Title Page

Contents

[1 The Introduction 3](#_Toc57037490)

[2 Architecture of Tesseract 4](#_Toc57037491)

[Types of Preprocessing 5](#_Toc57037492)

[3.1 Thresholding techniques 5](#_Toc57037493)

[3.1.1 Simple Thresholding 5](#_Toc57037494)

[3.1.2 Otsu Algorithm 6](#_Toc57037495)

[3.1.3 Adaptive Thresholding 7](#_Toc57037496)

[3.2 Other preprocessing techniques 7](#_Toc57037497)

[3.2.1 Dilation 7](#_Toc57037498)

[3.2.2 Blurring 8](#_Toc57037499)

[3.2.3 Normalization 8](#_Toc57037500)

[4 Experimental Details (To edit) 10](#_Toc57037501)

[5 Results and Discussion (To edit) 12](#_Toc57037502)

[5.1 Thresholding (To edit) 12](#_Toc57037503)

[5.1.1 Simple Thresholding (To edit) 12](#_Toc57037504)

[5.1.2 Adaptive Thresholding (To edit) 13](#_Toc57037505)

[5.2 Other image processing techniques (To edit) 13](#_Toc57037506)

[5.2.1 Effects of dilation (To edit) 13](#_Toc57037507)

[5.2.2 Effects of blur (To edit) 13](#_Toc57037508)

[5.2.3 Effects of Normalization (To edit) 14](#_Toc57037509)

[5.3 Shadow Removal (To edit) 14](#_Toc57037510)

[5.4 Character and word visualization (To edit) 14](#_Toc57037511)

[Conclusion (To edit) 16](#_Toc57037512)

[Bibliography 17](#_Toc57037513)

# 1 The Introduction

Optical character recognition (OCR) is the conversion of printed text or handwritten text into editable text [1]. This technology allows machine to recognize text automatically. However, one of the difficulties faced by machines is differentiating similar characters such as the letter ‘O’ and the digit ‘0’. There is also the issue of bad contrast or noisy environments that will affect the accuracy of OCR. Applications of this technology include license plate recognition, text extraction from natural scenes, extracting text from scanned documents etc. Hence, many OCR systems were created, with the aim of maximizing accuracy. A system presented by Apurva A. Desai [2] is uses Artificial Neural Network (ANN) to recognize Guajarati handwritten numeral, achieving an avg of 82% recognition for Gujarati digits. U. Pal et al. [3] used Support Vector Machines (SVM) method for Bangla and Devnagari text recognition, achieving accuracy scores of 99.18% and 98.86% for Devnagari and Bangla characters, respectively. Bilal et al. [4] suggested local binarization method by using a Thresholding method and dynamic windows, which achieved F-mean values of 85.1% for handwritten text and 90.93% for printed text.

There are many types of OCR tools available in the current market, such as Desktop OCR, Server OCR, Web OCR etc. The accuracy of such systems ranges from 71% to 98% [1]. However, most of the OCR tools available are locked behind a pay wall, with only a few being free and open sourced. One such system is Tesseract, which is a free OCR software [5]. Tesseract is available on multiple platforms such as Windows, Ubuntu, Linux etc.

Tesseract performs best under the following conditions. Firstly, the document images must have a good contrast between the foreground text and the background. Secondly, the images must be horizontally aligned and scaled appropriately. Lastly, the image should not have blurriness and noise. However, it can be difficult to guarantee those conditions in practice. The better the quality of the image (size, contract, lighting), the better the recognition result. Therefore, this experiment aims to improve the accuracy of OCR on a given document image by preprocessing. The methods being tested include simple thresholding, Otsu algorithm, adaptive thresholding, and a combination of dilation, blurring and normalization.

