Report on Methods of Binarization, Skew Correction and Word Segmentation Employed in Indic Tools

Suryoday Basak
PESIT Bangalore South Campus
suryodaybasak@gmail.com

September 2015

Abstract

Computer vision is a widely used tool to create software that can recognize text characters from images. The aim of the current work is to create functions that will aid in achieving better accuracy in scenarios where text is to be segmented out. The current work includes methods for binarization, skew correction and word segmentation in text based images.

This report briefly summarises the methods used in the Indic Tools project to binarize images, correct skew angle and segment words. A lot of the Mathematics behind individual methods has been omitted in order to provide an overview.

Keywords: *Computer vision, text images, word segmentation from images, binarization, skew correction*

1 Introduction

The demand for softwares that process text based images for information extraction has risen as camera technology has improved. Another breakthrough is the increasing use of smartphones that allow users to conveniently save images; this has been long used to share images of documents on paper, pamplets, books, etc.

The Indic Tools project aims at annotating images of text documents that are originally written in Indian languages, such as Hindi, Tamil, Sanskrit, etc. by employing pattern matching of words over the same image file as well as other image files. Upon annotating a word, the program looks for similar patterns as the selected word in the image and tells the user that the matched patterns could possibly have the same meaning. This would make annotation in documents easier. The objective of the project is to provide an interface for users to share information that isn't directly available in English in images of text. This report explains the methods used to binarize images, to correct skew angles in images and to segment words in images.

- For binarization of relatively noisy images, a sequence of steps, including histogram equalization, edge detection and adaptive thresholding are used.
- Words are segmented based on contours, and noise and/or outliers are eliminated statistically.
- The skew angle is corrected by taking a random sample from the set of detected words, and computing the individual angles of each word. The angles are approximated to the nearest whole number, and the median of these angles is taken as the skew angle of the image file.

For implementing these, OpenCV 2.4.8 wrappers were used along with Python 2.7. For all statistical analysis, Numpy library was used.

2 Methods

Various methods were tried to achieve the best results. Out of these, the following yielded good results.

2.1 Image Binarization

Binarizing an image refers to converting all the pixels in an image to either a high state or a low state, usually denoted by pixel values 255 for high and 0 for low (in standard 8-bit channel images). For image binarization, the algorithm used is as follows.

Algorithm1: Binarizing a text image

Input: 24-bit RGB image

Output: 2-bit binary image

Function:

1. Image is denoised using median blur
2. Convert RGB image to GRAYSCALE
3. Perform histogram equalization using CLAHE
4. Detect edges using Canny Edge Detection
5. Adaptive Gaussian Thresholding is performed on the image from step(2)
6. Mask image retrieved in step(3) using image in step(4) and invert

Image denoising is done using a grid size of $5 * 5$. Histogram equalization is performed using CLAHE (Contrast Limited Adaptive Histogram Equalization), with a clip limit of 2 and grid size of $8 * 8$ pixels. Canny edge detection is performed with gradient values between 100 and 200, inclusive. The thresholding on histogram equalized image is adaptive, to cater to images with different background and foreground colours. The figures below demonstrate the steps involved.

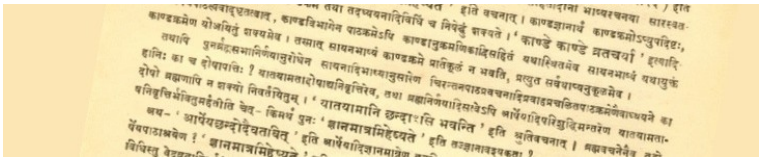

Figure 1(a): Original image (skewed)

