataset comprising real world document snippets. In addition the line removal was evaluated on a synthetically generated dataset where an F-score of 0.931 is achieved. This dataset is made publicly available so as to allow for benchmarking.

Categories and Subject Descriptors

I.4 [**Image Processing and Computer Vision**]: Miscellaneous; I.7.4 [**Document and Text Processing**]: Document Capture—*Document Analysis*

General Terms

Document Analysis; Ruling Analysis; Ruling Classification

1. INTRODUCTION

Ruling analysis is proposed for applications such as music score analysis [10], form analysis [18, 17] or handwriting processing [9, 3]. The former two applications detect ruling for further analysis or segmentation tasks while in the latter ruling is assumed to be noise, needed to be removed in a pre-processing step [1, 9]. If ruling lines in handwritten documents are considered, a few properties can be derived as presented by D. Lopresti and E. Kavallieratou [9]. Ruling lines have a uniform contrast, thickness, orientation and spacing within a page. They are – on purpose – brighter than the writing which overlaps with ruling.

DocEng'14, September 16–19, 2014, Fort Collins, Colorado, USA.

http://dx.doi.org/10.1145/2644866.2644876

Y. Zheng et al. [18] extract ruling lines in forms using Directional Single-Connected Chains (DSCC). They analyze white runs in binary images in order to detect lines. *Abnormal* run-lengths and crossings are removed so that lines which are merged with other foreground elements (e.g. text) get isolated. A dynamic length threshold, which is estimated based on the character size, rejects short lines such that lines of characters are not removed. This methodology is fast and able to extract clean lines present in document images without degrading foreground elements. However, broken ruling lines that occur due to the properties previously mentioned cannot be extracted using the DSCC. In order to overcome this drawback, Y. Zheng et al. [17] extend the algorithm. In this approach they again detect lines in the binary image using DSCCs. Then, a Hidden Markov Model HMM is trained on the document's Projection Profiles (PP) for accurate line positioning. Finally, lines are represented by polylines which account for non-rigid distortions present due to bad storage or digitization conditions.

W. Abd-Almageed et al. [1] present a ruling line removal for handwritten documents. For each pixel in the binary image features are computed within a local neighborhood based on central and statistical moments (standard deviation and kurtosis). The pixels are then classified into *line* or *non-line* pixels using a linear subspace. Though this algorithm has a potential for general line detection scenarios (no assumption about line parallelism or uniform spacing is incorporated), it tends to classify stroke endings of characters, ruling lines close to text, and noisy lines falsely.

D. Lopresti and E. Kavallieratou [9] present a ruling line removal based on a scanning approach. Lines are initialized in the left and right area of a page and the mean line slope and thickness are derived. Then, each column is scanned for pixel groups that have a similar thickness and are potentially on a line that was detected during initialization. The line slope and position is iteratively updated to account for quantization errors. Recently, J. Chen and D. Lopresti [3] present a ruling line detection for handwritten documents. A rough estimation of potential ruling lines is made by a simplified Hough transform. Then, broken line segments are clustered using Sequential Clustering. A multi-line linear regression finds a global optimum in the sense of Least Square Error (LSE) of ruling lines.

In contrast to state-of-the-art methods, this work focuses on ruling analysis for document clustering. The subjects of

investigation are manually torn Stasi records. Due to the number of snippets ($\approx$ 600 million), documents need to be grouped before reassembling so as to minimize the number of comparisons [11]. Beside text analysis [6], the ruling is analyzed for document grouping and – if present – accurate snippet alignment. The ruling analysis classifies snippets into *void*, *lined* and *checked*. In addition, it performs a line localization that can subsequently be used for snippet alignment and removal if ruling is present.

Figure 1 shows an example of each ruling class. For both ruled classes (*lined*, *checked*), the ruling is clearly visible and in c) the text does not cover the entire snippet which improves the ruling classification. In contrast to this, Figure 2 shows challenging examples. The first example (a,b) is *lined* but the horizontal lines have hardly any recognizable contrast. In b) the contrast is stretched for an improved illustration. Note that the histogram stretching additionally increases the noise and renders extruded pen strokes from other pages visible. The last two images (c,d) show a similar example with *checked* paper. Here, the histogram stretching d) shows scanning artifacts that can be traced back to the low contrast.

Figure 1: A snippet of each ruling class *void* a), *lined* b) and *checked* c).

In addition to the methodology proposed in this paper, a synthetic dataset is presented that can be used to empirically evaluate the performance of ruling line removal and compare different approaches. The dataset is based on images from the ICDAR 2013 Handwriting Segmentation Contest [12] and contains in total 600 handwritten images. The dataset together with an evaluation script can be downloaded from: `http://caa.tuwien.ac.at/cvl/research/ruling-database/`.

The remainder of the paper is organized as follows. Section 2 outlines the approach proposed including background localization, feature extraction, classification and ruling line localization. The method presented is evaluated in Section 3 with respect to the classification performance and the line localization. Finally, the paper is concluded in Section 4.

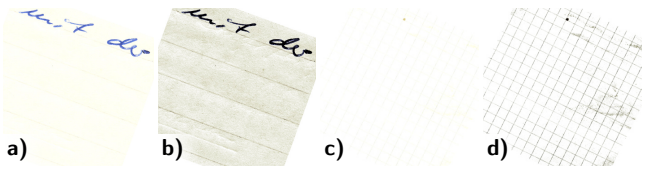

Figure 2: Challenging ruling examples. Low contrast ruling lines in a) and c). After a manual histogram stretch in b) and d) the ruling lines are visible with the ancillary effect of amplified noise.

2. METHODOLOGY

The ruling analysis can be divided into four modules (see Figure 3). First, a rectangle which maximizes background pixels, and thus minimizes the inscribed content, is estimated. Then, features are extracted from this patch. A classifier labels documents in one of the three classes mentioned previously. If ruling is present, an accurate ruling line estimation is performed. Depending on the application, ruling lines can be removed after this processing stage.

