Manuscript Text Line Detection and Segmentation using Second-Order Derivatives

David Aldavert and Marçal Rusiñol
Computer Vision Center (CVC), Dept. Ciències de la Computació
Edifici O, Universitat Autònoma de Barcelona
08193 Bellaterra (Barcelona), Spain
{*aldavert,marcal*}*@cvc.uab.cat*

***Abstract*—In this paper, we explore the use of second-order derivatives to detect text lines on handwritten document images. Taking advantage that the second derivative gives a minimum response when a dark linear element over a bright background has the same orientation as the filter, we use this operator to create a map with the local orientation and strength of putative text lines in the document. Then, we detect line segments by selecting and merging the filter responses that have a similar orientation and scale. Finally, text lines are found by merging the segments that are within the same text region. The proposed segmentation algorithm, is learning-free while showing a performance similar to the state of the art methods in publicly available datasets.**

***Keywords*-text line detection; text line segmentation; text region detection; second-order derivatives;**

I. Introduction

The history of our ancestors is locked in libraries and archives within the vast volumes of preserved historic manuscripts. The accessibility and dissemination of such cultural assets will provide an important impact to our society. However, manually inspecting such huge collections for extracting relevant data is unfeasible and that is why automatic tools for processing such historic data have a paramount importance. But before applying any content extraction method, there is usually a pre-processing step of layout analysis of document images that is needed.

Within the different layout analysis tasks, text line segmentation is one of the pillar stages. Subsequent recognition methods depend on a proper text-line segmentation step. Although text line extraction is somehow an easy step for modern typewritten documents, where a simple algorithm do already perform perfectly, it is not the case for historic handwritten documents. Document degradation, the lack of layout regularity, variability in handwriting styles, text skew and text elements, ascenders and descenders, touching other text lines makes the problem much more difficult.

A proper segmentation of the different text lines that appear within historic manuscripts is a critical point for many document image recognition applications, either because the subsequent algorithms work at line level, or because they can benefit from information extracted from such process such as baseline position, text height, ascenders and descenders localization, etc. Such applications range from document deskewing [1], handwriting recognition[2], [3], [4], or keyword spotting [5].

In the literature, there are many text line segmentation algorithms. The two classical approaches are either projection profile-based or rely on the Hough transform. The projection profile approach [6], [7] is based on projecting the document pixels which are relevant (detected through binarization for example) into the Y-axis of the image. Then, the location of the text lines corresponds to the local maxima of the projection histogram. This method requires some pre-processing to ensure that the text lines are aligned with the Y-axis of the image, otherwise the detection is not possible. On the other hand, a different classical approach is to base the text line detection on the use of the Hough transform [8]. These approaches use the Hough transform to find the parameters of the visual linear structures produced by the text in the document. Unlike projection-profile approaches, these approaches can detect text lines which have a different orientation than the dominant one of the text. Both of those approaches process the image in a holistic fashion which might be problematic when the document layout does not follow a classic Manhattan structure, such as when the text is distributed into different columns or does contain tabular structures. Therefore, other approaches try to obtain line information at local level. Some methods group the pixels belonging to the same character into regions and then try to group them into lines. For instance, Cruz and Ramos-Terrades use in [9] the regions detected via connected components analysis to use an expectation-maximization algorithm to detect the text lines. However, connected component-based methods are not particularly interesting when nearby text lines touch each other because of ascender and descender artifacts. Other methods, create an energy map which corresponds to the distribution of the elements over the image and the try segment the lines over this energy map [10], [11] with seam carving-like approaches. Recently, some authors use a learning based methods to detect image regions where text lines are likely to appear [12], [13]. This approach is more robust to image transformations and noise than binarization or filter based methods. However, it has the drawback that a large amount of information is needed to properly train the classifiers. For a detailed explanation of the different approaches used for text line segmentation, we refer the reader to the survey [14].