# 2 Architecture of Tesseract

Tesseract works in an iterative manner as per block diagram in fig 1. The first step is adaptive thresholding, which transforms the image into a binary image. The decision threshold is determined based on a small region around it, resulting in different threshold values for different regions in the image. Compared to simple binary threshold, adaptive thresholding has better performance for images with varying illumination. The next step is connected component analysis, which is an algorithmic application of graph theory, where subsets of connected components are uniquely labelled, producing character outlines. Next, techniques for character chopping and character association are used to organize the outlines into words. This is done by organizing text into blobs, and the lines and regions are analyzed for fixed pitch and proportional text. Text lines are then broken into words based on their type of character spacing. The words then proceed into a two-pass word recognition process. The first pass attempts to recognize each word in turn. If the word is satisfactory, it is passed to an adaptive classifier as training data, which increases the accuracy of identifying each word down the line. In the final phase, the knowledge learnt from the training data will be used to resolve various issues and extract the text from the images. More details regarding every phase are available at [5].

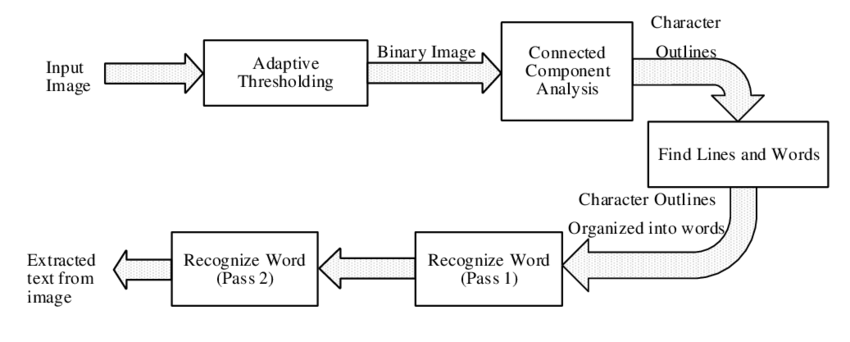


Fig. 1. Architecture of Tesseract OCR

# Types of Preprocessing

## 3.1 Thresholding techniques

In digital image processing, thresholding is usually done to segment the images, where the pixels of the image are changed to make it easier to analyze. A grayscale image is converted to a binary image, binary value of ‘1’ for white and ‘0’ for black, respectively. Most frequently, this is done to extract the point of interest from the image, as demonstrated by the main extracted using thresholding in fig 2.



Fig. 2. Simple thresholding on grayscale image of a man

### 3.1.1 Simple Thresholding

Simple thresholding is done by setting a threshold value of any constant between the intensity values of 0 (black in grayscale) and 255 (white in grayscale). Any pixel intensity below the constant is set to a value of ‘0’ and any pixel intensity above the constant is set a value of ‘1’ to binarize the image. An example of simple thresholding is in the fig 3 which has a threshold value of 75. From the histogram, we observe that the picture is distinctly separated into two groups. The group of pixels with low intensity is mostly found on the main character while the group of pixels with high intensity is mostly found in the background. This makes it easy to decide on the threshold, as seem by the red line in the second histogram, and thus being suitable for simple thresholding.