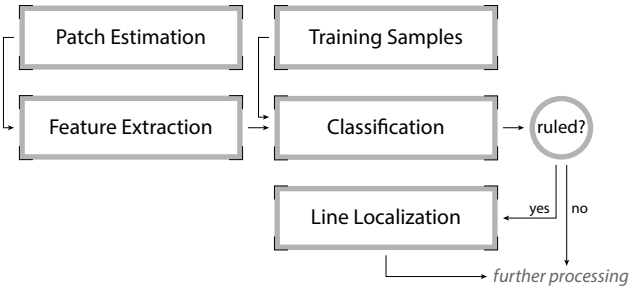

Figure 3: Ruling analysis processing pipeline proposed in this paper.

2.1 Patch Extraction

Ruling lines are repeated with a fixed line frequency. For classification, a homogeneous frequency is assumed for a whole snippet. In real world scenarios local frequency changes might occur because of scanning artifacts (paper is moved while scanning) or documents imperfectly pieced together. In order to speed up the computation, a patch of maximal 512×512 px is extracted. The reason for the patch size being 512 px is discussed in Section 3. An area filter is applied so that a patch with maximal background is detected in a page. The maximum of the filtered image indicates the region which contains most background pixels. Figure 4 shows a sample document a), its filtering result b) and the resulting image patch (blue rectangle). The area filter is visualized using pseudo color to increase its contrast.

A convolution using large kernels is slow if it is computed in the spatial domain (for area filtering a 512×512 px kernel is needed). Since an area filter is symmetric, two convolutions can be applied successively with a row and a column filter to speed up the computation. According to the convolution theorem [2], a convolution in the spatial domain is equal to a multiplication in the Discrete Fourier Transform (DFT) domain and vice versa. Hence, transforming an image to the DFT and back is faster than convolving it in the spatial domain if large kernels ($M > 256$) are used.

Because we are dealing with an area filter – which is a box filter – the convolution speed can be further improved using integral images which were first introduced for texture mapping in computer graphics [5] and later extended for general filtering tasks [15]. Hence, filtering using integral images is constant with respect to the kernel size. In order to filter images with this method, an integral image is first computed by summing each pixel with its left and lower pixel:

$$I_i(x, y) = I(x, y) + I(x, y-1) + I(x-1, y) \qquad (1)$$

where $I_i(x, y)$ is the resulting integral image of an image $I(x, y)$. In this way, each pixel value represents the area

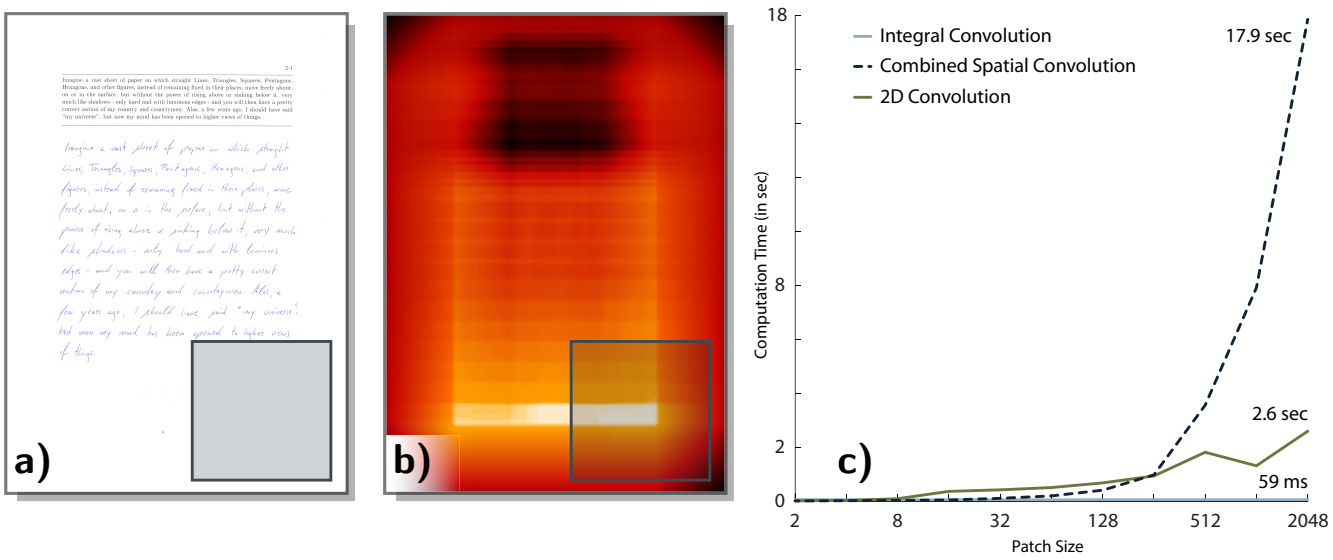

Figure 4: Sample from the CVL Database [8] with the resulting background patch a). Corresponding area filter b). Computation time of the three convolution techniques discussed c).

with respect to the coordinate origin. A drawback of this method is the memory usage, since the integral image needs to be allocated using 64 bit integers or floating points for 8 bit $N \times N$ images larger than $N = 2^{12} = 4096$. Having computed the integral images, box filter operations such as the mean or area filter are computed with 4 operations:

$$I_a(x, y) = p_3 - p_1 - p_2 + p_0 \tag{2}$$

where $p_i = I_i(x \pm \frac{M}{2}, y \pm \frac{M}{2})$

Figure 4 c) illustrates the computation time of the three convolution techniques presented. It was evaluated in a C++ environment on an Intel i7-3520M @2.9 GHz running Windows 8.1. Each operation was carried out 10 times as a trade off between total testing time and stability of the results. The integral image convolution takes 59 ms for a 2548×3510 px image independent to the kernel size. The spatial convolution with two 1D kernels is faster than the DFT convolution up to a kernel size of $M = 256$. The 2D filtering is performed in the spatial domain until a kernel size of $M = 16$. Then, the convolution is computed using the DFT transform. Though, the theoretical complexity is lower for integral images, the implementation is slower for small kernels $M < 8$ where the 1D convolution takes 32 ms.

