ANALYSIS OF IMAGE PREPROCESSING TECHNIQUES TO IMPROVE OCR OF GARHWALI TEXT OBTAINED USING THE HINDI TESSERACT MODEL

Sukhbindra Singh Rawat, Ashutosh Sharma and Rachana Gusain
Department of Computer Science, Doon University, India

Abstract

A huge amount of information exists in the form of textbooks, paper documents, newspapers, and other physical forms, that is required to be digitized for its effective access and long-time availability. Optical Character Recognition (OCR) is an effective way to digitize the text. In this study, we have used Google's Tesseract as the OCR tool. The focus of our study is to improve Tesseract's accuracy on machine-printed Garhwali documents by using image pre-processing techniques including Super-Resolution (SR), different binarization methods (Otsu and adaptive thresholding), skew correction, morphological operations, and ImageMagick methods. To improve the Tesseract results, we used the three proposed approaches – two approaches differed by the binarization method (Otsu and adaptive thresholding), and the third approach used ImageMagick methods for pre-processing. For evaluation purposes, we created a dataset by capturing images from a sample of five Garhwali textbooks using two mobile cameras with different resolutions; two books were captured by a high-resolution camera and the other three were captured through a low-resolution camera. Our experiments showed good results in specific cases, for high-resolution images, 88.13% accuracy was achieved for Otsu thresholding without applying the Super-Resolution and for low-resolution images, 87.44% accuracy was achieved for ImageMagick with Super-Resolution.

Keywords:

Optical Character Recognition, Garhwali Language, Devanagari Script, Image Preprocessing, ImageMagick

1. INTRODUCTION

With the advancement of technology over the past few decades, information is getting stored in digital form over the internet or on computers. The benefit of storing information digitally is that people can have easier and wider access to this digitized information. There is a huge amount of information which exists in the form of textbooks, paper documents, newspapers, etc, that is required to be digitized for its effective access and long-time availability. There are two ways to digitize information; one way is to manually digitize it by typing which is a time-consuming task and requires a lot of human effort. Another way is to first scan or capture images of the documents (or textbooks) and then converting these images into digital text using the Optical Character Recognition (OCR). We have used Google's Tesseract OCR model. It is not only open source and free but also developed in Java which makes it platform-independent. It can recognize more than 100 languages and has support for Unicode (UTF-8). From 1985 to 1994 Tesseract was originally developed at Hewlett-Packard Laboratories Bristol and Hewlett-Packard Co (HP), Greeley Colorado. It was open-sourced by HP in 2005 [1]. From 2006 to November 2018, it has been developed by Google. The Long-Short Term Memory (LSTM) based version 4.11 was released on December 26, 2019 by Google. We have used the Tesseract v5.0.0-alpha.20201127 version in this work.

Garhwali is an under-represented language spoken in the Garhwal region of Uttarakhand, a state in north India. As per the 2011 census of India, there are 2.48 million Grahwali speakers and the language is currently considered a dialect of Hindi, an official Indian language. It belongs to the Central Pahari subgroup of the Northern Indo-Aryan family of languages [2]. The term 'Pahari' means belonging to the mountains. Not being a scheduled language of India, Garhwali is a highly low resource language in terms of digitized material. Because most of the Garhwali literature is available in print form, we wanted to investigate the usage of Tesseract OCR, the one trained for Hindi, in extracting the text from the sampled images of Garhwali textbooks. While both Hindi and Garhwali use Devangari script in their written form, there are certain character sets that are specific to the latter. Through this research work, we want to analyze the effect of various image preprocessing techniques to improve the OCRed results. We compare the performance of these techniques using quantitative measures. The error analysis is done at the word level. This is a first of its kind of work on Garhwali language.

The digitized text can be utilized in applications such as text to speech, translation by computer, and analysing the data in a database, etc. In order to study the effectiveness of image preprocessing methods in improving the OCR results, we captured images from some textbooks using two mobile phone cameras with different resolutions, one has high-resolution and another has low-resolution. Since text detection in images captured by digital or mobile cameras is a difficult task, skewed orientation, complex background, low-resolution, unfavourable text fonts, and sizes cause a lot of difficulties even for state-of-the-art text detection methods [3]. Therefore, image pre-processing is applied before passing it to OCR model.

