Using Adaptive Run Length Smoothing Algorithm for Accurate Text Localization in Images

Martin Rais, Norberto A. Goussies, and Marta Mejail

Departamento de Computación, Facultad de Ciencias Exactas y Naturales,
Universidad de Buenos Aires, Buenos Aires, Argentina

Abstract. Text information in images and videos is frequently a key factor for information indexing and retrieval systems. However, text detection in images is a difficult task since it is often embedded in complex backgrounds. In this paper, we propose an accurate text detection and localization method in images based on stroke information and the Adaptive Run Lenght Smoothing Algorithm. Experimental results show that the proposed approach is accurate, has high recall and is robust to various text sizes, fonts, colors and languages.

1 Introduction

The tremendous increase of multimedia content has raised the need for automatic semantic information indexing and retrieval systems. Text information present in images and videos are an important source of high-level semantics.

Large variations in text fonts, colors, styles, and sizes, as well as the low contrast between the text and the often complicated background, make text detection and localization extremely challenging. Frequently, image compression tends to deteriorate image quality resulting in even harder to recognize texts.

Textual information extraction (TIE) is usually split into five steps: detection, localization, verification, segmentation and recognition. Text detection is used to separate text regions from non-text regions. Text localization is used to localize text lines using rectangular bounding boxes (BBs). During text verification, all localized text lines are verified in order to eliminate false positives. Text segmentation is performed to compute the foreground of the text and finally, text recognition is where the detected text image is converted into plain text.

In this work, we present a method that accurately detects, localizes and verifies text in images and video frames. Inspired by [7], stroke information and a machine learning approach is used to detect text with high recall rates and very good precision. To obtain the initial bounding boxes (BBs), we propose a combined Adaptive Run Lenght Smoothing Algorithm (ARLSA) and Connected Components Analysis (CCA) based algorithm that localizes text more accurately than Li et al [7]. After a refinement phase, an innovative ARLSA-based verification is proposed followed by a final SVM verification step using its output scores to further check results.

C. San Martin and S.-W. Kim (Eds.): CIARP 2011, LNCS 7042, pp. 149–156, 2011.

This paper is organized as follows: In section 2 we present current state-of-the-art methods, in section 3 our method will be described in detail, in section 4 we will discuss about experimental results and finally, in section 5 we will draw our conclusions regarding this topic.

2 Related Work

Text detection and localization methods are usually clasified into four distinct categories. The first category uses connected component analysis (CCA) [14] in which regions with maximum homogeneity are obtained from an image and then non-text connected components (CCs) are filtered out based on geometrical constrains. CCA-based methods are robust against font size, however, they are sensible to noise and tend to fail when texts are not homogeneous. The second category uses an edge-based approach to detect and localize text [9]. Techniques within this category detect text based on strength, density or distribution information from edges, and assumes high contrast differences between the text and the background. Edge-based methods are fast and have high recall rates, however, the large amount of false alarms is usually its main problem. Also, it fails when text is not contrasted enough with the background. The third category is based on textures [11], and makes use of the fact that texts have specific texture patterns that allows us to distinguish them from the background. Texture-based methods are robust to noise and low quality images, however they are time-consuming and tend to fail when the background is cluttered with text. Recently, pattern clasiffication methods have been addressed to detect and localize text in images based on elaborately selected features [3]. For more information about textual information extraction in videos and images, the reader should refer to the survey paper [5].

Using stroke information to detect and localize text was first introduced by [8]. Since then, several text localization methods have used stroke information to detect and localize text in images such as [7]. In those works, it has been proven that stroke information can be used to succesfully detect and localize text in images since it captures the intrinsic characteristics of text itself.

3 Methodology

As shown in Fig. 1, the proposed algorithm has three distinct stages for text localization: coarse text detection, a refinement stage followed by text verification.

During the Coarse Text Detection stage, we perform a fast morphology-based coarse text detection using a Multi-Scale Wavelet edge map. Then, stroke filter is performed over the original image using the results from the first detection as a mask to improve performance. We later apply a sliding window algorithm in which we perform SVM classification to generate the initial bounding boxes based on the Adaptive Run Length Smoothing Algorithm (ARLSA) [10].