2.2 Pre-Processing

The subsequent feature computations are carried out on the patch extracted previously. As the contrast between lines and background is low for some images (see Figure 2), the gradient magnitude is computed which emphasizes edges:

$$I_x(x, y) = I_g(x - 1, y) - I_g(x + 1, y)$$
$$I_y(x, y) = I_g(x, y - 1) - I_g(x, y + 1)$$
$$M(x, y) = \sqrt{(I_x(x, y))^2 + (I_y(x, y))^2}$$

with $M(x, y)$ being the resulting magnitude image and I_g the image patch smoothed by a Gaussian with $\sigma = 3$ (see Section 3.2). Foreground elements are removed by means of a local threshold [7] so that ruling lines are enhanced. A histogram stretch that maps values below the $Q_{99.5}$ Quantile linearly to [0 1] and sets values $x > Q_{99.5}$ to 1 further improves the ruling contrast. Figure 5 shows this enhancement for two documents a) and d). First, the gradients are computed in b) and e). Then, the foreground is removed and the histogram stretch is applied in c) and f). Especially f) shows the enhancement though the lines are noisy.

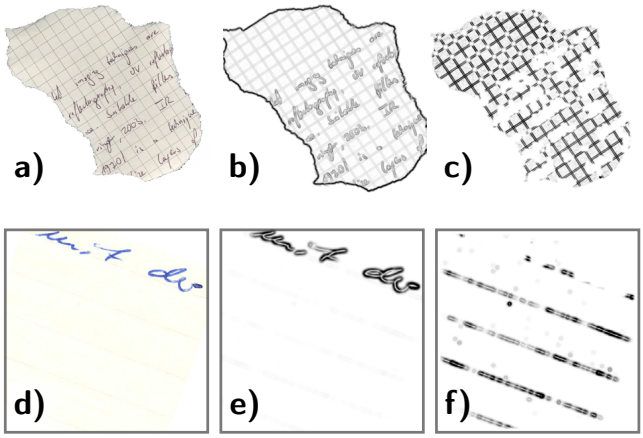

Figure 5: Two samples of *checked* a) and *lined* d) snippets. The gradient magnitude b), e) and the enhanced version c), f).

2.3 Texture Feature Extraction

Lines in the enhanced patches could be extracted using e.g. the Hough transform. However, ruling – compared to other lines present in documents – has two additional properties. First, ruling lines are parallel. Second, they have a fixed distance to their nearest neighbor. Since features with fixed (known) frequency and orientation can be easily extracted from the DFT, the patch is transformed to the DFT domain.

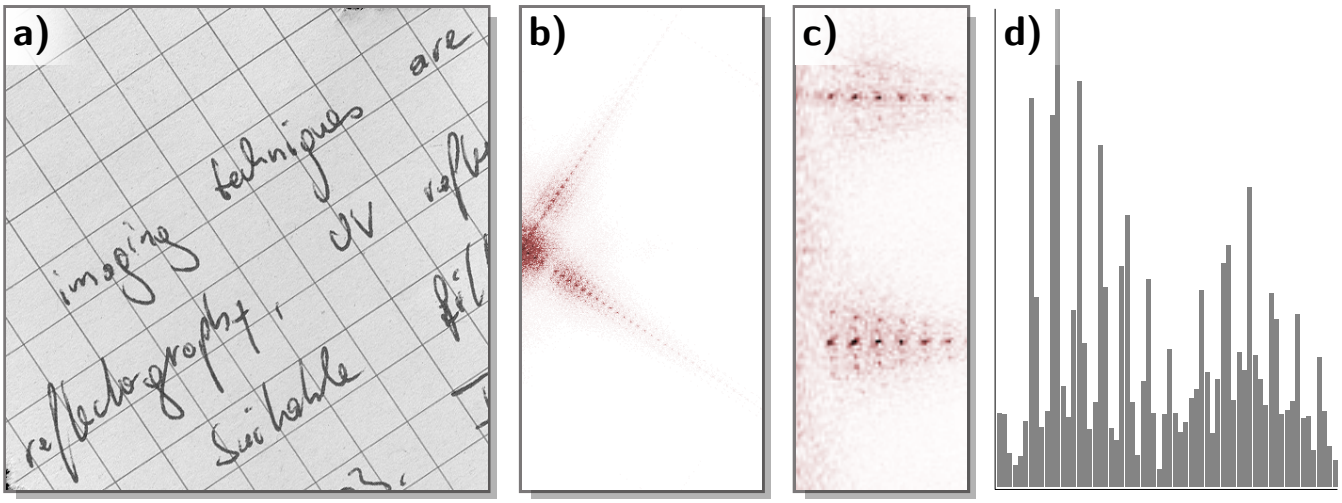

Figure 6: Feature extraction of a *checked* image patch. Power spectrum of the cleaned patch in b), polar transformed feature space c) and the final feature vector d).

The DFT is symmetric for real-valued inputs. Hence, solely half of the DFT image is used for subsequent computations. In addition, it is shifted by $N/2$ so that low frequencies are at the vertical center of the Fourier image rather than the border. The power spectrum combining real and imaginary part is computed for the texture features:

$$\mathcal{P}(x,y) = \sqrt{Re(I_{\mathcal{F}}(x,y))^2 + Im(I_{\mathcal{F}}(x,y))^2} \qquad (3)$$

where $I_{\mathcal{F}}(x,y)$ is the Fourier transformed image and $\mathcal{P}(x,y)$ the resulting power spectrum (see Figure 6 b)). The ruling orientation is unknown up to this processing stage. However, the features should be extracted rotationally invariant. That is why, the power spectrum is transformed using a polar transformation with a center located at the DFT's lowest frequencies $c = (0, N/2)$. By these means, the rows represent the changing angle θ and columns stand for the frequency. As can be seen in Figure 6 b), high frequencies (at the right image border) do not carry much information for these images. Thus, solely 64 columns are considered for the feature computation. By these means a polar-power spectrum (see Figure 6 c)) is extracted from the image patch. The dotted rows in c) show the features we are interested in. The horizontal distance between the peaks is $1/f$ of the ruling frequency f in the spatial domain.