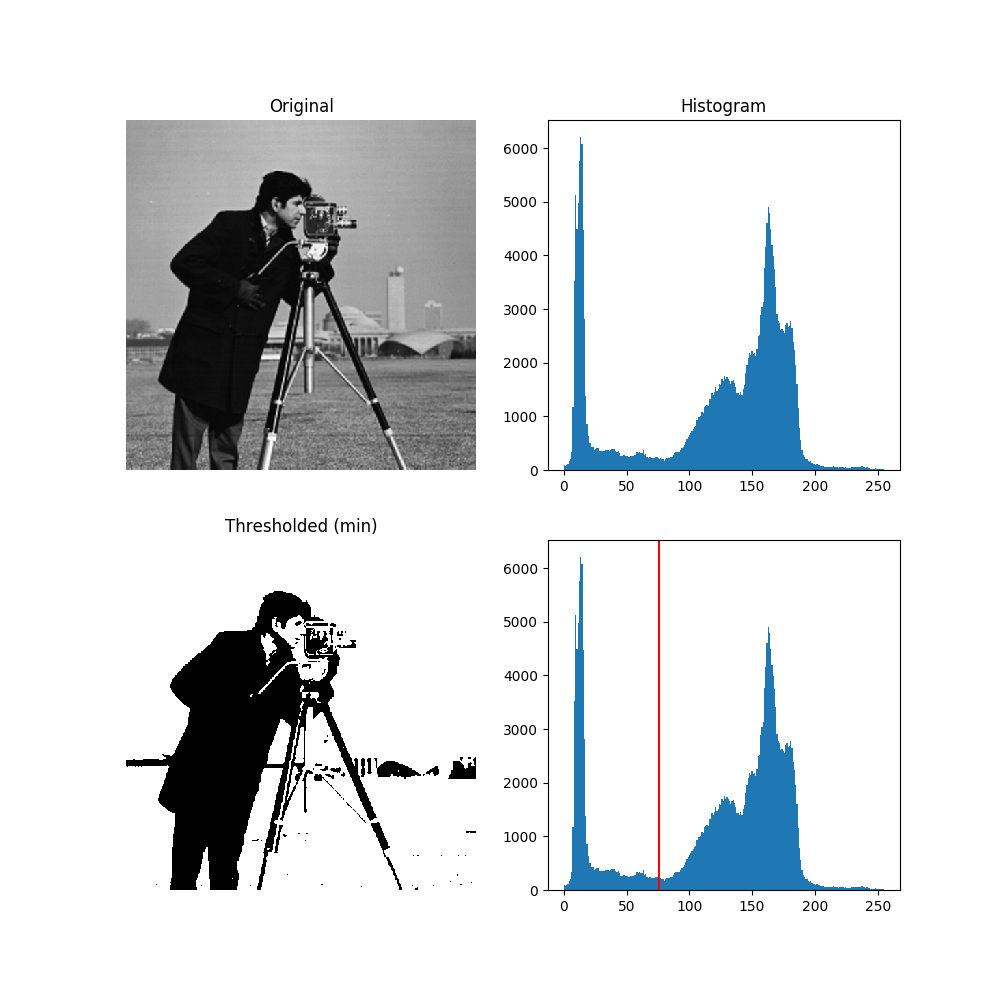


Fig. 3. Simple thresholding on grayscale image of a man

### 3.1.2 Otsu Algorithm

Otsu algorithm is used in conjunction with simple thresholding to perform automatic image thresholding. The Otsu method iterates through all possible threshold values and determines the optimal threshold by evaluating the inter-class variance between two classes, the foreground and background. The threshold that has the minimum inter-class variance will produce the best thresholding result. Fig 4 shows Otsu thresholding done on an image. The object of interest, the trees, are clearly defined after thresholding.



Fig. 4. Simple thresholding using Otsu Algorithm on a colored image of a tree

### 3.1.3 Adaptive Thresholding

Adaptive thresholding is done by calculating threshold values for smaller regions and having different thresholds for different regions around the image. There are two kinds of adaptive thresholding namely adaptive mean and adaptive gaussian. The difference between them is that adaptive mean takes the mean of the region and produces the threshold value while adaptive gaussian takes a weighted sum of the region and produces the threshold value. An example of adaptive thresholding is seen in fig 5. In the second image, we observe that simple(global) thresholding has difficulties extracting the object of interest due to the variance in light intensity. However, with both adaptive thresholding methods, the numbers and lines of the sudoku board are clearly defined.



Fig. 5. Adaptive thresholding done on a grayscale sudoku image

## 3.2 Other preprocessing techniques

As adaptive thresholding will already be done in the first stage of Tesseract OCR, three other image processing techniques are explored. They are dilation, blurring, and normalization. By combining these techniques, we hope to achieve even higher accuracy.