During the refinement stage, we first apply a Zero Crossing projection profile technique to split multiple line bounding boxes, followed by an Expand method to improve precission.

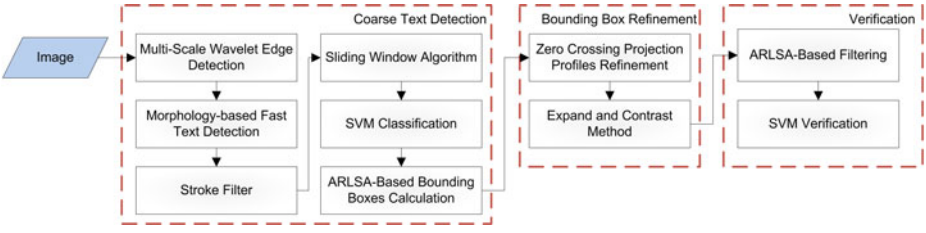


Fig. 1. Flowchart of the proposed algorithm

Finally, in the verification stage, we introduce an ARLSA-based filtering method followed by an SVM verification to further improve precission.

3.1 Coarse Text Detection

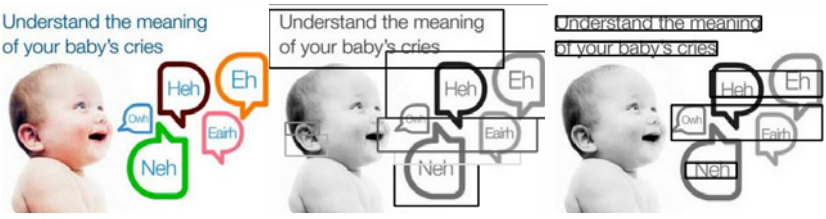
Wavelet Multi-Scale Edge Detection. We use the method in [6] for fast multi-scale edge detection. The scale of the method controls the threshold for which edges are detected. A large scale tends to remove small signal fluctuations, filtering background edges from the results. However, text edges with small signal fluctuations may also be removed if the scale is too large. In our work, we use 3 as the scale of the algorithm.

Initial text region detection. We first convert the input image into grayscale. Then we calculate stroke information using the stroke filter from [8] but only on text areas from the morphology-based text detection. Using the stroke information, we employ the sliding window algorithm from [7] to detect text regions.

A Support Vector Machine (SVM) [2] classifier is used in this work since, as stated in [12], compared with other classifiers such as neural networks and decision trees, SVM is easier to train, needs fewer training samples and has better generalization ability. The SVM was trained on a dataset containing 368 text samples and 611 non-text samples using the same feature set as [7]. The sliding window in this case moves $W/4$ horizontally and $H/4$ vertically. Fig. 3b gives an example of the output result after the stage.

CC Analysis and ARLSA. The authors of [7] used a fixed set of computation steps that heuristically obtain initial bounding boxes partitioning the connected polygons that result from the sliding window procedure mentioned above. However this have several drawbacks, specially the fact that it frequently split text lines in two bounding boxes as in Fig. 2b.

Because of this, we take a new and more effective way in order to obtain the initial bounding boxes. We first calculate a saliency map as in [1]. For every window that is classified as a text line, all the values of its pixels are incremented in the saliency map by one. Meanwhile, we calculate another map that represents the number of visits by a sliding window for each pixel. After the whole image is classified and the sliding window algorithm is over, we use this second map to normalize the saliency map from 0 to 1. The results of this can be seen in



(a) Original Image **(b)** Li et al. method **(c)** Our method

Fig. 2. Initial Bounding Boxes comparison

the Fig. 3b. Then, we generate a binary detection map using the method in [1]. Example of the binary detection map is shown in the Fig. 3c.

We later use this binary detection as a mask for the edges map, removing all edges that are not text-related. We use this new map as the input of the horizontal ARLSA algorithm detailed in [10] generating what can be seen in Fig. 3d. The purpose of the ARLSA algorithm, as used in [13], is to join all characters among the same line, implying that all isolated edges cannot belong to text so we remove them by performing a morphological opening leading to Fig. 3e. Then, we generate the bounding boxes based on the connected components as in Fig. 3f. Finally, we put together all pairs of rectangles based on the following conditions:

- The relation between their heights is lower than a certain threshold T_{height}.
- Their vertical distance is lower than a certain threshold T_{vdist}.
- Their horizontal distance is lower than a certain threshold T_{hdist}.