The paper is organized as follows. Section 2 explains the related work that has already been done in the field of image pre-processing. Section 3 explains all the methods/techniques we have used for image pre-processing. Section 4 introduces the dataset, parameters, and the setup environment used to evaluate our proposed approaches and presents the accuracy results for these approaches. Finally, the outputs are concluded in section 5.

2. LITERATURE SURVEY

One of the important aspects of image pre-processing is image binarization. Global binarization methods such as Otsu's method [4] perform an automatic threshold selection. It evaluates the threshold value by maximizing the between-class variance of foreground and background. Sahoo et al. [5] did the comparative analysis of more than 20 global thresholding methods based on shape and uniformity measures. The study showed that Otsu's method performed well if more uniformity and better shape of the object is required in the binary image.

Local binarization methods such as Niblack's method [6] generally give better results for the images degraded with noise and uneven illumination. It evaluates threshold based on the local mean μ and standard deviation σ. The threshold T for pixel $p(x,y)$ is evaluated in a window centered at (x,y) as

$$T(x,y) = \mu(x,y) + k*\sigma(x,y) \qquad (1)$$

Trier and Jain [7] evaluated the performance of 11 local adaptive thresholding methods based on the OCR's ability to recognize handwritten numerals from hydrographical images. In this study, Niblack's method combined with the postprocessing step of Yanowitz and Bruckstein's method gave the best performance. Niblack's method is sensitive to the k value and it is difficult to find one that gives good results for all the images. Sauvola and Pietikainen proposed an improved version of Niblack's method. The modified method defines the threshold T as

$$T(x,y) = \mu(x,y) + [1+ k*\sigma(x,y)/R\text{-}1)] \qquad (2)$$

where $\mu(x,y)$ and $\sigma(x,y)$are as in Niblack's formula. R is the dynamic range of standard deviation, and k is the correction factor, which ranges from 0 to 1. This modification reduces the sensitivity of k and improves the results for document images with noise and uneven illumination.

According to Philips [8] thresholding sometimes may remove some pixels resulting in characters having holes in them. The thresholding may also join separate objects resulting in objects looking similar to blobs which are hard to interpret. These problems can be solved by applying morphological operations.

More recently, super-resolution reconstruction using deep learning methods has received great attention. Dong et al. [9], [10] proposed Super-Resolution Convolution Neural Network (SRCNN). SRCNN performs an end-to-end mapping directly between low resolution (LR) and high resolution (HR) images. It provides good performance but the computation cost is huge which hinders it from practical usage. Dong et al. [11] proposed a new network system named Fast Super-Resolution Convolutional Neural Network (FSRCNN). They re-designed the SRCNN network structure and accelerated the network by a factor of almost 40. Their method produced satisfactory results and was also time-efficient.

3. IMAGE PREPROCESSING

Preprocessing is a stage that can be used to improve the quality of the input image. It is required to remove or minimize different types of ambiguities and noises that occur during the scanning/clicking of an image and paper conditions so that the image can be accurately and efficiently processed by the OCR. A noisy image is more likely to be misread by an OCR model.

In our study, we have used two different methodologies, one that employs Otsu and adaptive thresholding and another using ImageMagick. The Fig.1 shows the steps required in generating the output (in text form) from the given input (an image). A detailed discussion on both the preprocessing methodologies is given in the following subsections.

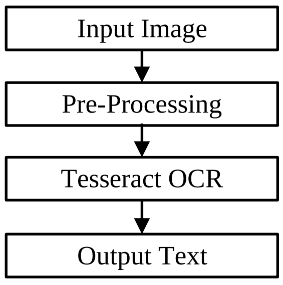


Fig.1. Steps to generate the output text from the input image

3.1 METHODOLOGY 1: OTSU AND ADAPTIVE THRESHOLDING

In case of Otsu's and adaptive thresholding, we have used the following steps to improve the quality of the image. The Fig.2 outlines the steps taken to employ this strategy. A brief discussion on the methods is given below.

