ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 4, Issue 10, September 2019

A Review on Image, Video Frames, Text Detection and Character Recognition in Scene

[1] Anirudha Kolpyakwar, [2] Ritesh Deshmukh, [3] Kiran Chavhan, [4] Aashna Rukhsaar, [5] Rahul Raut
[1][2][3][4][5] Assistant Professor, Department of Computer Engineering, Jagadambha College of Engineering and Technology, Yavatmal, India

***Abstract:*---Reading text from image is very difficult that has received a significant amount of attention. The major key components of most systems are (i) text detection from images and (ii) character recognition, and many recent methods have been proposed to design better feature representations and models for both. In this paper, an efficient algorithm which can automatically detect, localize and extract horizontally aligned text in images (and digital videos) with complex backgrounds is presented. The proposed approach is based on the application of a color reduction technique, a method for edge detection, and the localization of text regions using projection profile analyses and geometrical properties. The output of the algorithm are text boxes with a simplified background, ready to be fed into an OCR engine for subsequent character recognition. Our proposal is robust with respect to different font sizes, font colors, languages and background complexities. The performance of the approach is demonstrated by presenting promising experimental results for a set of images taken from different types of video sequences.**

***Key words:*---optical character recognition, subtitles, visually impaired people, Character recognition, feature extraction, pattern matching, training..**

I. INTRODUCTION

Detecting text in natural images, as opposed to scans of printed pages, faxes and business cards, is an important step for a number of Computer Vision applications, such as computerized aid for visually impaired, automatic geocoding of businesses, and robotic navigation in urban environments. Retrieving texts in both indoor and outdoor environments provides contextual clues for a wide variety of vision tasks. Moreover, it has been shown that the performance of image retrieval algorithms depends critically on the performance of their text detection modules. For example, two book covers of similar design but with different text, prove to be virtually indistinguishable without detecting and OCRing the text.

The recognition rate in these algorithms depends on the choice of features. Most of the existing algorithms involve extensive processing on the image before the features are extracted that results in increased computational time.

In this paper, we propose a new text detection method which allows to detect, localize and extract texts from color images with complex backgrounds. The approach is targeted towards being robust with respect to different kinds of text appearances, including font size, color and language. To achieve this aim, the main focus of the proposed algorithm is centered on the recognition of the specific edge characteristics of characters. Based on the way how possible text areas are detected and localized, our method can be classified as a connectedcomponent based approach. It essentially works as follows: Color images are first converted to grayscale images. An edge image is generated using a contrast segmentation algorithm, which in turn uses the contrast of the character contour pixels to their neighboring pixels. This is followed by the analysis of the horizontal projection of the edge image in order to localize the possible text areas. After applying several heuristics to enhance the resulting image created in the previous step, an output image is generated that shows the text appearing in the input image with a simplified background. The performance of our approach is illustrated by presenting experimental results for different sets of images.

II. LEARNING CHARACTERS

The Text recognition algorithm relies on a set of learned characters and their properties. It compares the characters in the scanned image file to the characters in this learned set.Generating the learned set is quite simple. It requires that an image file with the desired characters in the desired font be created, and a text file representing the characters in this image file. If a character such as pi, has a multicharacter translation, angled brackets should be placed around the translation.Once the learned set has been read in from the image file and its properties recognized, it can be written out to a "learn" file. This file stores the properties of the learned characters in abbreviated form, eliminating the need for retaining the images of the learned characters, and can be read in very quickly.

The main goals are image acquisition, pre- processing, feature extraction and pattern generation. The main task of

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 4, Issue 10, September 2019

image acquisition module is to obtain text image from a scanner or a pre-stored image file. It is called „image" because scanner inherently scans pixel of the text and not characters when patterns are scanned and digitized, the data may carry some unwanted noise. For example, a scanner with low resolution may produce touching line segments and smeared images. A pre-processor is used to smooth the digitized characters. Moreover, the system must be able to handle touching characters, proportional spacing, variable line spacing and change of font style in the scanned text, in addition to the problems of multi-fonts.

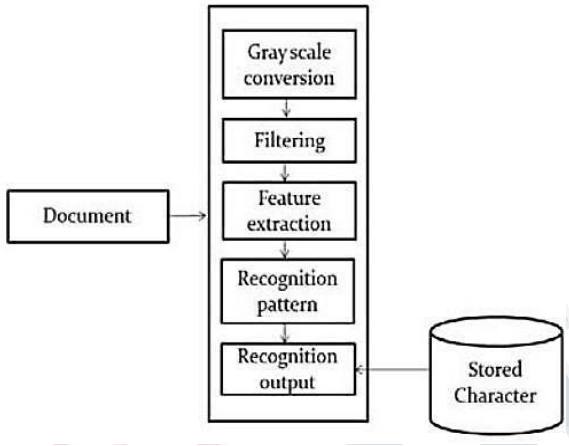

Figure 1. System Block Diagram

III. LINE DETECTION

To detect lines of text (which is later useful in determining the order of characters and possibly their layout on the page) we do a horizontal projection of the page. Our original plan was to detect line breaks using some kind of statistical analysis, but for now, adjacent lines in the image having a very small number of pixels constitute a line break. Noise can deceive this simple algorithm, but adjusting the NoiseTolerance global variable can usually allow the user to overcome this shortcoming.