In this paper, we use the second-order Gaussian derivatives to find an estimation of the dominant orientation

at each pixel. By filtering the image at different scales, we are able to locate also the characteristic scale of the text line, as the second-order Gaussian derivatives give a stronger response at the scale where the blurred text lines is closer to the σ of the Gaussian. Then, these local estimations are accumulated into a histogram to determine which are the most common orientation and scale of the elements present in the image. The pixels belonging to these distributions are processed independently and grouped first into text regions and later text lines. Finally, we use the binarized components within the text region to determine the line segmentation. The main contributions of this paper are twofold: First, we show that the second-order Gaussian filter can be used as a local operator to determine the orientation and scale of the text lines. Second, we propose a method to group the Gaussian filter estimates into text lines. Although the proposed method is quite straightforward, the obtained results are promising.

The paper is structured as follows: in Sect. II we present the method used to obtain a pixel-wise estimation of the orientation and scale. Then, in Sect. III we show how the pixel-level estimations are grouped into text lines. Finally, in Sect. IV we show qualitative results of the obtained segmentation and in Sect. V, we discuss the main contributions of the paper.

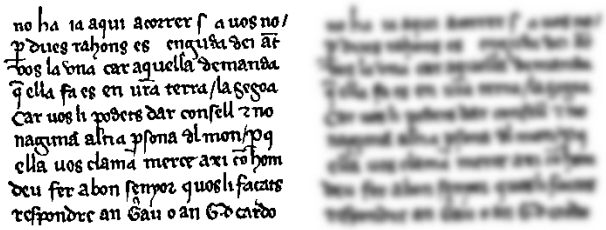
II. Second Order Derivative Analysis

We analyze the output of the second-order Gaussian derivatives of the image to extract which is the orientation and characteristic scale of the text lines present at each pixel of the image. First, we are going to review how we compute the second-order Gaussian derivatives and extract the line orientation efficiently at a given scale. Then, we are going to show how this approach is used at multiple scales to retrieve also the characteristic scales of the text elements in the document.

In order to achieve a certain degree of invariance to illumination and degradation conditions of the document, we first binarize the images with the Sauvola and Pietikäinen algorithm [15]. Therefore, from now on, all document images are supposed to be binary images.

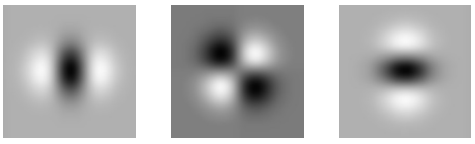
A. Second Order Derivative

In order to locally estimate the orientation of the text line, we want a filter that yields strong responses at the locations where a text line is present. Therefore, the oriented second-derivative Gaussian function is a good choice, since this operator has strong responses over lines. Since text lines resemble a line when blurred by a Gaussian with a large enough σ, we expect that this operator will have a strong response when the appropriate σ is used. For example, in Fig. 1 the text lines appear as blobs when they are filtered with a Gaussian filter with $\sigma = 12$. Although we can expect that other operators like Gabor or anisotropic Gaussian filters would give a better response than the second-order derivative, this filter has the advantage that it is steerable, i.e. the response of the filter at any given orientation can be calculated as the



Binary Blurred.

Figure 1: Blurring the binary document with a large enough Gaussian generates an image where text lines appear as blobs.



$\frac{\partial}{\partial x^2}\mathbf{G}(x,y)$ $\frac{\partial}{\partial x \partial y}\mathbf{G}(x,y)$ $\frac{\partial}{\partial y^2}\mathbf{G}(x,y)$

Figure 2: Bases used to compute the second order Gaussian derivative at any orientation.

combination of base filters. This is a well known property for the first-order Gaussian derivatives, where

$$\mathbf{G}'(i,\theta) = \cos(\theta)\frac{\partial}{\partial x}\mathbf{G}(i) + \sin(\theta)\frac{\partial}{\partial y}\mathbf{G}(i).$$

For higher-order derivatives, Freeman and Adelson present in [16] a method to select the minimum set of bases that better represent the given Gaussian filter. For the second-order Gaussian derivative, they show that it can be computed as a function of $\mathbf{C}_1(i) = 0.921\ \partial/\partial x^2\mathbf{G}(i)$, $\mathbf{C}_2(i) = 1.843\ \partial/\partial x\partial y\mathbf{G}(i)$ and $\mathbf{C}_3(i) = 0.921\ \partial/\partial y^2\mathbf{G}(i)$ (see Fig. 2):

$$\mathbf{G}''(i,\theta) = \cos^2(\theta)\mathbf{C}_1(i) - \cos(\theta)\sin(\theta)\mathbf{C}_2(i) + \sin^2(\theta)\mathbf{C}_3(i). \quad (1)$$

