A Novel Approach of Tesseract-OCR Usage for Newspaper Article Images

***Chaitanya Tejaswi*[1*], *Dr. Bhargav Goradiya*[2], *Prof. Ripal Patel*[3]**

[1]*B. E. (Electronics & Communication, 2018), Birla Vishvakarma Mahavidyalaya, India*
[2]*Electronics & Communication Dept., Birla Vishvakarma Mahavidyalaya, India*
[3]*Electronics & Communication Dept., Birla Vishvakarma Mahavidyalaya, India*

Abstract

A novel approach for optical character recognition of newspaper article images (captured as smartphone camera images) is presented, with evaluation based on two sets of images; both captured using the same camera, under varying lighting conditions.

Keywords: *Optical Character Recognition (OCR), Binarization, Tesseract OCR, OpenCV.*

[*]***Author for Correspondence***: *Email: crtejaswi13@gmail.com*

INTRODUCTION

Our original goal was to create an Android app that gives a text to speech output for text extracted from printed newspaper articles, using open-source software tools. The goal was to achieve this for English language & extend it to Indian languages.
Such a solution is ideal for

- people with visual impairment,
- journalists, who intend to deal with specific stories,
- daily commuters, who wish to consume daily news in an audio form,
- students, who wish to compile notes from a variety of sources, et cetera …

just to name a few.
An Android application for the same should comprise of these major steps:

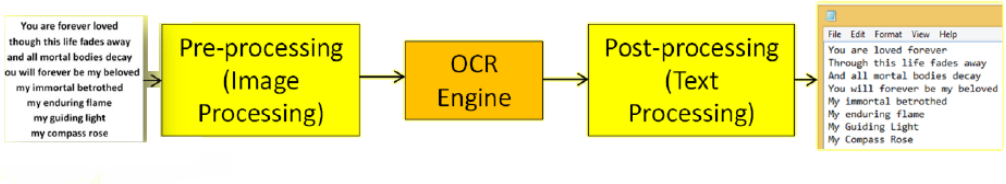

Fig. 1. Naive OCR System Process Diagram. Every OCR system is composed of these 3 elements.

Working on the concept for a month, we realized the shortcomings of this approach:

- Application complexity
- No improvements beyond a certain extent
- Random discrepancies

So, we decided to focus on the pre-processing stage instead.

PRE-PROCESSING METHODOLOGY

1. Improving the Image Quality

The ImproveQuality page on Tesseract-OCR's github repository offers an excellent insight into pre-processing techniques that can be used to improve the quality of text output.

In some of our experiments, we've observed that the need for post-processing using an open-source natural language processing toolkit such as NLTK can altogether be avoided, whereas in some cases, the character accuracy is barely 2%.There is no easy way to conclude the reason for such a discrepancy. However, we can look for reasons by analyzing the binarized image used by Tesseract-OCR to carry out character recognition ($tessdata.tif$).

2. Choosing the Binarization Scheme

For the pre-processing stage, we used OpenCV, which offers 5 binarization schemes:

- **THRESH_BINARY**

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{maxval} & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases} \quad \text{---------- (1a)}$$

- **THRESH_BINARY_INV**

$$\mathtt{dst}(x,y) = \begin{cases} 0 & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{maxval} & \text{otherwise} \end{cases} \quad \text{---------- (1b)}$$

- **THRESH_TRUNC**

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{threshold} & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{src}(x,y) & \text{otherwise} \end{cases} \quad \text{---------- (1c)}$$

- **THRESH_TOZERO**

$$\mathtt{dst}(x,y) = \begin{cases} \mathtt{src}(x,y) & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ 0 & \text{otherwise} \end{cases} \quad \text{---------- (1d)}$$

- **THRESH_TOZERO_INV**

$$\mathtt{dst}(x,y) = \begin{cases} 0 & \text{if } \mathtt{src}(x,y) > \mathtt{thresh} \\ \mathtt{src}(x,y) & \text{otherwise} \end{cases} \quad \text{---------- (1e)}$$

Fig. 2. Binarization schemes offered by OpenCV [15].