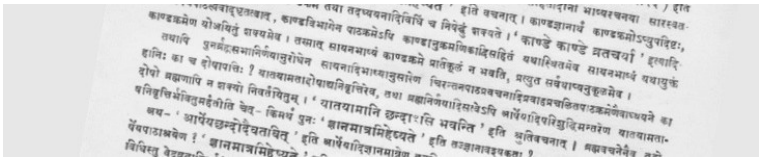

Figure 1(b): Original image converted to grayscale

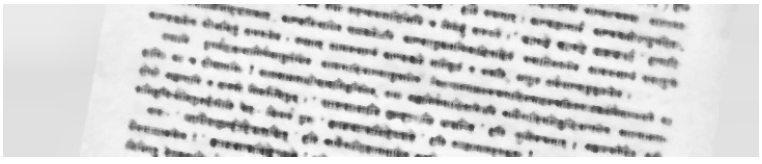

Figure 1(c): Grayscale image after CLAHE

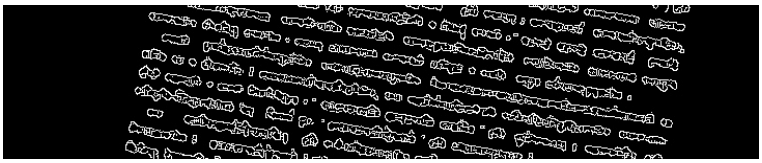

Figure 1(d): Canny edge detection

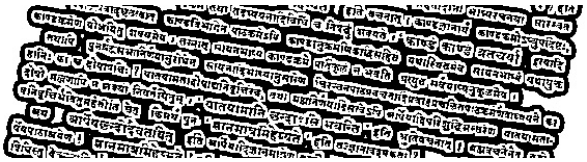

Figure 1(e): Adaptive Gaussian thresholding after CLAHE

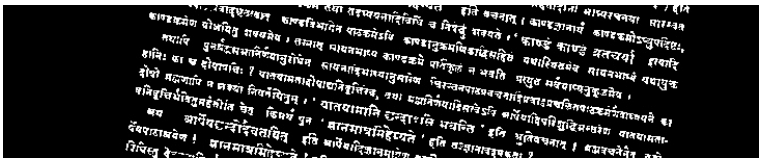

Figure 1(f): Result after masking images from steps (3) and (4) and inverting

2.2 Skew Correction

While taking an image input from a camera, the input may be skewed by an arbitrary angle, and this will eventually lead to errors in matching patterns of words. To avoid this, the program needs to automatically detect the skew angle and rotate the image accordingly, so the user (as well as the program) may view the image(s) of text without any apparent perception of skewing.

Algorithm2: Correcting skew angle in an image

Input: 2-bit binary image with a skew angle θ, $0° \leq \theta \leq 90°$

Output: 2-bit binary image with a skew angle $\theta \simeq 0°$

Function:

1. The Sobel gradient along the Y-axis of the binary image is computed, then thresholded.
2. The areas of all the contours are computed
3. The standard deviation of the areas is calculated
4. Contours are randomly sampled, such that their normalized areas lie in the range of $[-\sigma,+\sigma]$, where σ is the standard deviation of the areas.

5. The skew angles of each of the contours is calculated by approximating an *ellipse* around each of them. The angle of the major axis is taken as the skew angle.
6. The skew angles is approximated to their nearest whole number, and the *mode* of these angles is taken as ϕ
7. The entire image is rotated by an angle $\theta = 90° + \phi$

The Sobel gradient along the Y-axis is computed with a kernel size of 5.The following figures represent intermediate images created in the algorithm.

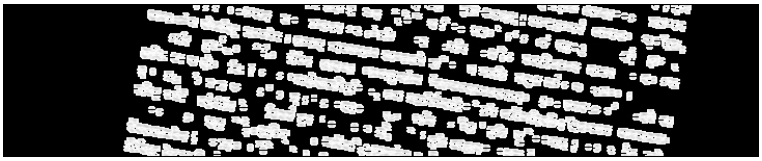

Figure 2(a): Sobel Y-gradient

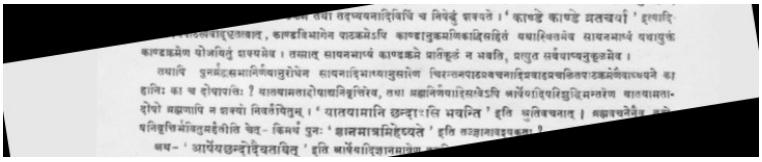

Figure 2(b): Skew correction applied to original image

2.3 Word Segmentation

Once the image is optimally binarized and corrected of any skew angle, the words can be detected using contour approximation.

Algorithm3: Segmenting words (contours) in an image

Input: 2-bit binary image with a skew angle θ, $\theta \simeq 0°$

Output: Locations of words in the given text-based image

Function:

1. Contours are detected in the image
2. Their coordinates are computed and saved

The following figure indicates detected words in a text-based image.

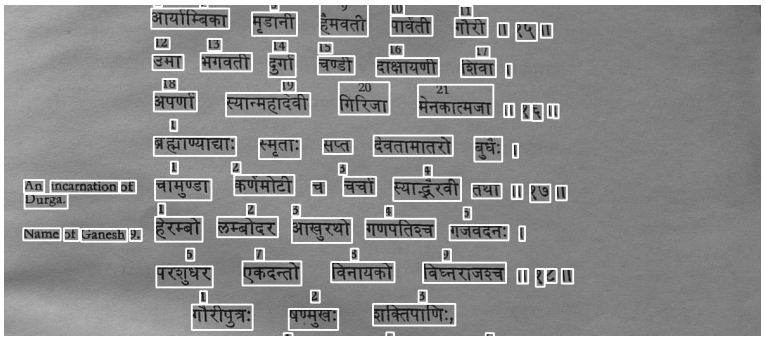

Figure 3(a): Indications of detected words in an image

3 Results and Conclusion

The current methods employed worked well on the assigned test images, but were unsatisfactory on images with low resolution. A constraint for the described algorithms to work well is that the images should be of good resolution. Much scope remains to make the algorithms used more efficient in terms of both execution time and memory used.

As future work, algorithms may be developed for character segmentation in text-based images which are in Indian languages, following a similar approach. This can further be scaled to create an OCR (Optical Character Recognition) software for Indian languages.

4 Acknowledgements

I would like to thank Shashank Chakravarty for considering my desire of wanting to work on *something big* and introducing me to Dr. Kanchi Gopinath; Dr. Gopinath for introducing me to the Indic Tools project; Milan Kumar Danga for being patient with me and personally guiding me through the project, both as a mentor and as a team mate, and finally, Sai Susarla for guiding the entire project as a whole.

5 References

1. OpenCV-Python official documentation
2. Dan S. Bloomberg and Luc Vincent: Document Image Applications