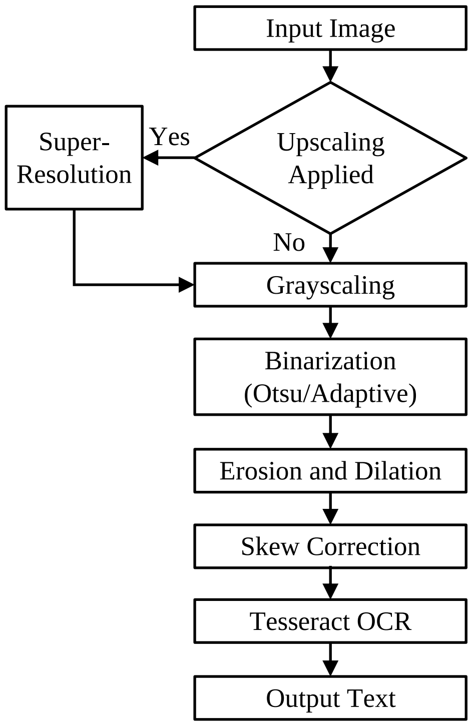


Fig.2. Methodology for Otsu and Adaptive thresholding

3.1.1 Super-Resolution:

OpenCV provides a very useful technique called Super Resolution (SR), which converts a low-resolution image to a desirable high-resolution image based on deep learning methods. It provides some pre-trained models that can be used to get desired results very easily and effectively. There are lots of pre-trained models available for SR like EDSR (Enhanced Deep Residual Networks for Single Image Super-Resolution), ESPCN (Efficient sub-pixel Convolutional Neural Network), and FSRCNN (Fast Super-Resolution Convolutional Neural Network). These models can upscale an image by a level of 2, 3, or 4. In our work, we have used the FSRCNN model [11]. It has the following important attributes. First, the last convolution layer is replaced with a deconvolution layer, then the mapping is learned from the LR image to the HR image. Second, the mapping layer is redesigned by shrinking the input feature dimension before mapping and expanding back afterward. Third, the model uses smaller filter sizes but more mapping layers. The activation values are computed using Parametric Rectified Linear Unit (PReLU). The model achieves a speedup of more than 40 times with even superior restoration quality. Table 1 shows the resolution of different images after super-resolving with FSRCNN.

Table.1. FSRCNN results for five different images

Image type	Initial Resolution	Upscaled Resolution
High-Resolution	2445 × 4096	7335 × 12288
High-Resolution	2380 × 3872	7140 × 11616
Low-Resolution	642 × 821	1926 × 2463
Low-Resolution	524 × 835	1572 × 2505
Low-Resolution	535 × 849	1605 × 2547

3.1.2 Grayscaling:

Grayscaling is a process that converts a continuous tone image to grayscale levels. Grayscaling converts a colored image to grayscale which can reveal more information than a normal colored one. Normally a computer can represent up to 256 levels of gray color. This process is widely used in a lot of real-time applications such as CCTV and traffic light cameras [12].

3.1.3 Binarization:

Binarization is the process of converting a grayscale image into a black and white image. This is done by selecting a threshold value and classifying all the pixels above and below this value [13]. Selecting threshold values has a great influence on the binarization process. Choosing a high threshold value has a higher chance of removing some of the character strokes and a low threshold may result in unclear boundary areas of characters [14].

There are generally two thresholding methods: global threshold and local threshold. In the global threshold method, as the name suggests, only one constant value is selected as a threshold value for the entire image. It works on the principle of estimating the image background level measured by using the intensity histogram, but this technique is not suitable for poor quality images because there may be a case where some portion of the image is very dark or light. On the contrary, the local technique segments the entire image into sectors and processes each sector individually which results in each sector having its own threshold value. Because of this way of processing the dark regions of the image are processed separately than the regions having lighter color intensity. This technique not only improves the digital image but also reduces its size. We have used the following two thresholding methods in our research work.