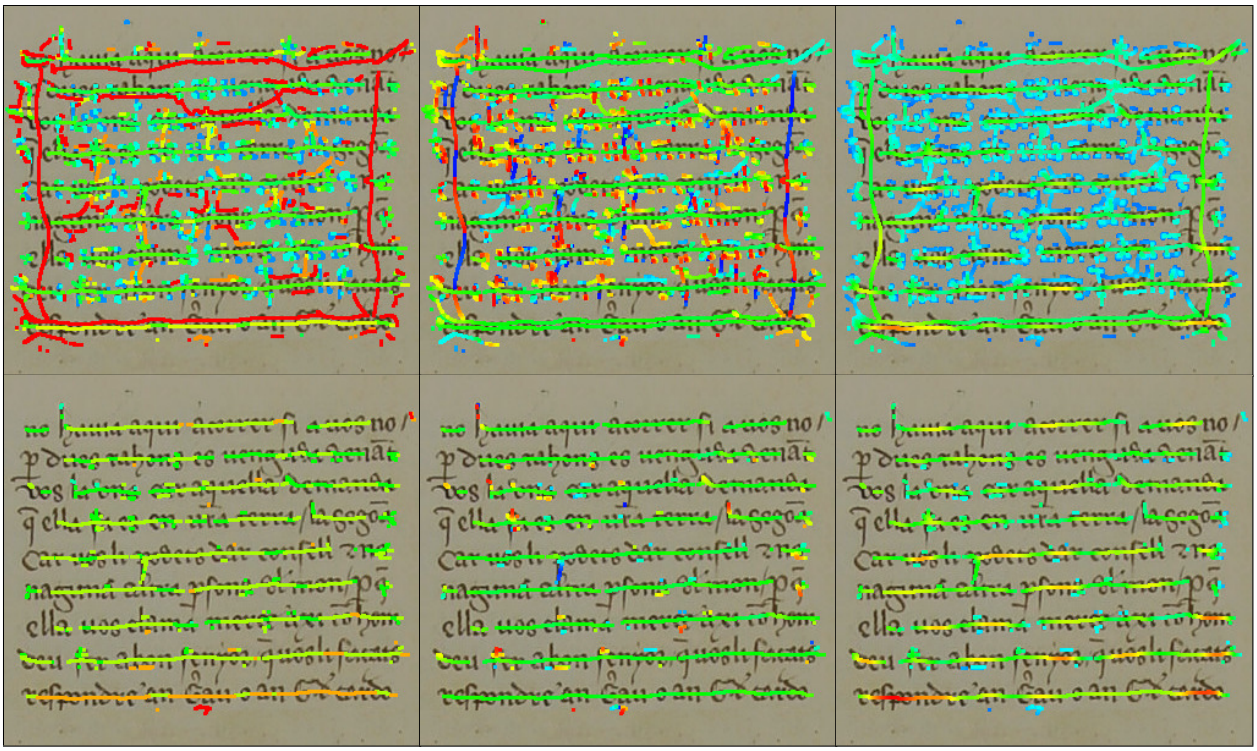
Then, by setting the derivative of Eq. 1 to zero and solving for θ, we find the angles where the filter attains a maximum or minimum value for

$$\theta_a(i) = \frac{1}{2}\arctan\left(\frac{2\mathbf{C}_2(i)}{\mathbf{C}_3(i) - \mathbf{C}_1(i)}\right),$$

and $\theta_b(i) = \theta_a(i) + \pi/2$. We choose the angle $\theta_a(i)$ or $\theta_b(i)$ using Eq. 1 and looking at the orientation which gives the strongest response.