### 3.2.1 Dilation

Dilation is a type of morphological operation, which adjusts the shape or morphology of features in an image. Morphological techniques probe an image with a small shape or template called a structuring element. The structuring element is positioned at all possible locations of the image and compared with its corresponding neighborhood of pixels. Changes are made to the image based on if the structuring element fits or hits the section of the image. For dilation, a layer of pixels will be added to both the inner and outer boundaries of non-zero regions. This will reduce the gaps between regions and possibly fill holes in the characters. An example of dilation is demonstrated in fig 6.

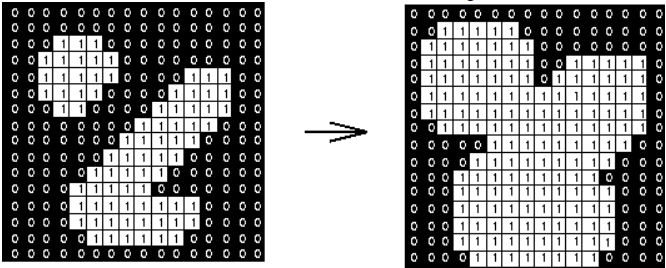


Fig. 6. Dilation: a 3×3 square structuring element  
(www.cs.princeton.edu/~pshilane/class/mosaic/).

### 3.2.2 Blurring

Blurring or smoothing the image removes “outlier” pixels that may be noise in the image. In a blur, we consider a square group of pixels surrounding a selected center pixel. This group of pixels, called a kernel, is moved along the image, adjusting the center pixel as it traverses. In averaging blur, the center pixel will be replaced with an average value of its surrounding pixels, marked by the kernel. A second type of blur is a Gaussian blur, which places more weight on the pixels closer to the center. Lastly, there is median blur, which takes the median value in the kernel to replace the selected pixel. As in fig 7, median blur is highly effective against salt-and-pepper noise, which is present in our samples. Hence, it will be used in our experiment.

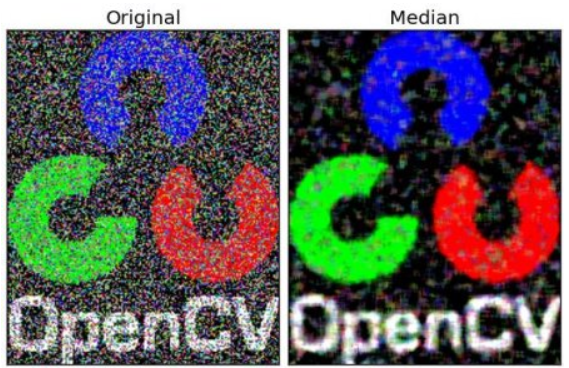


Fig. 7. Median Blur with a 5x5 kernel

### 3.2.3 Normalization

In image processing, normalization refers to changing the range of pixel values in an image. Applications include images with poor contrast due to glare or shadows, which is present in our samples. Normalization is also known as contrast stretching or histogram equalization. With higher contrast, finding the edges that form words will be easier and Tesseract will be able to identify the words more accurately. As seen in the example in fig 8, the features of the face are more prevalent as compared to its background after normalization, especially in dark areas such as the jaw on the left of the image.



Original

Normalized

Fig. 8. Example of normalization on a human face

# 4 Experimental Details (To edit)

* Intro
* Samples
  + 01 and 02
  + Focus on 01 for experiments
  + Accuracy comparison for 01 and 02 at the end
* Techniques used and what library we using for this
  + Simple thresholding: Binary Thresholding, Inverse Binary Thresholding, Truncate and Threshold, Threshold to Zero and Inverse Threshold to Zero
  + Simple thresholding with Otsu
  + Adaptive thresholding: Adaptive Thresholding using Mean and Adaptive Thresholding using Gaussian window
  + Others: Dilation, Blur, Abs Diff, Normalization
* Tesseract for character recognition
  + For accuracy, using imgtostr
  + For bounding boxes, character-imgtoboxes, word-imgtodata
* Sequence matcher from difflib for accuracy
* General code overview
  + Import img
  + Determine hyperparameters based on highest accuracy
    - Use exhaustive function
  + Run preprocessing techniques with best hyperparameters
  + Display image outputs
  + Find accuracy
  + Visually show bounding boxes and results for character and words for best simple thresholding, best adaptive thresholding and shadow removal combo
  + Compare and analyze
  + Repeat all steps for sample02 and compare the accuracy