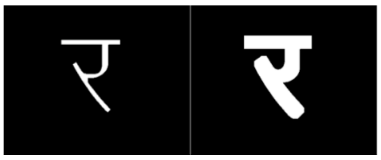
1. **Otsu's Method:** In global thresholding, we choose the threshold arbitrarily. On the other hand, Otsu's method avoids having to choose a value and determines it automatically by evaluating various characteristics of the entire image (like lighting conditions, contrast, sharpness, etc). It segments the elements of the image into two classes by minimizing the variance on each of the classes. The basic idea is to segment the image histogram into two clusters with a threshold value which is defined as a result of minimization of the within-class variance or maximization of the between-class variance [4].
2. **Adaptive Thresholding:** This method evaluates the threshold for a small region of the image based on the characteristics of its locality and neighbours. Different settings can be obtained by supplying different values to the parameters of this method like the adaptive method to be used, threshold value and its type, the block size that specifies the size of a pixel neighbourhood used to evaluate the threshold value.

3.1.4 Morphological Operations:

Morphological operations can be applied to grayscale images as their light transfer functions are unknown and therefore their absolute pixel values are not considered important or are of minor interest [15]. Morphological techniques examine an image with a small shape known as a structuring element. The structuring element is a small matrix of pixels (binary image) where each pixel has a value of zero or one [16]. Out of many morphological operations, some are described as follows.

1. **Erosion:** It thins or shrinks objects in a binary image and the extent of shrinkage is determined by a structuring element. The erosion forms a new binary image $g=f\ominus s$, where f is the input binary image, s is the structuring elements and g is the new binary image. The new binary image being produced consists of ones in all locations (x,y) of the structuring element's origin at which s fits the input image f, i.e., $g(x,y)=1$ if s fits f and 0 otherwise, for all pixel coordinates (x,y).
2. **Dilation:** It thickens objects in a binary image and the extent of thickening is determined by a structuring element. The dilation forms a new binary image $g=f\oplus s$, where $g(x,y)=1$ if s hits f and 0 otherwise.
3. **Closing:** The closing of an image is a dilation followed by an erosion, $f\cdot s=(f\oplus s)\ominus s$. It fills holes in the regions while keeping the initial region sizes. It is similar to dilation as it tries to enlarge the boundaries of the foreground regions in an image.
4. **Opening:** The opening of an image is an erosion followed by a dilation, $f\cdot s=(f\ominus s)\oplus s$. This operation tends to open up a gap between objects connected by a thin bridge of pixels. Any regions that have survived the erosion are restored to their original size by dilation. It is similar to erosion as it removes some of the foreground pixels from the edges of regions of foreground pixels. However, it is less destructive than erosion in general.

Original Binary Image



(a) Erosion (b) Dilation

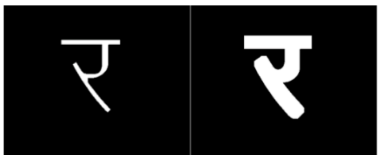

(c) Closing (d) Opening

Fig.3. Effect of morphological operations on the image of a Devanagari alphabet "ra"

The results of above morphological operations applied on the Devanagari alphabet "ra" is shown in Fig.3 above. See that erosion eliminates small-scale details from the image, whereas dilation fills in small incursions into the boundaries of a region, thereby reducing the gaps between different regions. Closing and opening, which combine the two, retain many details. In our study, we found out that our binarization methods were sometimes turning the characters into blobs. To solve this problem, we used to open as a morphological operation.

3.1.5 Skew Correction:

We have manually scanned the data from mobile cameras so there are chances that the images might be slightly skewed. So, in these cases, the efficiency of Tesseract may decrease, and to avoid such situations we have applied skew detection and correction using the Python library named deskew [17]. The deskew library uses Canny Edge Detection and Hough transform to determine the peaks, then it evaluates the deviation of each peak from 45-degree angles. The detected peaks are separated into bins, then the probable skew angle is chosen using the value in the bins.

3.2 METHODOLOGY 2: IMAGEMAGICK

ImageMagick [18] is a free and effective software delivered as a ready-to-run binary distribution or as source code. To implement ImageMagick, we have used Wand [19] which is a ctypes-based simple ImageMagick binding for Python. In the case of ImageMagick, we have used the following steps to improve the quality of the image. The order in which we have implemented the various steps in the ImageMagick preprocessing tool are outllined in Fig.4 and a brief desription of the methods is given below.