In this paper, we set $T_{height} = 2.5$, T_{vdist} as the maximum of both heights divided by 2 and T_{hdist} as the minimum of both heights. This will lead to Fig. 3g and Fig. 3h.

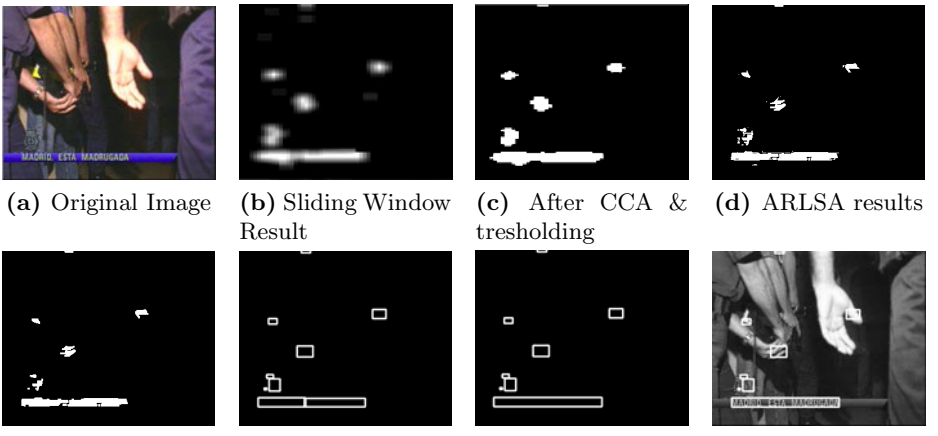
3.2 Localization

After we obtain the initial BBs, a text line refinement is performed. First, a projection profiles based method is used to split BBs with more than one line and shrink them to better adapt to text areas. This is followed by an expand method to further improve precission of the algorithm.

Zero Crossing and Projection Profiles refinement. We use the zero crossing technique from [11] to refine text BBs.

After applying this method, we perform a conventional vertical projection profile method. For each column, the pixel values are summed and if the sum is lower than a certain threshold, the column is deleted. In order to avoid splitting two words of the same text line, we require that the number of concurrent columns marked to be deleted equals a certain threshold.

In our work, both methods are applied sequentially, and will be repeated until no further changes are made to the bounding boxes after an iteration.



(a) Original Image **(b)** Sliding Window Result **(c)** After CCA & tresholding **(d)** ARLSA results

(e) Morphological Opening **(f)** Initial Bounding Boxes **(g)** Text Line Join **(h)** Final Bounding Boxes

Fig. 3. ARLSA-based bounding boxes calculation

Expand. It may happen that the Zero Crossing technique shrinks a BB supressing any of both horizontal borders of a text region (top or bottom). Also, it could happen that during the coarse detection, a BB missed any border of a text region. Thus, it is neccesary to expand the BBs in order to obtain better localization recall and precission.

For this purpose, we multiply the image resulting from the sliding window such as Fig. 3b by the multi-scale wavelet edge detection and we use this image as the input of our expand procedure.

To expand, we first take the next horizontal line above the top edge of the BB and calculate its sum. We expand bounding box B horizontally until threshold T_h is not met. We do the same with the next horizontal line below the bottom edge. We then make the same procedure vertically until treshold T_v is not verified. Treshold T_h is obtained in the following way:

$$T_h(B) = (Avg_h(B) - Min_h(B)) * k_e \tag{1}$$

where k_e is a constant and

$$Avg_h(B) = \frac{1}{H} \sum_{i}^{H} \sum_{j}^{W} B(j,i) \tag{2}$$

and $Min_h(B)$ stands for the minimum value of the horizontal projection profiles of the bounding box B while W and H are their respectives width and height. $k_e = 0.75$ is used for this work. The vertical treshold T_v is obtained the same as T_h but vertically.