Out of these, $THRESH\_BINARY$ and $THRESH\_BINARY\_INV$ are most suitable for our application since newspapers usually have a single foreground-background (B/W) pair for each article.

Next, to choose the appropriate binarisation scheme, we have to try each technique on a set of images and look at $tessdata.tif$ in each case.

In our case, we tried global, adaptive, Otsu (and its variants) thresholding algorithms.

In literature, techniques of Sauvola [12] and Niblack [13] are recommended for use. However, in our experience, Otsu's technique also gives decent results, if light variations at the time of image capture have been compensated for.Working with bimodal images, Otsu's algorithm [14] attempts to find a *threshold value (t)* which minimizes the weighted within-class variance given by the relation (from [15]):

$$\sigma_w^2(t) = q_1(t)\sigma_1^2(t) + q_2(t)\sigma_2^2(t) \quad \text{---------- (2)}$$

Where the coefficients $\boldsymbol{q_i}$, means $\boldsymbol{\mu_i}$, and variances $\boldsymbol{\sigma_i}$ are given by (from [15]):

$$q_1(t) = \sum_{i=1}^{t} P(i) \quad \& \quad q_1(t) = \sum_{i=t+1}^{I} P(i) \quad \text{---------- (3a)}$$

$$\mu_1(t) = \sum_{i=1}^{t} \frac{iP(i)}{q_1(t)} \quad \& \quad \mu_2(t) = \sum_{i=t+1}^{I} \frac{iP(i)}{q_2(t)} \quad \text{---------- (3b)}$$

$$\sigma_1^2(t) = \sum_{i=1}^{t} [i-\mu_1(t)]^2 \frac{P(i)}{q_1(t)} \quad \& \quad \sigma_2^2(t) = \sum_{i=t+1}^{I} [i-\mu_1(t)]^2 \frac{P(i)}{q_2(t)} \quad \text{---------- (3c)}$$

It actually finds a value of *t* which, in the histogram plot, lies in between two peaks such that variances to both classes are minimum.

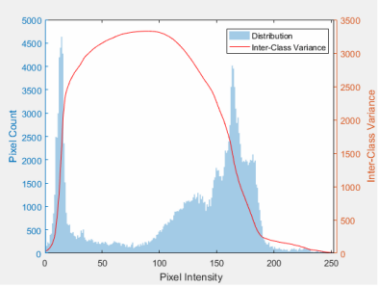

Fig. 3. Histogram Visualization for Otsu's Algorithm implementation.

DPI enhancement is done using OpenCV. This is done so as to match the minimum standard spatial resolution for paper printing, which is 300 dpi. By convention images captured by mobile devices or obtained on the internet are 72 or 96 dpi.

3. Upsampling Images for Improved Accuracy

The Gaussian Pyramid approach is used for upsampling the image after increasing its spatial resolution to 300 dpi. To do so, the image is convolved with the Gaussian kernel, given by (from [15]):

$$\frac{1}{16}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix} \qquad \text{-------- (4)}$$

4. Border Pixel Addition

Before binarizing the image, we need to center the contents of the image. To do this, we add border strips of white pixels, 0.75x and 0.5x the horizontal and vertical resolution of the image (assuming height > width in original image) to top and bottom regions of the image. This step is crucial for reducing OCR stage processing time.It can be carried either after the Cropping stage, or after the Upsampling stage. However, we've used it after Cropping stage, to compensate for the decrease in spatial resolution caused due to cropping the image.

5. Generate Text to Speech Output

To generate the TTS output from the obtained text, we use Google Text to Speech (gTTS) as gtts-cli.py, a common command line utility.

PROPOSED SCHEME

Steps involved in the proposed scheme have been described in the previous section. A block diagram representation is given as follows:

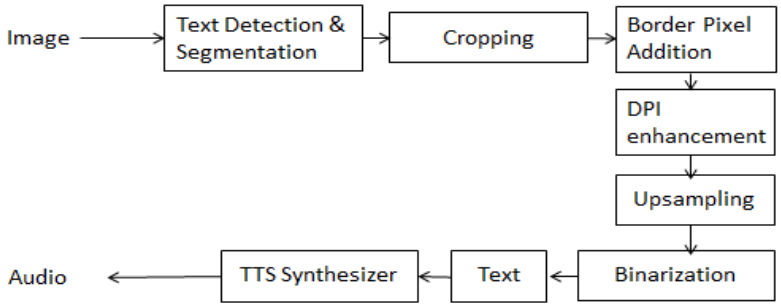

Fig. 4. Steps involved in implementation of proposed scheme. *Image* refers to segmented images of newspaper articles, while *Audio* refers to MP3 files generated by gTTS.