3.2.1 Resizing:

In this step, the input image is resized so as to increase its resolution.

3.2.2 Unsharp Masking:

This step aims to enhance text details and edges by using an unsharp masking filter. Sharpness describes the clarity of detail in a photo (document text in our case) and can be a valuable tool for emphasizing texture [3]. This method is used to sharpen the blurry edge of the characters present in the image. Here, we can control the blend between the filter and the threshold. The parameters used in this method are described below.

1. ***Radius***: defines the size of the area to sample.
2. ***Sigma***: defines the standard deviation.
3. ***Amount***: controls the sharpening ability of the filter.

if amount > 1, more sharpening filter is used.

if amount < 1, less sharpening filter is used.

4. ***Thresholding***: if thresholding is greater than 0, it reduces the sharpening quality.

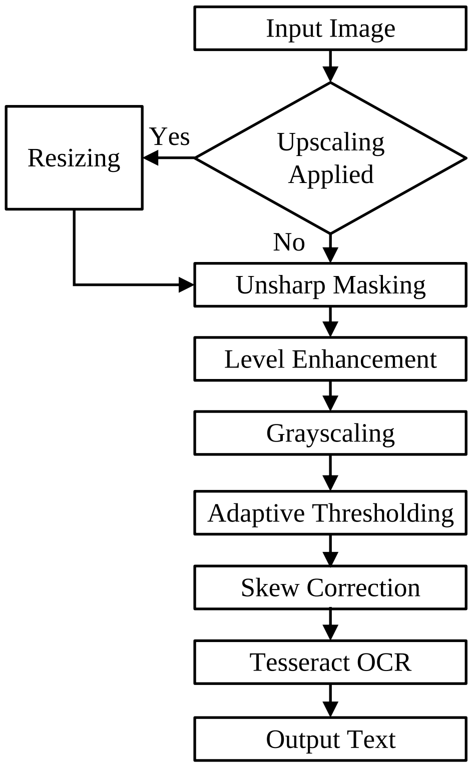


Fig.4. Steps applied in ImageMagick

The value of amount parameter plays an important role in sharpening the image but we should not take very large value of the amount (i.e., more sharpening) since it will result in unwanted noise in the image which can adversely affect the accuracy of the Tesseract model.

3.2.3 Level:

Level method is used to enhance the color of an image by controlling the black and white boundaries of an image. We define the following parameters to set the level.

1. Level range (l) gives two values white points and black points. So, for example, if white = 0.2 and black = 0.8 then we will have a level range [.2, .8]. Now, colors darker than the black point i.e., greater than 0.8 are set to 0 and colors brighter than the white point i.e., less than 0.2 are set to the maximum value.
2. Gamma (γ) is used to change the intensity of an image. A pixel is modified as pixel$^{(1/\gamma)}$. Higher the γ value (more than 1), lower the intensity of the resulting pixel.

3.2.4 Grayscaling:

Here, conversion of colored image into grayscale image is done using ImageMagick's transform_colorspace method.

3.2.5 Adaptive Thresholding:

As the name suggests, it adapts the threshold value for each pixel based on their neighbourhood. The area of the neighbouring pixels is defined by passing width and height arguments and an offset argument is used to apply a $+/-$ value to the current pixel. The method has following parameters.

1. Width gives the width of surrounding pixels.

2. Height gives the height of surrounding pixels.
3. Offset is used to change the current value of the pixels that can be added or subtracted from the current pixel.

3.2.6 Skew Correction:

This step aims to correct the skewness of the image. For skew correction, we again used the Python's deskew library.

4. EXPERIMENTS AND RESULTS

We used OpenCV library to perform Otsu thresholding, adaptive thresholding, and morphological operations, while Wand library is used to implement ImageMagick's pre-processing methods. We also used the hin.traineddata [20] file which is provided by Google's Tesseract OCR. The hin.traineddata file is used for training the Tesseract model to OCR Hindi text. Hindi is one of the two official languages of India and is written in Devanagari script. Since there is no specific OCR model trained on Garhwali language as of writing this paper, and the Garhwali language is also written in Devanagari script, we used hin.traineddata for the text recognition purpose.