3.3 Verification

Once the localization stage has refined the rectangles, some false positives bounding boxes may still be part of the answer. Therefore, we first apply an innovative ARLSA-based verification procedure followed by a combined SVM approach that uses SVM output scores to verify candidate bounding boxes.

ARLSA-based verification. To eliminate false positive, we perform an ARLSA-based filtering. For each bounding box, we calculate the ratio of ones among the bounding box on the ARLSA image, such as in Fig. 3d. If this ratio is below a certain treshold k_{ARLSA}, we will classify the bounding box as non-text and remove it. For this work, we used $k_{ARLSA} = 0.7$.

SVM-based verification. In order to perform an SVM-based verification, we employ two SVMs. Our first SVM is trained using the same feature set as in the text detection stage, but in this case, using variable sized bounding boxes. We consider the prediction score obtained from this SVM and if it is within a predefined range [$-k_{SVM}$, k_{SVM}] where $k_{SVM} \leq 1$, it implies that the prediction is not certain. Thus, in those cases, we further verify the bounding box using a third SVM. The feature set used in this case includes the 6 features used in the second SVM of [7] plus one new feature to achieve better recognition accuracy. The new feature introduced is the mean of the image map that results from the ARLSA algorithm in the bounding box. This is discriminative since, as shown in Fig. 3d, the ARLSA map of a text region must have most of their pixels marked.

4 Results

In order to detect bigger font sizes, we employ a multiresolution approach performing our three stages of textual information extraction to an image in different resolutions and finally results are combined to the original resolution. To avoid combining overlapping bounding boxes of different resolutions, the approach from [1] is used eliminating the edges in the current resolution of the already detected characters in the previous resolutions.

As explained in [1], most authors use box-based or pixel-based recall, precission and accuracy measures that in general, fail to reflect the true quality of the results. Very few works deal with the problem of evaluation methods.

Our evaluation method was proposed in Anthimopoulos et al. [1], and is character oriented, meaning it will base its results on the quantity of characters recognized/missed. They conclude that the number of characters in a text line is proportional to its ratio width to height. Based on that, let GB_i be the ith ground truth bounding box with $1 \leq i \leq N$ and hg_i its height while DB_j be the jth detected bounding box with $1 \leq j \leq M$ and hd_j its height, they propose to calculate the recall and precision as:

$$\textbf{Recall} = \frac{\sum_{i=1}^{N} \frac{|GDI_i|}{hg_i^2}}{\sum_{i=1}^{N} \frac{|GB_i|}{hg_i^2}} \tag{3}$$

$$\textbf{Precision} = \frac{\sum_{i=1}^{M} \frac{|DGI_i|}{hd_i^2}}{\sum_{i=1}^{M} \frac{|DB_i|}{hd_i^2}} \tag{4}$$

where GDI_i and DGI_i are the corresponding intersections:

$$GDI_i = \begin{cases} GB_i, & \text{if } \frac{GB_i \cap \left(\bigcup_{i=1}^{M} DB_i\right)}{GB_i} \geq th \\ GB_i \cap \left(\bigcup_{i=1}^{M} DB_i\right), & \text{otherwise} \end{cases} \tag{5}$$

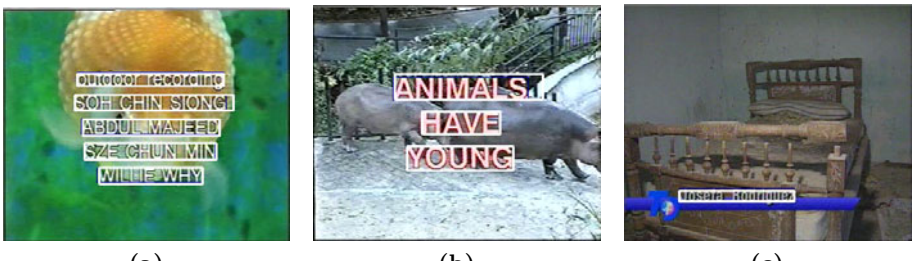
$$DGI_i = \begin{cases} DB_i, & \text{if } \frac{DB_i \cap \left(\bigcup_{i=1}^{N} GB_i\right)}{DB_i} \geq th \\ DB_i \cap \left(\bigcup_{i=1}^{N} GB_i\right), & \text{otherwise} \end{cases} \tag{6}$$

and $th = 0.75$ is a treshold that avoids penalizing minor inconsistencies.