B. Scale Selection

The Gaussian filter has a strong response over the text lines when the σ of the Gaussian is similar to the height of the text line. When this happens, the details of the text line characters' have been blurred enough so the line appears as a blob (see Fig. 1). Therefore, selecting the appropriate σ value for each text line is important to detect line elements independently of the size of the text. The best σ value for each pixel can be automatically calculated by selecting the σ that its second-order Gaussian derivative scaled by



SCALE ORIENTATION VALUE

Figure 3: Example of the selected pixels over the original document image. The magnitude of the values is shown using a hue color map. In the orientation images, green pixels correspond to horizontal orientations while red and blue pixels correspond to vertical orientations (orientation is circular). In the first row we show all local maxima, while in the second row we see the local maxima after applying the non-maxima suppression filter.

$\sqrt{\sigma}$ gives the strongest response. We use the scale factor $\sqrt{\sigma}$ to increase the response at larger scales. Otherwise, we would only consider the smaller scales as document details dilute as we increase the σ of the Gaussian filter. Once we have processed all the scales, we have an image with the line orientation, scale and strength for each pixel. Although this is a computationally expensive approach, it allows to obtain the scale parameter automatically and is efficiently computed by employing a spatial pyramid and recursive Gaussian filters [17] that present a computational cost that is independent of the size of the Gaussian.

Finally, we only consider the pixels that have the strongest response within their line. Therefore, we discard all the pixels that do not have a local maximum when compared to the two neighbors at the perpendicular of the selected orientation. By comparing the neighbors, we assume that two neighboring pixels may belong to the same line when they have a small difference between their orientations and scales. This is similar to extract the ridges of the Gaussian filter response. As a further filtering step, we apply a non-maxima suppression approach where any selected ridge is removed if it falls within the area influence of another local maxima that has a higher filter response. In Fig. 3, we show the selected orientation, value and scale of the selected pixels for the image in Fig. 1. In the scale image, we see that the ascenders and descenders have a low value (shown in blue) as they are detected by filters with a smaller σ value. The pixels with a mid-low value (marked as cyan) correspond to the center of the text lines and the pixels marked as red correspond to filters with a large σ. In this example, most red pixels are at the margins of the image and they correspond to the margins of the text paragraph. In the orientation image, we can see that pixels corresponding to horizontal structures all have a similar color (they are green) while elements corresponding to vertical elements (e.g. ascenders and descenders) are marked as red and blue. In this example, we can also see that the non-maxima suppression filters most of the undesired responses.

III. LINE SEGMENTATION

Once we have obtained the image with the local orientation, scale and value of the filter, we aggregate them to select the most common orientation and scale of the text lines, detect the text regions, the text lines and finally, obtain the text line segmentation.

A. Orientation and Scale Selection

The first step we take is to obtain which are the most common orientation and scale of the detected line segments. To do so, we accumulate all orientation and scale values of the selected pixels into a histogram. The contribution of each pixel is weighted by the strength of the response of the Gaussian filter. In Fig. 5, we present an example of the different orientation-scale histogram

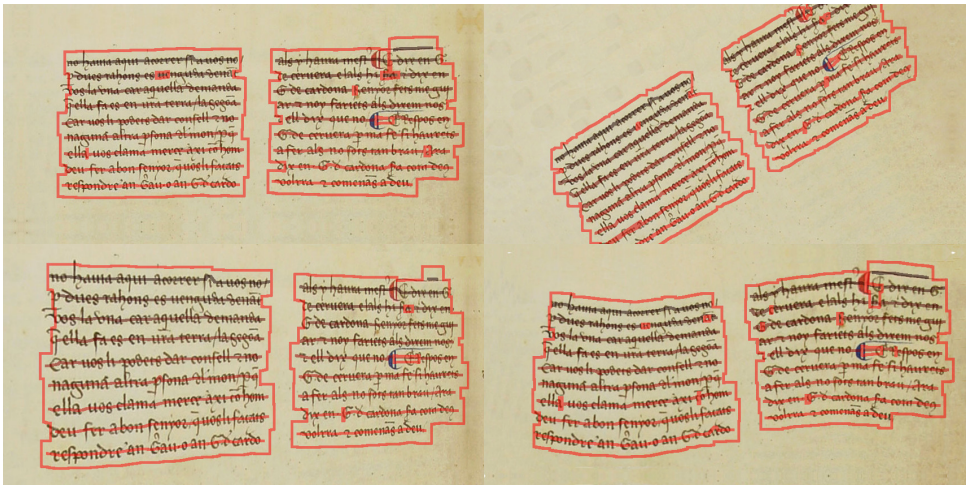
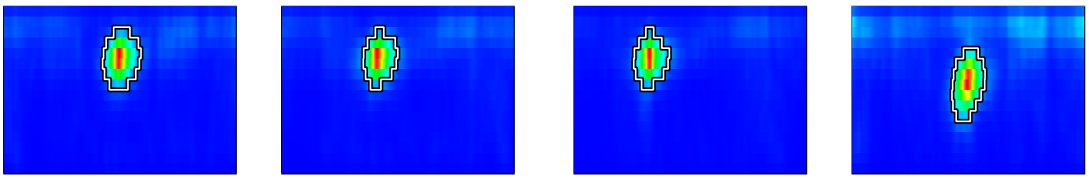