Having extracted the polar-power spectrum, the ruling orientation is extracted using PPs. Since frequent peaks indicate the ruling, the maximum in the vertical PP is the ruling's main orientation. Skew issues of the PP do not apply for the polar-power spectrum, as the features are – by definition – perpendicular to the ordinate. The row at the global maximum $max(p)$ and the corresponding perpendicular row $mod(max(p) + 90, 180)$ are extracted for the final feature vector. Concatenating both rows results in a 128 dimensional feature vector which is normalized to [0 1] for robustness with respect to luminance changes. Instead of solely taking one row at each location, five rows are accumulated which improves the feature's profile if the ruling lines have a non-integer angle (e.g. 12.5°). Figure 6 d) shows a concatenated feature vector. This vector has random peaks if *void* paper is regarded, solely the first 64 dimensions have recurring peaks for *lined* paper and all 128 dimensions have peaks for *checked* paper.

2.4 Classification

The features are classified using Support Vector Machines (SVM) (the OpenCV[1] implementation is used) with a Radial Basis Function (RBF) kernel [13, 4]. The classifier and kernel selection is based on tests with empirical data which are presented in Section 3. The training set is comparatively small with 54 training images (18 per class). The training samples were randomly chosen from an annotated data set which consists in total of 436 document snippets. The advantage of a small training set is on the one hand that training is fast[2] and on the other hand a low manual effort which includes ground truth tagging. In contrast to the evaluation presented in this paper, the SVM is trained on all images of the dataset for real world recognition scenarios.

Since the SVM is – by definition – a binary classifier, the multi-class problem is reduced to three binary class decisions using a *one-versus-all* scheme. Therefore, three kernels are used where each corresponds to a class label *void*, *lined* or *checked*. Hence, the classifiers are trained with all 54 sample features where the labels of the currently observed class are set to 1 and all others are set to -1. Having trained the SVMs, the prediction value for a newly observed feature vector is computed for each kernel. A negative prediction value indicates that the feature does not belong to the class of the respective kernel while features with a positive value belong to the respective class. In addition to the class label, the prediction value which is the normalized distance between the observed feature vector and the hyperplane is used. Thus, features with low values are close to the hyperplane and therefore more likely to be wrong. The final class decision is the maximum of all three prediction values. Note that the class decision is made even if all kernels classify

[1] `http://opencv.org`

[2] training takes $\approx$ 23.1 seconds including the feature computation and cross validation

a feature as not belonging to their class. The 3×1 prediction value histogram is further used during re-assembling where a decision is made if the ruling feature is disregarded (depending on the likelihood of other features such as text classification labels).

In order to train the SVMs, a 3-fold cross validation is carried out for each kernel. Therefore, the training set is split into three data sets of equal size where two are used for training and one is used for testing. This procedure is repeated until each set was once used for testing. The SVM – if an RBF kernel is used – has two parameters (C, γ) which need to be optimized for new training sets. The cost $C > 0$ is the penalty of a falsely classified feature vector. Low cost values result in a low penalty. The second parameter $\gamma > 0$ controls the flexibility of the RBF kernel. Here, high values allow the hyperplane to be more *flexible*. Where *flexible* means that the hyperplane better fits the training data. The parameters are tuned for specific data using a parameter grid where each parameter is varied. The bounds are chosen to be $\exp(-5) \leq C \leq \exp(14)$ and $\exp(-14) \leq \gamma \leq \exp(1)$ which covers all variations present in the training set. The SVM is trained for each parameter tuple in the grid and evaluated using the 3-fold cross validation. The parameters which maximize the cross validation are chosen for training. Figure 7 shows the cross validation for each kernel. The gray dot indicates the final parameter tupel that maximizes the cross validation while minimizing the parameter values. Note that the a-priori class probability is the same for all classes since the number of samples is chosen to be equally distributed.

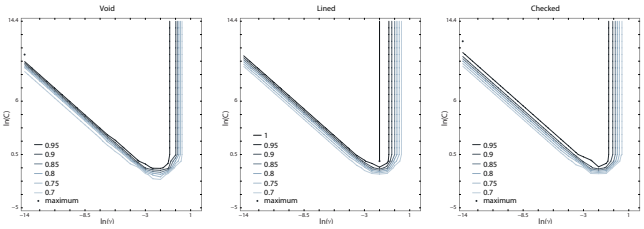

Figure 7: Cross validation on the training set having 18 samples per class.

2.5 Ruling Line Estimation

After classifying the document image into *void*, *lined* and *checked*, lines are extracted for images classified as *lined* or *checked*. The ruling lines are enhanced and writing is removed as described in Section 2.1. Then, the image is rotated according to the ruling angle which was detected in the polar-power spectrum. A vertical PP (for documents classified as *checked* a vertical and horizontal PP) is used in order to determine the accurate position of ruling lines. Local maxima p in the PP which have a stronger peak than 0.2 of the global maximum ($p \geq 0.2 \max(\text{PP})$) are used as initial guess.

It was previously mentioned, that ruling has a fixed frequency. Hence, the ruling frequency f_r is estimated by the $Q_{0.75}$ quartile of the distance between each line and its closest neighbor. Lines whose distance between the preceding or succeeding neighbor is not in the range of ± 10 *px* with respected to the frequency f_r are removed. Finally, missing lines are added with respect to the frequency. Figure 8 shows the ruling lines found of two sample snippets. Green lines are those detected in the PP while blue lines illustrate *virtual* lines that are estimated based on the ruling frequency.

The skew estimation in the polar transformed power spectrum is accurate enough for the initial line localization. However, slight deviations ($\pm 0.3°$) result in a degraded line removal. In order to overcome this, an image patch is extracted in the de-skewed background image. The height is set to 40 px while the width is the width of the currently observed line. All pixels whose vertical derivation is larger than 0.2 and horizontal deviation is below 0.2 are marked as line candidates. A robust line fitting using the Welsch distance [16] is performed on these pixels. If less than 40 line candidates are detected the line gets rejected. In addition, lines whose angle difference is larger than 0.8° with respect to the global line angle, are replaced with the initial guess.

Cleaning lines is performed by a logical AND operation of the lines found and the binary image. Overlapping text elements are then removed by means of Local Projection Profiles (LPP) which are perpendicular to the lines detected. An empirical evaluation of the LPP size showed that a kernel size of 9 px performs best.