4.1 DATASET

Our dataset has images from five Garhwali textbooks. Each book has different paper quality (background quality) and font style. Out of the five textbooks, two have been captured through high-resolution camera and the other three are captured through low-resolution camera. For experimental part, we have selected 10 random images from each book resulting in a sample of 50 images and a total of 15011 words. We also created five ground truth files by manually correcting all the mismatched or unrecognized words, so that we can compare these files with the output files generated by the Tesseract OCR. The approximate resolution of the selected images and the number of words present in the ground truth file for each of the five books are given in Table 2.

Table.2. Details of images and their sources

Book Title	Approx. Resolution	No. of words
Basumati	2380 × 3872	3748
Udrol	2463 × 4046	3507
Naubat	582 × 909	2362
Ab Kya Holu	516 × 837	2919
Kab Khulali Raat	524 × 835	2475

4.2 PARAMETERS

The accuracy of the pre-processing methods depends upon the values of the parameters. Thus, it is important to choose the optimal value of parameters to achieve good accuracy. In the case of Otsu thresholding, we do not have the option to work with parameters but in the case of adaptive thresholding and ImageMagick, we can adjust the parameters to improve the results of Tesseract OCR.

In adaptive thresholding, we have used adaptive threshold Gaussian method where the threshold value is the weighted mean of the neighbouring pixels minus the constant value. We can change the size of a pixel neighborhood and the constant value that is subtracted from the weighted mean or the mean value of neighboring pixels to calculate the threshold value. The value of the block size for both types of images is 51 and the constant value is 22 for HR and 7 for LR images.

In case of ImageMagick, we have resized each image to 3 times to maintain the same standard as used in the FSRCNN model. So, we have resized the HR images to 7500 × 11000 and LR images to 1600×2500. The same level of sharpening is used for both types of images. The radius of the area to be sampled is 8 with a standard deviation of 4 and the amount value 1. The same level of color enhancement is used for both the high-resolution and low-resolution images. We have used a level range of [0.2, 0.8] and gamma value 1.1 to change the intensity of each pixel in the image. For thresholding we have used a sample area of height 80 and width 80 for both types, but used different offset values -0.15 and -0.08 for HR and LR images respectively.

4.3 RESULTS

For calculating the number of errors, the text generated by different approaches is matched with the ground truth file. This is done by splitting the text on whitespaces (blanks and newlines) which gives us the lists of words present in the text files. Now, we need to calculate the number of missing words as well as the number of mismatched words between the ground truth file *g* and the OCR output file *t*. For quantifying the models' performance, we define the following terms.

The number of missing words MSW in t can be computed by finding the difference between the size of *g* and *t* as:

$$MSW = length(g)\text{-}length(t) \tag{3}$$

where *length*(*g*) and *length*(*t*) represent the number of words in the two files respectively. It is to be noted that the OCR converted text may fail to capture some words from the input image as well as it might also lead to the presence of extra words in the output text. The term MSW accounts for both missing as well as extraneous words in the output file.

After determining the difference in the number of words in g and *t*, we need to check the degree of matching in the two files. The number of mismatched words MMW is calculated by matching each word present in t to g and if no match is found, it means that the word is recognized incorrectly by the Tesseract OCR. We can then compute the total number of errors *E* as defined by Eq.(4)

$$E = \begin{cases} MMW + MSW & MSW > 0 \\ MMW & MSW \le 0 \end{cases} \tag{4}$$

Finally, the accuracy of the entire mechanism can be calculated as

$$Accuracy=(1\text{-}E/n)\times 100 \tag{5}$$

where *n*=*length*(*g*).

Using Eq.(4) and Eq.(5), we now evaluate the image preprocessing methodologies discussed in section 3 and report the results in the following subsections for images of different resolutions.

4.3.1 Results for High-Resolution Images

The number of errors and accuracy for HR images from Basumati and Udrol books are reported in Table.3 (without SR) and Table.4 (with SR). From Table.2, we obtain the total number of words n=7255. Direct application of Tesseract OCR to the HR images without any preprocessing give an accuracy of 86.78% with number of errors equal to 959.