IMPLEMENTATION

Two sets of tests were conducted.

Both sets comprised of 10 images each, and consisted of newspaper article images obtained from Times of India (Ahmedabad) issues. Both sets were obtained under different circumstances, using the same smartphone camera. The two sets differed in the way the images were organized. The first set followed a random flow of content, whereas, in the second set, the images were grouped according to the layouts found in common practice.

Fig. 4(A). Images from the first set. Notice that the first two images have the same content. However, due to lighting variations, each leads to a different output.

Fig. 4(B). Images from the second set. Notice that the first 3 images have wide, single column blocks, while the next 3 images have thinner single column blocks.

RESULTS

In contemporary literature, two generic terms are used as measure of OCR accuracy – *Precision* & *Recall*. These two are defined as [7]:

$$precision = \frac{number\ of\ correct\ items}{number\ of\ items\ in\ OCR\ output} \quad \text{-------- (5)}$$

$$recall = \frac{number\ of\ correct\ items}{number\ of\ items\ in\ ground\ truth} \quad \text{-------- (6)}$$

Consequently, *recall* is the percentage of words in the original text correctly found by the OCR engine, while *precision* is the percentage of correctly found words with respect to the total word count of the OCR output.

We post the tabulated results for the two sets:

		Without Preprocessing		With Preprocessing	
	No. of Characters	Precision	Recall	Precision	Recall
Set 1	3809	14.33	22.47	56.27	48.24
Set 2A	3620	36.2	43.33	94.4	82.13
Set 2B	3513	44.28	48.27	69.86	78.37
Set 2C	3508	13.33	37.81	69.86	74.25
Set 2D	3318	15.75	32.47	64.24	68.92

Table 1: Character-wise Precision & Recall Percentages for the two datasets. Note that sets 2A,B have single columns, sets 2C and 2D have two and three columns of text,

LIMITATIONS & FUTURE WORK

Limitations

1. Limited to News Articles from/ a Single Newspaper with Consistent Fonts.

Times of India (*ToI*) is known to make use of *Times New Roman*, *Georgia*, and *Verdana* fonts. These fonts are consistent with the English language sets used for training Tesseract OCR. Although this makes it easier for us to have a starting point, it isn't indicative of how the system would fare with fonts from other newspapers. In addition, usage of cursive typeset fonts will worsen the problem further.
However, in practice, we have observed Tesseract OCR to give decent results, at least for English language text.
For example, refer the image below:

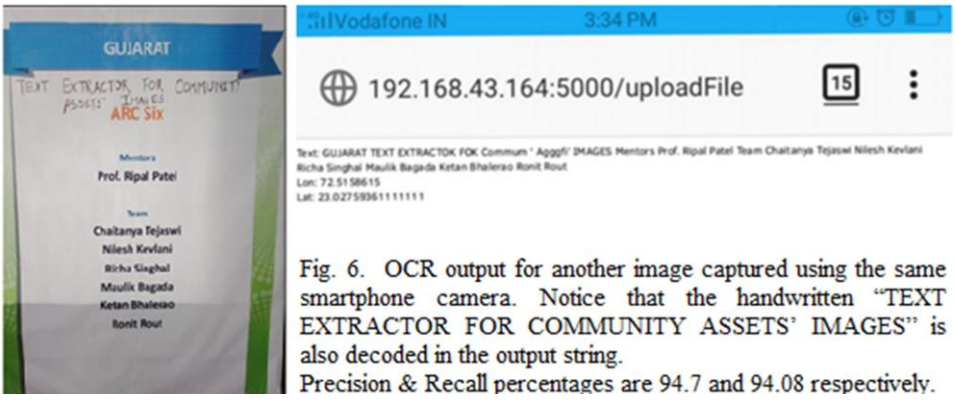

Fig. 6. OCR output for another image captured using the same smartphone camera. Notice that the handwritten "TEXT EXTRACTOR FOR COMMUNITY ASSETS' IMAGES" is also decoded in the output string.
Precision & Recall percentages are 94.7 and 94.08 respectively.