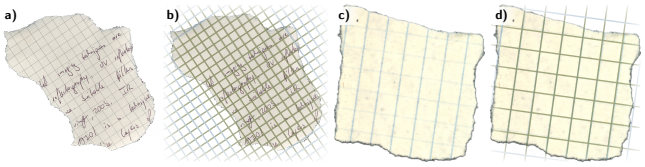

Figure 8: Ruling estimation of two sample snippets (a, c). Green lines are found in the PP while the location of blue lines is estimated using the ruling frequency.

3. EVALUATION

The ruling estimation was evaluated on a data set which has a total of 434 images. The data set is real world data from fragmented Stasi files scanned at 300 *dpi*. The data is particularly challenging because of its great variety. Thus, it comprises snippets with varying area, background and background clutter. The paper fragments have a mean area of 42.4 cm^2 with a standard deviation of ± 37.1 cm^2 where a DIN A4 page has 623.7 cm^2. The data set has an equal class distribution.

3.1 Classifier Evaluation

The SVM proposed previously for classification is compared with three frequently used classifiers, namely Naïve Bayes, Linear Discriminant Analysis (LDA) and k-NN. It can be seen in Figure 9 that the Bayes and the LDA perform significantly worse compared to e.g. SVMs. This can be attributed to the curse of dimensionality. As previously discussed, the training is carried out on solely 18 samples per class while the feature dimension is set to 128. The LDA and the Bayes classifier are both *generative models*. Hence, the model parameters – whose number depends on the dimension – need to be estimated from the training samples. In contrast, the VC theory, which is the basis of SVMs, utilizes the entropy of training samples rather than their dimension [14]. Thus,

SVMs can find good classification boundaries even if the feature space is sparse. The same applies to k-NN which is a *predictive model* and therefore an empirical loss is minimized rather than finding the model that generates the data.

Besides, evaluating different classifiers, three popular SVM kernels were tested. Figure 9 (left) shows the results. It can be seen that the polynomial kernel of 2^{nd} degree has the worst performance being 92.4%. The linear SVM kernel has a lower precision by 1.05% compared to the RBF. Hence, this is the best choice if speed is of importance as the classification can be carried out with one vector multiplication. Note that the classification is carried out once per document snippet and that the whole ruling estimation presented (including feature extraction) takes 141 *ms* to compute if a 2512×3510 px document is observed. That is why, the higher precision of the RBF kernel is chosen rather than the linear SVM.

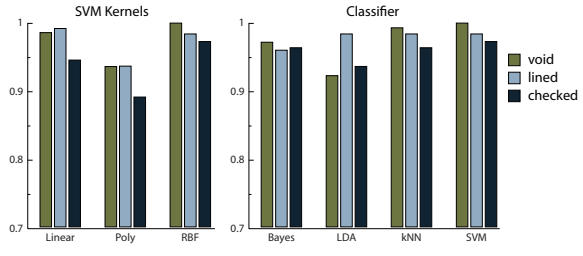

Figure 9: Evaluation with different SVM kernels (left) and classifiers (right).

Furthermore the classifier's weights are evaluated to show their use for subsequent processing steps such as the reassembling. Figure 10 shows a plot of the precision versus classification weights. The precision is accumulated for all classes with respect to the classification weights w. For small weights, the precisions are bending because of the small number of samples. As the number of samples increases ($w \approx 0.4$), the plot gets more stable and therefore more reliable. The bins represent the underlying sample distribution. Most snippets classified have a classification weight around one which hints at a reliable decision. The plot allows for choosing thresholds on which classification results are rejected when combining different features. In more detail, if someone would need a precision of at least 0.9, a classification weight $w \geq 0.43$ has to be chosen.

3.2 Empirical Parameter Evaluation

The previously discussed ruling estimation has four parameters which influence its behavior:

- **Sigma** σ the filter size of the Gaussian which is applied before computing the gradient magnitude.
- **Patch Size** of the patch with most background pixels where the DFT is applied.
- **Feature Dimension** of the features classified.
- **Interpolation Interval** for extracting the feature from the polar-power spectrum.

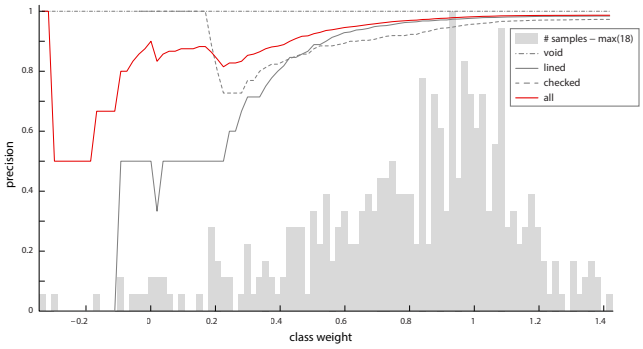

Figure 10: Classification weights versus precision. The precision is accumulated for each class with respect to the maximal class weight. It can be seen that the precision increases with increasing weights.

In order to find optimal values for these parameters, they are empirically evaluated. The Gaussian kernel size σ was evaluated with $1.5, 3$ and 4.5. Figure 11 (left) shows the evaluation. The best performance is achieved if it is set to $\sigma = 3$ which is a trade-off between removing too few noise but keeping the relevant line information. It can be seen additionally, that *lined* paper is hardly affected by σ. This can be traced back to the fact that *checked* paper has a higher frequency (lower distance between lines) and therefore peaks are merged if σ is too large. Choosing σ too low on the other hand leaves too much noise which results in a lower accuracy.

The *patch size* controls the window size which is used for observation. The maximum performance is achieved if it is set to 512 px as can be seen in Figure 11 (right). A too small patch (e.g. 128×128) has a negative effect on the recognition of *lined* paper because the mean line frequency is ≈ 106 px if documents are scanned at 300 *dpi*. Hence, a small patch captures solely one line which is too few for an accurate recognition. Large patch sizes on the other hand capture more foreground information and therefore increase the risk of an erroneous binarization.