Figure 4: Text region and line detection on the same image under different transforms.



ORIGINAL IMAGE | 15 DEGREES ROTATION | 30 DEGREES ROTATION | AFFINE TRANSFORM

Figure 5: Histograms obtained for the original image, the image transformed by a rotation and an affine transform. In this histogram images, the x-axis corresponds to the orientation while the y-axis corresponds to the σ of filter.

changes obtained when the image from Fig. 1 is wrapped by a rotation or an affine transform (c.f. Fig. 4). Comparing histograms of the original and rotated images, we see that the histogram are simply shifted in the x-axis depending on the rotation applied to the image. The peak at the affine histogram does not move but the values around the peak show a higher *dispersion* as the scale and orientation is slightly different at the margins of the paragraph.

The histograms of Fig. 5 show that selecting only the pixels which have the same orientation and scale as the histogram's local maxima position is going to filter out too many filter responses. Also, selecting the values which lay within a small window defined around the local maxima is likewise too restrictive (e.g. the window in the affine case has to be larger than in the rigid transform images). Therefore, we apply a watershed algorithm [18] on the histogram in order to assign each histogram value to the local maximum that we will reach while following the gradient direction. Following this procedure, a set of orientation and scale pairs are associated to each local extrema of the histogram. Finally, we discard values which are lower than a ratio of the global maximum value of the histogram. In Fig. 5, these regions are delimited by the black and white border defined around each local extrema.

B. Text Region and Line Detection

Once we have selected a scale and an orientation, we use the responses of the Gaussian filter at these parameters to detect the text regions and lines. First, we group the ridges into line segments using connected components. Then, we compute the median separation between consecutive overlapping segments to have an estimation of the separation between text lines L_s. This measure is used to set the maximum distance between neighboring segments. So, by grouping all segments that are within this maximum distance, we obtain regions of the image where text lines are likely to appear. Regions which are too small (a 10% of the largest region) are filtered out. In Fig. 4, the contour around the text show the two text regions detected following this procedure.

The text regions are processed independently to search for text lines. These lines are formed by successively merging the closest segments within the region until their distance exceeds $\sqrt{L_s L_s}$ or only a single segment is left. We compute the distance between two segments as the smallest distance between its end points. In order to avoid merging segments which are in two different lines, we give a higher weight to the height difference than to the separation between lines. So, the distance between the endpoints $\mathbf{p}_a = [p_a^x, p_a^y]$ and $\mathbf{p}_b = [p_b^x, p_b^y]$ is computed as

$$d(\mathbf{p}_a, \mathbf{p}_b) = |p_a^x - p_b^x| + \min(|p_a^y - p_b^y|, L_s/3) + \max(0, |p_a^y - p_b^y| - L_s/3)^2.$$

This measure only gives a quadratic weight to the height differences greater than $L_s/3$, so it greatly penalizes segments which are at a different text line but allows small height differences between consecutive segments. Thus, it gives a certain flexibility to adapt to text line curvature. In Fig. 4, we show detected lines in document images under