Table.3. Results for HR images without SR

Method	Number of Errors	Accuracy
Otsu Thresholding	861	88.13%
Adaptive Thresholding	1115	84.63%
ImageMagick	1227	83.08%

Table.4. Results for HR images with SR

Method	Number of Errors	Accuracy
Otsu Thresholding	2763	61.91%
Adaptive Thresholding	4882	32.70%
ImageMagick	2713	62.60%

As we can see from the above tables that without applying super-resolution, Otsu thresholding performs better than the other two approaches with an accuracy of 88.13%. It is evident from the results that super resolution does not help if the input image is already a high-resolution image. The performance of all these approaches degrades resulting in an increase in the number of errors. This may be due to some reversal effects when SR is applied on images which already have high resolution causing the image quality to become poor. Amongst the three methods, Otsu thresholding performs best for higher resolution images. Also, the best accuracy reported with Otsu thresholding without SR is better than that of Tesseract OCR applied to the images without any preprocessing.

4.3.2 Results for Low-Resolution Images:

The number of errors and accuracy for LR images from Naubat, Ab Kya Holu and Kab Khulali Raat books are reported in Table 5 (without SR) and Table 6 (with SR). As before, from Table 2, we obtain the total number of words $n = 7756$. Tesseract OCR applied without any preprocessing to the LR images give an accuracy of 86.86% with number of errors equal to 1019.

Table.5. Results for LR images without SR

Method	Number of Errors	Accuracy
Otsu Thresholding	4421	42.99%
Adaptive Thresholding	4644	40.13%
ImageMagick	1399	81.96%

Table.6. Results for LR images with SR

Method	Number of Errors	Accuracy
Otsu Thresholding	1259	83.76%
Adaptive Thresholding	1332	82.82%
ImageMagick	974	87.44%

From Table.5 and Table.6, we observe that when super-resolution is not applied, the accuracy rate is very low for Otsu and adaptive thresholding but the results are quite good for ImageMagick. When super-resolution is applied, we can see the improvement by over 40% in the case of Otsu and adaptive thresholding and an improvement of 5.48% in the case of ImageMagick. Overall, ImageMagick with super-resolution performs better than the other two approaches in the case of low-resoluton images with an accuracy of 87.44%, which again is better than the Tesseract results without any preprocessing applied. The Fig.5 gives an overall comparison of all the methods in terms of the number of errors found in the output text for both HR and LR images for a quick analysis.

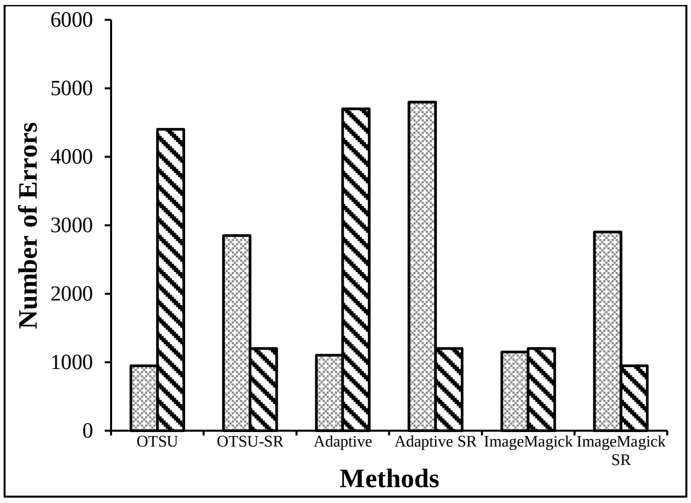


Fig.5. Error graph for HR and LR images using different approaches

5. CONCLUSION

In this work, we used Tesseract OCR for the purpose of OCRing Garhwali text. In order to evaluate the accuracy of the model, we took sample images from five different books. For each of these images, we obtained the OCRed text using the Tesseract model trained on Hindi language. In order to evaluate the efficiency of this model, compared with various image preprocessing techniques, we manually corrected the text files to match the input images. Obtaining these ground truth files took a significant amount of time. The time and effort needed to manually correct the OCR results can be significantly improved if we can apply appropriate preprocessing techniques. So, we experimented with the widely used preprocessing techniques, namely, Otsu, adaptive thresholding and ImageMagick, with and without super-resolution, to see their effect on the output performance. From the results obtained using these approaches, we can deduce that super-resolution performs well only on low-resolution images; using it on high-resolution images decreases the accuracy of the Tesseract OCR. We also found out that for high-resolution images, Otsu performs better than the others and for low-resolution images, ImageMagick performs better out of all the approaches we have used.