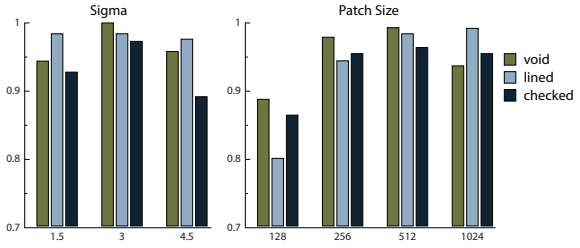

Figure 11: Evaluation of σ (left) and *patch size* (right).

A low *feature dimension* (see Figure 12 (left)) has a stronger effect on the ruling estimation (-4.7%) than a higher dimension (0.1%). Hence, the classifiers are capable of dealing with high dimensional features although the size of the training set is not increased. In addition, it can be argued from this test that a feature dimension of 64 is too low if the ruling is extracted from the DFT.

In contrast to the previously evaluated parameters, the *interpolation interval* has a comparatively low impact on the ruling estimation performance. Figure 12 (right) illustrates the test with varying parameters for the interpolation. An interval ≥ 1 removes noise that might be present in the line of the polar-power spectrum. Additionally, it improves the feature vector if the ruling orientation is not integer valued as it reduces interpolation artifacts from the polar transform.

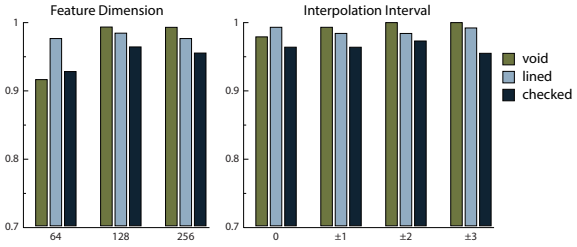

Figure 12: Evaluation of *feature dimension* and *interpolation interval*.

3.3 Dataset Evaluation

The performance of the ruling analysis presented is empirically evaluated on the data set previously introduced (Set A) and a second dataset Set B which is also presented in [6]. The first dataset Set A contains clean data with no class ambiguity while the second one contains ambiguous real world examples including e.g. tables or underlined text. The groundtruth of both datasets was manually annotated and saved to an Xml file. For each image, the ruling class (*void, lined, checked*) and the dominant orientation were annotated. First, a confusion matrix is given which shows the class confusion numerically. Table 1 shows how the confusion matrix can be interpreted. The rows indicate predictions while the columns represent the ground truth. Hence, diagonal elements are documents predicted as i^{th} class which are actually class i (true positives **tp**). All other values are false positives (**fp**) for the predicted class and false negatives (**fn**) for the true class. Table 2 shows the evaluation

Table 1: Confusion matrix with $n = 2$; a_i are predictions of class i, and c_i are the true class labels.

	a_0	a_1	$\cdots$	$\mathbf{a_i}$	$\cdots$	a_n
c_0				fp		
c_1				fp		
$\vdots$				$\vdots$		
$\mathbf{c_i}$	fn	fn	$\cdots$	tp	$\cdots$	fn
$\vdots$				$\vdots$		
c_n				fp		

results. It can be seen that using the methodology proposed with the previously evaluated parameters, all *void* classes are classified correctly. Two *lined* documents are falsely predicted as *void* and three *checked* documents are falsely classified as *lined*. Four out of the five errors can be attributed to a flawed foreground extraction. The error either arises from colored ink that is not binarized or binarized lines (see Figure 13 a),b)) that are removed in the magnitude image. Again, the magnitude images are inverted for illustration and therefore dark areas represent high gradient magnitudes. Figure 13 c), d) show a scanning artifact. In this scenario, the snippet was moved during scanning. Since the ruling lines are not parallel anymore, the proposed methodology cannot correctly classify this snippet.

Table 2: The rows of the confusion matrix show the groundtruth labels, while the columns represent predicted labels (e.g. 1.6% of the *lined* paper is falsely classified as *void*).

	predicted			
	void	lined	checked	#
void	**1.0**	·	·	143
lined	0.016	**0.984**	·	126
checked	·	0.027	**0.973**	111
	145	127	108	380

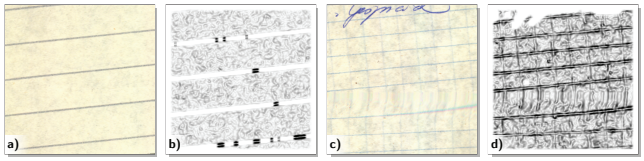

Figure 13: Two falsely classified examples.

In Table 3 the confusion matrix for Set B is given which was presented in [6]. In this dataset *void* document snippets have a precision of 92.4%. This can be traced back to the fact that tables such as table of contents are present which have recurring horizontal and vertical lines. The comparatively low precision of *lined* documents results from empty carbon copies where lines are – similarly to the example given in Figure 13 a) – falsely binarized as foreground elements and therefore removed from the magnitude image.

Table 3: The rows of the confusion matrix show the groundtruth labels of SET B, while the columns represent predicted labels.

	predicted			
	void	lined	checked	#
void	**0.924**	0.029	0.048	314
lined	0.116	**0.884**	·	103
checked	0.049	·	**0.951**	41
	304	100	54	458

Furthermore, F-score, precision, and recall are calculated for each class in order to allow for drawing conclusions about the nature of errors and class confusions. For these error measures, true positives tp_i, false positives fp_i, and false negatives fn_i of a given class i are defined by:

$$\begin{aligned} tp_i &\ldots \langle a_i, c_i \rangle \\ fp_i &\ldots \langle a_i, c_{j \neq i} \rangle \\ fn_i &\ldots \langle a_{j \neq i}, c_i \rangle \end{aligned}$$

where $i, j \in 0...n$ and $n = 2$ with $0 = \textit{void}$, $1 = \textit{lined}$ and $2 = \textit{checked}$. The precision p_i, recall r_i, and F-score F_i of a class i are thus defined as:

$$p_i = \frac{tp_i}{tp_i + fp_i}$$
$$r_i = \frac{tp_i}{tp_i + fn_i}$$
$$F_i = \frac{tp_i}{2tp_i + fp_i + fn_i}$$

Figure 14 shows the precision and recall for each class on Set A and Set B respectively. The recall is low for *checked* paper in Set B being 0.722. This can be attributed to the priors in Set B where *checked* paper has a prior of solely 8.9%. This prior reflects a small subset of the real world data where printed documents are more common than handwritten. However, the true a-priori probability of all classes is unknown. That is why a second test set (Set A) was created with similar priors for all classes that reduce the bias. The recall of *checked* paper is 1 in Set A since no other class is falsely predicted as *checked*. The F-score of Set A is 0.987 and 0.919 for Set B.