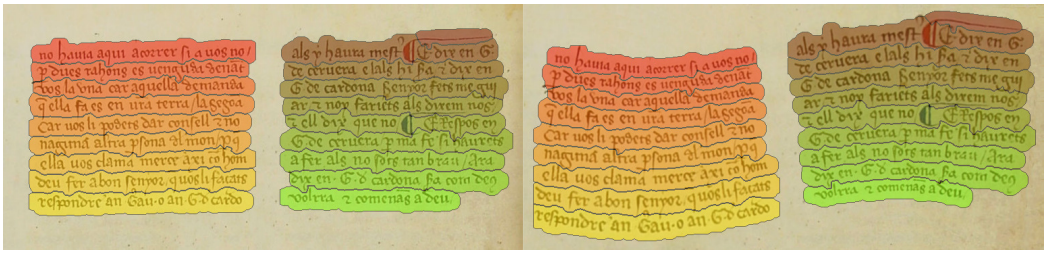


Figure 6: Final segmentation under different image transforms.

different transforms. The coordinates of the end-points are rotated according to the text line orientation, so the x-axis represents the separation between segments and y-axis the separation between the text lines.

C. Text Segmentation

The algorithm so far has only detected the center of the lines, a text line height given by the Gaussian filter and the separation between lines. This parameters can be used to obtain a coarse segmentation of the text lines, but to obtain a finer segmentation we employ a segmentation schema similar to the one proposed by Vo and Lee in [19]. We assign the binary connected components to the closest text line. Components which can be assigned to multiple text lines are decomposed into smaller components using line adjacency graphs [20]. Components which are still assigned to multiple lines are assigned to the line closer to the most of its pixels. The final segmentation is obtained by assigning any pixel within L_s pixels distance to the line of its closest component. Fig. 6 shows the segmentation obtained in the original and deformed document images.

IV. Experimental Results

We have evaluated the segmentation algorithm on the IAM database [21], the GRPOLY-DB dataset [22] and the Saintgall and Parzival datasets [23]. The performance of the algorithm is evaluated following the ICDAR 2009 [24] competition, i.e. assigning each predicted line to the groundtruth line with the highest intersection over union score of its binarized pixels and only considering the matches that have more than 90% overlap. Then, the performance is simply reported with the attained precision and recall. Additionally, we have tested the line detection algorithm on the simple documents track of the cBAD dataset [25]. In this dataset, results are also given in precision-recall score but instead of evaluating the segmentation they evaluate the baseline detection accuracy. Therefore, we return as baselines the lines detected by the algorithm displaced $\sigma_s/2$ down where σ_s is the σ of the selected Gaussian filter. For all datasets we use the same parameters. The size of the images is reduced by half, the Gaussian scale-space is computed at $\sigma \in \{2, 4, 6, 8, 10, 14, 18, 22, 26, 30\}$, the maximum distance between line segments is set to $1.2L_s$ and the Sauvola-Pietikäinen binarization window is set to 50 pixels. The only exception is the cBAD dataset where we do not down-scale the images so the Gaussian filters are larger with $\sigma \in \{5, 10, 14, 18, 22, 26, 30, 38, 46\}$. The results obtained in all datasets are shown in Table I. These results show that the proposed algorithm is comparable to the state of the art algorithm despite not relying on any supervised learning stage to robustly detect the text lines. The algorithm is also fast as it processes an image from the Parzival dataset in around 1.5 seconds only relying on the CPU. Since the most computationally expensive step of the algorithm is the computation of the Gaussian filter derivatives, this runtime can be greatly reduced by the use of GPU acceleration.