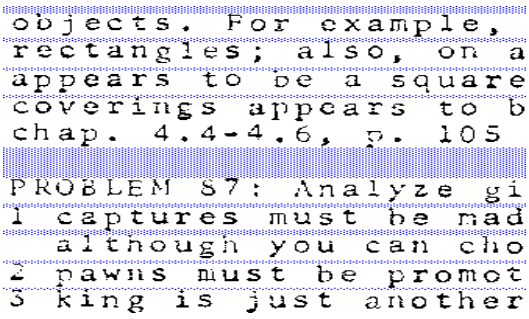

Figure 2. Line detection using pixels

IV. COMPONENT EXTRACTION/PROJECTION ABOVE AND BELOW

To extract the connected components from each line, OCRchie, starting at the upper right corner of each line, removes touching intervals of black pixels from the run-length-encoded representation of the image until nothing more connected can be found. The extraction routine then looks upward and downward to see if there are possible "extra parts", such as the dot on an 'i', hanging directly above or below the component. It ignores components that are only minimally above or below the extracted component, as when the tail of a 'y' extends below the letter preceding it. The projection does not occur if the region has been marked as an equation in the user interface.

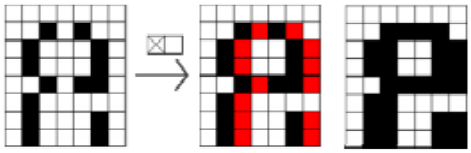

Figure 3: Principle behind dilation using 2 pixel structural element

Figure 5:Letter before filtering

Figure 4:Letter after lowpass-filter and dilation

V. CHARACTER PROPERTY EXTRACTION AND COMPARISON

After extracting the individual characters in a document and determining their properties, we compare them to the learned set. The comparison algorithm is simple: it sums the squares of the differences between each property in the extracted character and each property in the learned character, returning a "Confidence". Initially, we compared each extracted character to the entire linked list of learned characters, but in the interest of speed, we modified this slightly to classify characters in relation to the baseline of their line of text. Letters like "g", "j" & "y", which extend below the baseline are in one group, tall letters, such as "l" and "T" are in another, short characters, like "a" and "c" another, and floating characters, such as "'" and "^" are in the last. Once classified, an extracted character is compared

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 4, Issue 10, September 2019

to learned characters which are in the same group. If no good match is found, the extracted character is then compared to the other learned characters, regardless of group.

The major problems comes in the detection or in the extraction process the comparison of upper case and lower case letters are easy with the help of ASCII text but it comes to very difficult in special characters. The difference between „I‟ and the „ I‟ is giving false. In these one is small L and other is capital I. same thing happening in A and P., O. and Q. To resolve these things we divide the frames in to micro partitions and after that merge small groups. It is only possible with high clarity images.

VI. SPLITTING WIDE CHARACTERS AND GROUPING LETTERS INTO TEXT

The most obvious cause of misrecognition in our original program was linked characters. An "r" would just barely touch an "i", and the character would be recognized as an "n". To alleviate this problem, when encountering a low confidence match on a wide character, OCRchie will split the character at its most narrow point. If the confidence of the left side of the split higher than before, the character is assumed to be joined, and the split remains. Otherwise, the split is rolled back.

This approach could possibly cause problems with something like "mi"-- with a poorly scanned "m", the joined character could be broken in the middle of the "m", find an "n", and do something unpredictable with the remnants of the "m" and the "i".

An important cue for text is that it appears in a linear form. Text on a line is expected to have similarities, including similar stroke width, letter width, height and spaces between the letters and words. Including this reasoning proves to be both straightforward and valuable. For example, a lamp post next to a car wheel would not be mistaken for the combination of letters “O” and “I” as the post is much higher than the wheel. We consider each pair of letter candidates for the possibility of belonging to the same text line. Two letter candidates should have similar stroke width. The distance between letters must not exceed three times the width of the wider one. Additionally, average colors of candidates for pairing are compared, as letters in the same word are typically expected to be written in the same color. All parameters were learned by optimizing performance on the training set. At the next step of the algorithm, the candidate pairs determined above are clustered together into chains. Initially, each chain consists of a single pair of letter candidates. Two chains can be merged together if they share one end and have similar direction. The process ends when no chains can be merged. Each produced chain of sufficient length (at least 3 letters in our experiments) is considered to be a text line. Finally, text lines are broken into separate words, using a heuristic that computes a histogram of horizontal distances between consecutive letters and estimates the distance threshold that separates intra-word letter distances from inter-word letter distances.