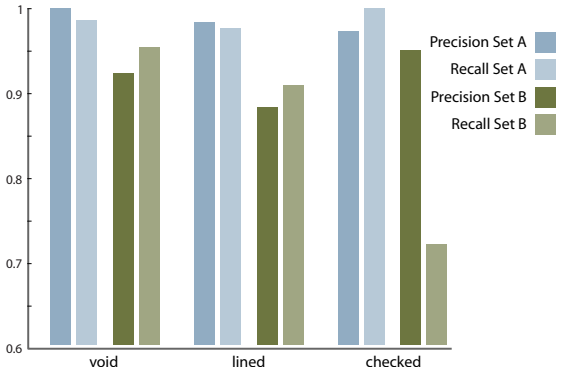

Figure 14: Precision recall plot of SET A and SET B.

3.4 Line Removal Evaluation

In addition to the classification evaluation, a line removal evaluation is performed. The evaluation is performed similar to that presented in [1, 9]. Therefore, a dataset is created by synthetically merging handwriting images from the Icdar 2013 Handwriting Segmentation Contest [12] with four different ruling masks. Figure 15 shows four examples of the ruling images created synthetically. The masks are extracted from scanned images and exhibit typical degradations such as broken lines. The Cvl ruling dataset is made publicly available[3] and consists of 150 images written in English, Greek and Bangla. This results in a total of 600 test images.

Figure 16 shows a sample page of the dataset. Green pixels indicate true positives (line pixels detected that are actually line pixels), red pixels illustrate false positives (line pixels detected that are actually text pixels) and black pixels are false negatives (true line pixels that are not detected). The gray pixels are for illustration reasons, they correspond – similar to white pixels – to the true negative class (pixels that are not line pixels and were not detected as such). Figure 16 shows that the Lpp is good at removing false line detections if strokes are perpendicular to the currently observed line (e.g. upper text in the zoomed area). However, it is not capable of correctly removing false positives if text strokes are parallel to a line (e.g. lower right text). It can be seen that the methodology presented is able to detected broken and noisy ruling lines.

[3] `http://caa.tuwien.ac.at/cvl/research/ruling-database/`

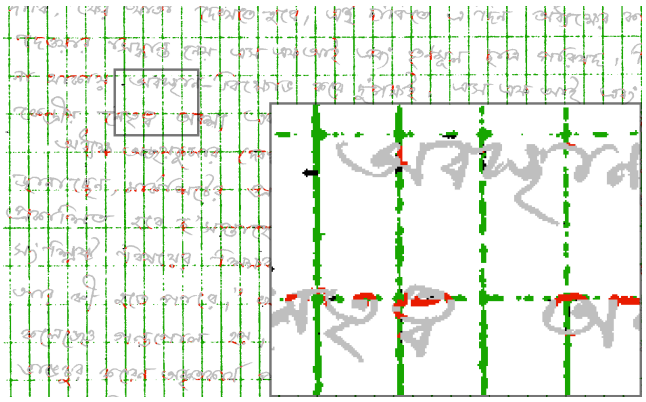

Figure 16: A sample page of the dataset presented. Green pixels are true positives, red pixels are false positives and black pixels are false negatives. Gray and white pixels correspond to true negatives.

The sole parameter that influences the quality of ruling line removal is the kernel size of the Lpp. The Receiver Operator curve (Roc) is given in Figure 17 when varying the kernel size between 0 (text restoration) and 15 with a step size of 2 px. The blue line shows the median recall versus precision when varying the kernel size. In addition the area within the $Q_{0.25}$ and $Q_{0.75}$ Quartiles is illustrated in gray. The maximal F-score of 0.93 is achieved with a kernel size of 9.

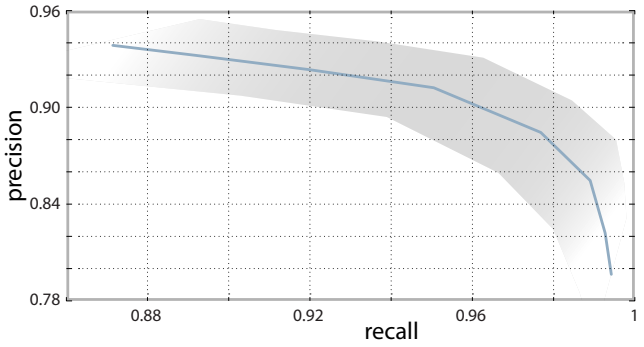

Figure 17: ROC curve when varying the kernel size of the LPP.

Since solely 9% of an image are line pixels (for checked paper), true negatives are not considered in the evaluation as this would bias the results. Abd-Amageed et al. [1] and Lopresti et al. [9] create synthetic datasets similar to the one proposed with 50 and 100 sample images respectively. Although, the dataset generation is similar and the error measurements are the same, the results presented in Table 4 cannot be directly compared since their datasets are not published.

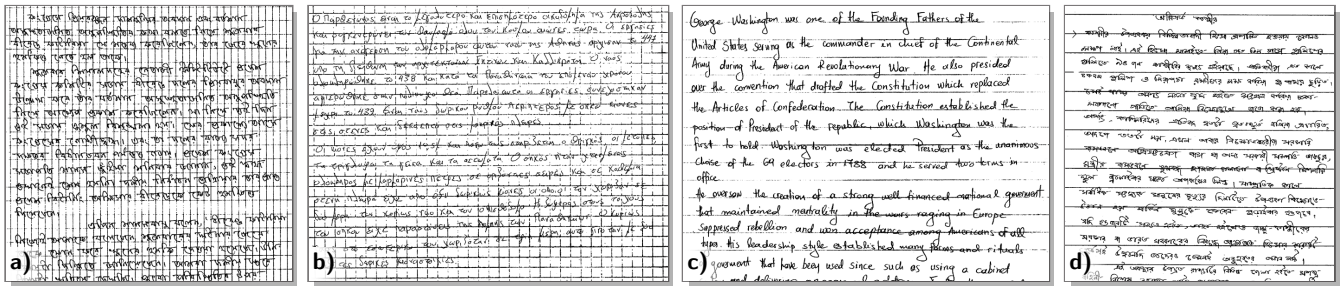

Figure 15: Four different samples of the synthetically generated line removal database.