Dataset	Method	Precision	Recall	f-Measure
Parzival	CNN [12]	98.9%	98.7%	98.8%
	RLSA [12]	70.2%	50.0%	58.5%
	Ours	*93.3%*	*92.7%*	*93.0%*
	Ours test only	*95.0%*	*95.2%*	*95.1%*
Saintgall	CNN [12]	96.5%	96.4%	96.5%
	RLSA [12]	92.6%	89.6%	91.1%
	Ours	*97.8%*	*97.7%*	*97.8%*
	Ours test only	*98.5%*	*98.5%*	*98.5%*
GRPOLY-DB	Shredding [26], [22]	80.6%	92.4%	86.1%
	Hough [27], [22]	94.2%	96.7%	95.4%
	Ours	*90.2%*	*93.1%*	*91.6%*
IAM	Projection [28]	–	–	37.7%
	Rectangle [29], [28]	–	–	96.7%
	Hough [27], [28]	–	–	92.6%
	Shredding [26], [28]	–	–	36.0%
	Ours	*99.8%*	*99.7%*	*99.7%*
cBAD Track-A	DMRZ [25]	97.3%	97.0%	97.1%
	UPVLC [25]	93.7%	85.5%	89.4%
	BYU [25]	87.8%	90.7%	89.2%
	IRISA [25]	88.3%	87.7%	88.0%
	LITIS [25]	78.0%	83.6%	80.7%
	Ours	*74.7%*	*92.6%*	*82.7%*

Table I: Text line segmentation results

V. Conclusions

In this paper, we show that the second order derivatives can be used to detect the orientation and scale at pixel level. The main advantage of this operator, when compared to anisotropic Gaussian filters or Gabor filters, is that it is steerable and can be computed very efficiently with a space pyramid and recursive Gaussian filters. Then, we use the output of this operator to select the dominant orientation and scale of the lines present at the image and obtain the detected text regions and lines by simply merging the detected line segments. Finally, the line detection is easily extended to line segmentation by assigning

the binary components to the closest detected line and assigning all pixels to the nearest component. Although the algorithm is simple it attains a performance similar to more complex approaches which rely on supervised machine learning strategies.

ACKNOWLEDGMENTS

This work was supported by the Spanish projects TIN2014-52072-P and TIN2017-89779-P, by the CERCA Programme / Generalitat de Catalunya and by the AGAUR and FEDER project 2016 LLAV 00057.

REFERENCES

[1] M. Brown and W. Seales, "Image restoration of arbitrarily warped documents," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 26, no. 10, pp. 1295–1306, October 2004.

[2] A. Graves, M. Liwicki, S. Fernández, R. Bertolami, H. Bunke, and J. Schmidhuber, "A novel connectionist system for unconstrained handwriting recognition," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 31, no. 5, pp. 855–868, May 2009.

[3] A. Fischer, A. Keller, V. Frinken, and H. Bunke, "Lexicon-free handwritten word spotting using character hmms," *Pattern Recognit. Lett.*, vol. 33, no. 7, pp. 934–942, May 2012.

[4] V. Frinken, A. Fischer, R. Manmatha, and H. Bunke, "A novel word spotting method based on recurrent neural networks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 34, no. 2, pp. 211–224, February 2012.

[5] D. Aldavert, M. Rusiñol, R. Toledo, and J. Lladós, "A study of bag-of-visual-words representations for handwritten keyword spotting," *Int. J. Doc. Anal. Recognit.*, vol. 18, no. 3, pp. 223–234, September 2015.

[6] V. Shapiro, G. Gluhchev, and V. Sgurev, "Handwritten document image segmentation and analysis," *Pattern Recognit. Lett.*, vol. 14, no. 1, pp. 71–78, January 1993.

[7] A. Antonacopoulos and D. Karatzas, "Document image analysis for world war ii personal records," in *Proc. Int. Workshop Doc. Image Anal. Libr.*, 2004, pp. 336–341.

[8] L. Likforman-Sulem, A. Hanimyan, and C. Faure, "A hough based algorithm for extracting text lines in handwritten documents," in *Proc. Int. Conf. Doc. Anal. Recognit.*, vol. 2, 1995, pp. 774–777.

[9] F. Cruz and O. Terrades, "Handwritten line detection via an em algorithm," in *Proc. Int. Conf. Doc. Anal. Recognit.*, 2013, pp. 718–722.

[10] R. Saabni, A. Asi, and J. El-Sana, "Text line extraction for historical document images," *Pattern Recognit. Lett.*, vol. 35, pp. 23–33, January 2014.