Figure 6: Text detection results on several images from the

Algorithm	Precision	Recall	*f*	Time (sec.)
Our system	**0.73**	**0.60**	**0.66**	**0.94**
Hinnerk Becker	0.62	0.67	0.62	14.4
Alex Chen	0.60	0.60	0.58	0.35
Qiang Zhu	0.33	0.40	0.33	1.6
Jisoo Kim	0.22	0.28	0.22	2.2
Nobuo Ezaki	0.18	0.36	0.22	2.8
Ashida	0.55	0.46	0.50	8.7
HWDavid	0.44	0.46	0.45	0.3
Wolf	0.30	0.44	0.35	17.0
Todoran	0.19	0.18	0.18	0.3
Full	0.1	0.06	0.08	0.2

Table 1: Performance comparison of text detection algorithms.

VII. CONCLUSION

In this work we show how to leverage on the idea of the recovery of stroke width for text detection. We define the notion of a stroke and derive an efficient algorithm to compute it, producing a new image feature. Once recovered, it provides a feature that has proven to be reliable and flexible for text detection. Unlike previous features used for text detection, we compared to the most recent available tests, our reached first place and was about 15 times faster than the speed reported there. The feature was dominant enough to be used by itself, without the need for actual character recognition step as used in some previous works.

ISSN (Online) 2456 -1304

International Journal of Science, Engineering and Management (IJSEM)
Vol 4, Issue 10, September 2019

This allows us to apply the method to many languages and fonts. There are several possible extensions for this work. The grouping of letters can be improved by considering the directions of the recovered strokes. This may allow the detection of curved text lines as well.

Figure 7: Examples of failure cases. These include: strong highlights (a), transparent text (b), text that is too small (c), blurred text (d) and text with curvature beyond our range (e)

Figure 8: The algorithm was able to detect text in very challenging scenarios such as blurry images, non planar surfaces, non uniform backgrounds, fancy fonts and even three dimensional fonts.

REFERENCES

1. R. Salakhutdinov and G. E. Hinton, "Deep Boltzmann Machines," in 12th International Conference on AI and Statistics, 2009.
2. M. Ranzato, A. Krizhevsky, and G. E. Hinton, "Factored 3- way Restricted Boltzmann Machines for Modeling Natural Images," in 13th International Conference on AI and Statistics, 2010.
3. Y. Bengio, P. Lamblin, D. Popovici, and H. Larochelle, "Greedy layer-wise training of deep networks," in Neural Information Processing Systems, 2006.
4. R. Raina, A. Battle, H. Lee, B. Packer, and A. Ng, "Selftaught learning: transfer learning from unlabeled data," in 24th International Conference on Machine learning, 2007.
5. J. C. van Gemert, J. M. Geusebroek, C. J. Veenman, and A. W. M. Smeulders, "Kernel codebooks for scene categorization," in European Conference on Computer Vision, 2008.
6. L.-J. Li, H. Su, E. Xing, and L. Fei-Fei, "Object bank: A high-level image representation for scene classification and semantic feature sparsification," in Advances in Neural Information Processing Systems, 2010.
7. Y. Pan, X. Hou, and C. Liu, "A robust system to detect and localize texts in natural scene images," in International Workshop on Document Analysis Systems, 2008.
8. J. J. Weinman, E. Learned-Miller, and A. R. Hanson, "A discriminative semi-markov model for robust scene text recognition," in Proc. IAPR International Conference on Pattern Recognition, Dec. 2008.
9. X. Fan and G. Fan, "Graphical Models for Joint Segmentation and Recognition of License Plate Characters," IEEE Signal Processing Letters, vol. 16, no. 1, 2009.
10. J. J. Weinman, "Typographical features for scene text recognition," in Proc. IAPR International Conference on Pattern Recognition, Aug. 2010, pp. 3987–3990.
11. A.K. Jain and B. Yu: Automatic Text Location in Images and Video Frames. Pattern Recognition. Vol. 31, No. 12, (1998)2055-2076.
12. X.S. Hua, W.Y. Liu, H.J. Zhang: Automatic Performance Evaluation for Video Text Detection. In: Int. Conf. on Document Analysis and Recognition (2001).
13. D. T. Chen, H. Bourlard, J-P. Thiran: Text Identification in Complex Background Using SVM. Int. Conf. on CVPR (2001).
14. Y. Zhong, H.J. Zhang, and A. K. Jain: Automatic Caption Localization in Compressed Video. IEEE Trans. on PAMI, Vol. 22, No. 4, (2000)385-392.
15. R. Lienhart and A. Wernicke: Localizing and Segmenting Text in Images and Videos. IEEE Trans. on CSVT, Vol.12, No.4 (2002).