2. Inadequate Number of Test Images.

Our goal was to create a proof of concept, rather than a thorough investigation. Also, having a smaller dataset makes it easier to work with Ground Truth. For a more rigorous analysis, a larger labeled dataset can be created, and the entire process can be automated.

3. Angular Defects & Incident Light Intensity Variations

The most common roadblocks to using OCR tools on Natural Scene Images (JPEGs) are *angular defects* caused by tilt in the smartphone when holding it to capture the image, and *variations in illumination*.
The first can be corrected using *deskewing* techniques, while the second requires *a knowledge of camera parameters*. We have tried to refrain from these topics, in order to keep the analysis brief and succinct. [8] provides a novel approach to deal with this situation.

Future Work

1. Extension (of model) to multiple Indian languages

The scheme we have proposed can easily be extended to native Indian languages, Hindi & Gujarati being two of them. For this, one can use the pre-trained language sets made available by IndicOCR exclusively for Tesseract OCR. In addition, IndicNLP, a popular Indian languages text processing module for Python, can serve as a useful post-processing tool.

2. Extension to other newspaper formats

Here, we have worked with article images captured using smartphone cameras. In addition, we have the option of comparing results with *Scanned Images* and *ePaper versions* of a newspaper. This will complete an elementary discourse of OCR application for newspaper images.

3. Comparison & Objective Evaluation with other Binarization Techniques

The work we have presented makes use of Otsu's binarization. A more rigorous discourse would be to compare the performance with other acclaimed binarization techniques, in order to determine & fix some basic parameters to be taken care of, when preparing a dataset with the goal of achieving highly efficient output and system performance.

CONCLUSION

By making use of the proposed scheme on datasets of about 3500 characters each, we have observed a two-fold improvement in the character recognition efficiency of Tesseract OCR, in the case of digital newspaper article images secured using a smartphone camera.

REFERENCES

Patents:

[1] US6577762B1, Background Surface Thresholding

[2] US7400768B1, Enhanced optical recognition of digitized images through selective bit-insertion.

[3] US9298980B1, Image preprocessing for character recognition.

[4] US20120063690A1, Object-Based Optical Character Recognition Pre-Processing Algorithm.

[5] US7106905B2, Systems and methods for processing text-based electronic documents.

[6] US20130329023A1, Text recognition driven functionality.

Literature:

[7] Eugene Borovikov, A survey of modern optical character recognition techniques.

[8] M Seeger, C Dance, Binarising camera images for OCR (ICDAR 2001, Proceedings of the 6th International Conference on Document Analysis and Recognition).

[9] Ranjith Unnikrishnan, Ray Smith, Combined Script and Page Orientation Estimation using the Tesseract OCR engine (ICDAR '07 Proceedings of the Ninth International Conference on Document Analysis and Recognition).

[10] Ray Smith, An Overview of the Tesseract OCR Engine (MOCR '09 Proceedings of the International Workshop on Multilingual OCR).

[11] Ray Smith, Daria Antoniva, Dar-Shyang Lee, Adapting the Tesseract Open Source OCR Engine for Multilingual OCR (MOCR '09 Proceedings of the International Workshop on Multilingual OCR).

[12] JJ Sauvola, Tapio Seppänen, Sami Haapakoski, Matti Pietikäinen, Adaptive Document Binarization (ICDAR '97 Proceedings of the 4th International Conference on Document Analysis and Recognition).

[13] Niblack, W (1986), An introduction to Digital Image Processing, Prentice-Hall, pp. 115–116.

[14] Nobuyuki Otsu (1979). "A threshold selection method from gray-level histograms". IEEE Trans. Sys., Man., Cyber. 9

[15] OpenCV Reference Manual (OpenCV3.0 Documentation), pg. 265, 294-295.