Table 4: Line removal comparison. #I number of images, p precision, r recall and F F-score.

	#I	P	R	F
W. Abd-Amageed et al. [1]	50	0.88	0.88	0.88
D. Lopresti et al. [9]	100	0.76	0.91	0.81
proposed	600	0.91	0.95	0.93

4. CONCLUSION

A ruling analysis which is applicable for torn documents was presented in this paper. The ruling analysis first extracts a background patch which is enhanced by means of gradient magnitudes and histogram stretching. Furthermore, foreground elements are removed for an improved frequency signature in the DFT. A polar power spectrum allows for feature extraction invariant with respect to rotation. Changing ruling frequencies are trained by means of SVMs with a one-versus-all classification scheme. The evaluation showed that the designed workflow is capable of recognizing different paper types including *void*, *lined* and *checked* paper. Since the features are able to separate all classes cleanly, training can be carried out on a relatively small training set (18 samples per class) which reduces the human effort. The evaluation set presented include small document snippets which shows the method's robustness with respect to reduced information (e.g. short lines). The evaluation showed, that the foreground extraction (binarization) is crucial with respect to the precision. Hence, foreground elements missed degrade the contrast enhancement and therefore reduce the feature's signature. In addition, the binarization detects strong ruling lines if a snippet has no written content. As shown in Figure 13 c), the method fails if ruling lines in the patch are not parallel to each other. This is inherent with the design of the method, as snippets might be composed of several lines that are not equidistant or parallel to each other. By definition these snippets must not be classified as *lined* or *checked*. In addition, the ruling line removal quality was evaluated on a synthetically generated dataset which is freely available[4]. The precision was increased if the overlapping of text and ruling lines was modeled rather than applying LPPs on the cleaned text image. The second dataset SET B pointed out that ruling is ambiguous if formal criteria are considered. Structured elements such as tables have the same properties as ruling if they are composed of equidistant parallel lines.

[4] `http://caa.tuwien.ac.at/cvl/research/ruling-database/`

Acknowledgment

The authors would like to thank Dirk Pöhler, Jan Schneider and the Fraunhofer-Institute for Production Systems and Design Technology (IPK), Berlin for supporting the work.

5. REFERENCES

[1] Wael Abd-Almageed, Jayant Kumar, and David S. Doermann. Page Rule-Line Removal Using Linear Subspaces in Monochromatic Handwritten Arabic Documents. In *Proceedings of the 10th International Conference on Document Analysis and Recognition*, pages 768–772. IEEE Computer Society, 2009.

[2] George B. Arfken, Hans J. Weber, and Frank E. Harris. *Mathematical Methods for Physicists, Sixth Edition: A Comprehensive Guide*. Academic Press, 6 edition, July 2005.

[3] Jin Chen and Daniel P. Lopresti. Model-based ruling line detection in noisy handwritten documents. *Pattern Recognition Letters*, 35:34–45, 2014.

[4] Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

[5] Franklin C. Crow. Summed-area Tables for Texture Mapping. *SIGGRAPH Comput. Graph.*, 18(3):207–212, January 1984.

[6] Markus Diem, Florian Kleber, Stefan Fiel, and Robert Sablatnig. Semi-Automated Document Image Clustering and Retrie. In *Document Recognition and Retrieval*, 2014.

[7] Florian Kleber, Markus Diem, and Robert Sablatnig. Scale Space Binarization Using Edge Information Weighted by a Foreground Estimation. In *Proceedings of the 11th International Conference on Document Analysis and Reconstruction (ICDAR 2011)*, pages 854–858, Beijing, China, 2011. IEEE Computer Society CPS.

[8] Florian Kleber, Stefan Fiel, Markus Diem, and Robert Sablatnig. CVL-DataBase: An Off-Line Database for Writer Retrieval, Writer Identification and Word Spotting. In *Proceedings of the 12th International Conference on Document Analysis and Recognition*, pages 560–564, 2013.

[9] Daniel P. Lopresti and Ergina Kavallieratou. Ruling Line Removal in Handwritten Page Images. In *Proceedings of the 20th International Conference on Pattern Recognition*, pages 2704–2707, 2010.

[10] Ana Rebelo and Jaime S. Cardoso. Staff Line Detection and Removal in the Grayscale Domain. In *Proceedings of the 12th International Conference on*

Document Analysis and Recognition, pages 57–61, 2013.

[11] Jan Schneider and Bertram Nickolay. The Stasi puzzle. *Fraunhofer Magazine, Special Issue*, 1:32–33, 2008.

[12] Nikolaos Stamatopoulos, Basilis Gatos, Georgios Louloudis, Umapada Pal, and Alireza Alaei. ICDAR 2013 Handwriting Segmentation Contest. In *Proceedings of the 12th International Conference on Document Analysis and Recognition*, pages 1402–1406, 2013.

[13] Vladimir Vapnik. *Estimation of Dependences Based on Empirical Data: Springer Series in Statistics (Springer Series in Statistics)*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 1982.

[14] Vladimir Vapnik. *Estimation of Dependences Based on Empirical Data*. Springer Science and Business Media, Inc., New York, USA, 2006.

[15] Paul A. Viola and Michael J. Jones. Robust Real-Time Face Detection. *International Journal of Computer Vision*, 57(2):137–154, 2004.

[16] Roy E. Welsch and Edwin Kuh. Linear Regression Diagnostics. Technical Report 923-77, Massachusetts Institute of Technology, April 1977.

[17] Yefeng Zheng, Huiping Li, and David S. Doermann. A Parallel-Line Detection Algorithm Based on HMM Decoding. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(5):777–792, 2005.

[18] Yefeng Zheng, Changsong Liu, Xiaoqing Ding, and Shiyan Pan. Form Frame Line Detection with Directional Single-Connected Chain. In *Proceedings of the 6th International Conference on Document*