REFERENCES

[1] R. Smith, "An Overview of the Tesseract OCR Engine", Proceedings of IEEE International Conference on Document Analysis and Recognition, pp. 1-14, 2007.

[2] G. A. Grierson, "*Linguistic Survey of India*", Superintendent of Government Printing, 1916.

[3] A. El Harraj and N. Raissouni, "OCR Accuracy Improvement on Document Images Through a Novel Pre-Processing Approach", *Signal and Image Processing: An International Journal*, Vol. 6, No. 4, pp. 1–18, 2015.

[4] N. Otsu, "A Threshold Selection Method from Gray-Level Histograms", *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. 9, No. 1, pp. 62-66, 1979.

[5] P.K. Sahoo, S. Soltani and A.K.C. Wong, "A Survey of Thresholding Techniques", *Computer Vision Graphics and Image Processing*, Vol. 41, pp. 233-260, 1988.

[6] W. Niblack, "*An Introduction to Digital Image Processing*", Englewood Cliffs Publisher, 1986.

[7] O.D. Trier and A.K. Jain, "Goal-Directed Evaluation of Binarization Methods", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 17, No. 12, pp. 1191-1201, 1995.

[8] D. Phillips, "*Image Processing in C*", 2nd Edition, Mc-Graw Hill, 1994.

[9] C. Dong, C.C. Loy, K. He and X. Tang, "Learning a Deep Convolutional Network for Image Super-Resolution", Available at http://mmlab.ie.cuhk.edu.hk/projects/SRCNN.html, Accessed at 2014.

[10] C. Dong, C.C. Loy, K. He and X. Tang, "Image Super-Resolution using Deep Convolutional Networks", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 38, No. 2, pp. 295-307, 2016.

[11] C. Dong, C.C. Loy and X. Tang, "Accelerating the Super-Resolution Convolutional Neural Network", *Proceedings of IEEE International Conference on Computer Vision*, pp. 1-13, 2016.

[12] S. Badla, "Improving the Efficiency of Tesseract OCR Engine", Master Thesis, Department of Computer Science, San Jose State University, pp. 1-154, 2014.

[13] B. Sankur and M. Sezgin, "Survey over Image Thresholding Techniques and Quantitative Performance Evaluation", *Journal of Electronic Imaging*, Vol. 13, No. 1, pp. 1-2, 2004.

[14] K. Jindal, "Optical Character Recognition of Machine Printed Dogri Language Documents", Ph.D. Dissertation, Department of Computer Science and Engineering, Thapar Institute of Engineering and Technology, pp. 1-178, 2018.

[15] R.C. Patil and A.S. Bhalchandra, "Brain Tumour Extraction from MRI Images using Matlab", *International Journal of Electronics, Communication and Soft Computing Science and Engineering*, Vol. 2, No. 1, pp. 2277-9477, 2012.

[16] G. Priya and K. Nawaz, "Effective Morphological Image Processing Techniques and Image Reconstruction," *International Journal of Trend in Research and Development*, Vol. 4, No. 17, pp. 18-22, 2017.

[17] Github, "GitHub - sbrunner/deskew: Library used to Deskew a Scanned Document", Available at https://github.com/sbrunner/deskew, Accessed at 2021.

[18] Image Magick, "ImageMagick Documents", Available at https://imagemagick.org/index.php, Accessed at 2021.

[19] Wand, "Wand 0.6.6", Available at https://docs.wand-py.org/en/0.6.6/, Accessed at 2021.

[20] Github, "GitHub tesseract-ocr/tessdata: Trained Models with Support for Legacy and LSTM OCR Engine", Available at https://github.com/tesseract-ocr/tessdata, Accessed at 2021.