To test our algorithm, we used the Microsoft common test set from [4]. As can be seen in Table 1, our method outperforms previous methods. Some sample results are shown in Fig. 4.

Table 1. Performance Comparison

Method	Recall (%)	Precision (%)	F-Measure (%)
Our approach	**93.34%**	**96.7%**	**94.99%**
Li et al. [7]	91.1%	95.8%	93.39%
Liu et al. [8][1]	91.3%	92.4%	91.85%
Ye et al. [12][1]	90.8%	90.3%	90.55%



(a) **(b)** **(c)**

Fig. 4. Sample text localization results using the proposed method

5 Conclusion

In this paper, we proposed an effective text detection and localization method based on stroke information. To overcome speed issues, we first perform a fast morphological text detection that we later use as a mask for calculating the stroke filter. We then detect text using a machine-learning approach and obtain

[1] As reported in [7].

initial bounding boxes by a mixed CCA and an ARLSA approach. To further refine the bounding boxes, we employ a zero-crossing projection profile technique followed by a expand technique to gain better recall. Finally, the text regions are verified by an SVM approach using their output scores to further check the results.

Experimental results show that the proposed text detection and localization method is robust to noise, text size, color and text language. It does also outperform other stroke-based methods such as [7].

References

[1] Anthimopoulos, M., Gatos, B., Pratikakis, I.: A two-stage scheme for text detection in video images. Image Vision Comput. 28, 1413–1426 (2010)
[2] Cortes, C., Vapnik, V.: Support-vector networks. Machine Learning 20(3), 273–297 (1995)
[3] Herv, D., Chen, D., Bourlard, H.: Text Identification in Complex Background Using SVM. In: Proc. of IEEE Conference on Computer Vision and Pattern Recognition, vol. 2, pp. 621–626 (2001)
[4] Hua, X., Wenyin, L., Zhang, H.: An Automatic Performance Evaluation Protocol for Video Text Detection Algorithms. IEEE Transactions on Circuits and Systems for Video Technology 14, 498–507 (2004)
[5] Jung, K.: Text information extraction in images and video: a survey. Pattern Recognition 37(5), 977–997 (2004)
[6] Li, J.: A Wavelet Approach to Edge Detection. Master's thesis, Sam Houston State University, Huntsville, Texas (August 2003)
[7] Li, X., Wang, W., Jiang, S., Huang, Q., Gao, W.: Fast and Effective Text Detection. In: International Congress in Image Processing 2008 (2008)
[8] Liu, Q., Jung, C., Kim, S., Moon, Y., Kim, J.: Stroke Filter for Text Localization in Video Images, pp. 1473–1476 (October 2006)
[9] Lyu, M.R., Song, J., Cai, M.: A comprehensive method for multilingual video text detection, localization, and extraction. IEEE Transactions on Circuits and Systems for Video Technology 15(2), 243–255 (2005)
[10] Nikolaou, N., Makridis, M., Gatos, B., Stamatopoulos, N., Papamarkos, N.: Segmentation of historical machine-printed documents using Adaptive Run Length Smoothing and skeleton segmentation paths. Image Vision Comput. 28, 590–604 (2010)
[11] Shivakumara, P., Phan, T.Q., Tan, C.L.: A Gradient Difference Based Technique for Video Text Detection, pp. 156–160 (2009)
[12] Ye, Q., Huang, Q., Gao, W., Zhao, D.: Fast and robust text detection in images and video frames. Image and Vision Computing 23(6), 565–576 (2005)
[13] Zhao, M., Li, S., Kwok, J.: Text detection in images using sparse representation with discriminative dictionaries. Image and Vision Computing 28(12), 1590–1599 (2010)
[14] Zhong, Y., Karu, K., Jain, A.K.: Locating text in complex color images. In: Proceedings of the Third International Conference on Document Analysis and Recognition, ICDAR 1995, vol. 1. IEEE Computer Society, Washington, DC, USA (1995)