[11] D. Fernández-Mota, J. Lladós, and A. Fornés, "A graph-based approach for segmenting touching lines in historical handwritten documents," *Int. J. Doc. Anal. Recognit.*, vol. 17, no. 3, pp. 293–312, September 2014.

[12] J. Pastor-Pellicer, M. Afzal, M. Liwicki, and M. Castro-Bleda, "Complete system for text line extraction using convolutional neural networks and watershed transform," in *Int. Workshop Doc. Anal. Syst.*, 2016, pp. 30–35.

[13] B. Moysset, C. Kermorvant, and C. Wolf, "Full-page text recognition: Learning where to start and when to stop," in *Proc. Int. Conf. Doc. Anal. Recognit.*, 2017, pp. 871–876.

[14] L. Likforman-Sulem, A. Zahour, and B. Taconet, "Text line segmentation of historical documents: a survey," *Int. J. Doc. Anal. Recognit.*, vol. 9, no. 2, pp. 123–138, April 2007.

[15] J. Sauvola and M. Pietikäinen, "Adaptive document image binarization," *Pattern Recognit.*, vol. 33, no. 2, pp. 225–236, February 2000.

[16] W. Freeman and E. Adelson, "The design and use of steerable filters," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 13, no. 9, pp. 891–906, September 1991.

[17] L. van Vliet, I. Young, and P. Verbeek, "Recursive gaussian derivative filters," in *Proc. Int. Conf. Pattern Recognit.*, 1998, pp. 509–514.

[18] A. Bieniek and A. Moga, "An efficient watershed algorithm based on connected components," *Pattern Recognit.*, vol. 33, no. 6, pp. 907–916, June 2000.

[19] Q. Vo and G. Lee, "Dense prediction for text line segmentation in handwritten document images," in *Proc. Intern. Conf. Image Process.*, 2016, pp. 3264–3268.

[20] S. Kahan, T. Pavlidis, and H. Baird, "On the recognition of printed characters of any font and size," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 9, no. 2, pp. 274–288, March 1987.

[21] U.-V. Marti and H. Bunke, "The iam-database: an english sentence database for offline handwriting recognition," *Int. J. Doc. Anal. Recognit.*, vol. 5, no. 1, pp. 39–46, November 2002.

[22] B. Gatos, N. Stamatopoulos, G. Louloudis, G. Sfikas, G. Retsinas, V. Papavassiliou, F. Sunistira, and V. Katsouros, "Grpoly-db: An old greek polytonic document image database," in *Proc. Int. Conf. Doc. Anal. Recognit.*, 2015, pp. 646–650.

[23] A. Fischer, E. Indermühle, H. Bunke, G. Viehhauser, and M. Stolz, "Ground truth creation for handwriting recognition in historical documents," in *Int. Workshop Doc. Anal. Syst.*, 2010, pp. 3–10.

[24] B. Gatos, N. Stamatopoulos, and G. Louloudis, "Icdar2009 handwriting segmentation contest," *Int. J. Doc. Anal. Recognit.*, vol. 14, no. 1, pp. 25–33, March 2011.

[25] M. Diem, F. Kleber, S. Fiel, G. Tobias, and B. Gatos, "cbad: Icdar2017 competition on baseline detection," in *Proc. Int. Conf. Doc. Anal. Recognit.*, 2017, pp. 1355–1360.

[26] A. Nicolaou and B. Gatos, "Handwritten text line segmentation by shredding text into its lines," in *Proc. Int. Conf. Doc. Anal. Recognit.*, 2009, pp. 626–630.

[27] G. Louloudis, B. Gatos, I. Pratikakis, and C. Halatsis, "Text line detection in handwritten documents," *Pattern Recognit.*, vol. 41, no. 12, pp. 3758–3772, December 2008.

[28] B. Moysset and C. Kermorvant, "On the evaluation of handwritten text line detection algorithms," in *Proc. Int. Conf. Doc. Anal. Recognit.*, 2013, pp. 185–189.

[29] Z. Shi, S. Setlur, and V. Govindaraju, "A steerable directional local profile technique for extraction of handwritten arabic text lines," in *Proc. Int. Conf. Doc. Anal. Recognit.*, 2009, pp. 176–180.