In this experiment, we will be exploring and developing various image preprocessing techniques with the aim of increasing character recognition accuracy. The OCR used will be Tesseract. OpenCV, an open sourced image processing library will be utilized for various thresholding and image editing purposes. For accuracy evaluation, the python SequenceMatcher from difflib will be used. It compares a pair of character sequences and returns a ratio that represents similarity between the two sequences.

The sample images, as in fig 6 and fig 7, are printed documents with a shadow across the images, leading to poor contrast in certain regions. As previously mentioned, this can negatively affect the accuracy of OCR. Hence, the focus of this experiment will be preprocessing techniques.

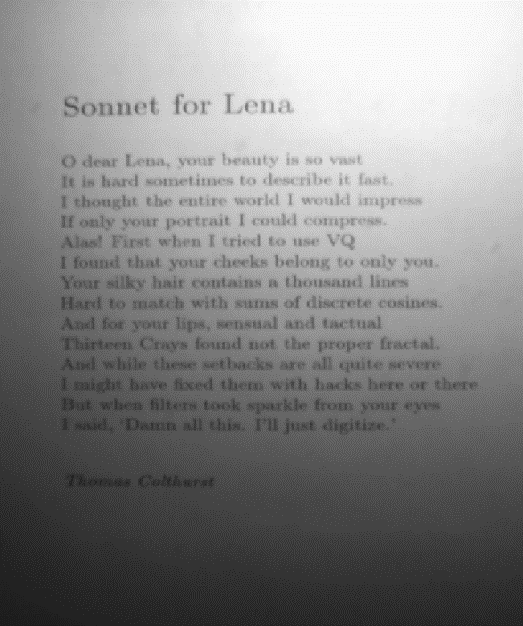
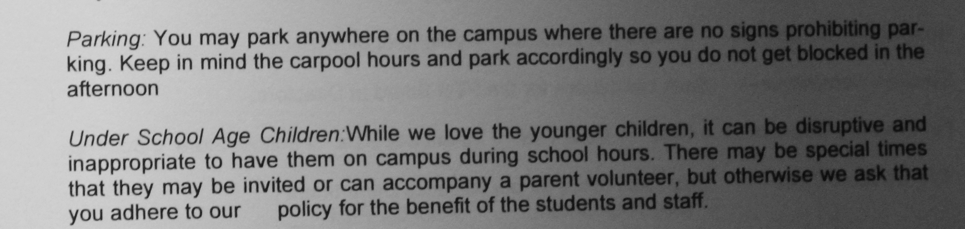


Fig. 6. “sample01.png”

Fig. 7. “sample02.png”

The simple thresholding techniques being tested are: Binary Thresholding, Inverse Binary Thresholding, Truncate and Threshold, Threshold to Zero and Inverse Threshold to Zero. Adaptive thresholding will also be used, namely: Adaptive Thresholding using Mean and Adaptive Thresholding using Gaussian window. Lastly, Otsu’s algorithm is used to choose the optimal threshold value for the above simple thresholding techniques.

Sequence of Code will be as follows:

1. Import images “sample01.png” and “sample02.png”
2. Convert color type of the images to grayscale
3. Perform Simple Thresholding and Adaptive Thresholding on the grayscale images
4. Visualize the results
5. Use Tesseract to convert text in the image into characters and words
6. Display resultant characters and words as an overlay of the original image
7. Calculate accuracy using SequenceMatcher with a preset correct sequence.
8. Repeat steps 3-7 for Thresholding with Otsu Binarization

[Mention how we combine the different techniques to achieve better accuracy]

[Mention how we find the parameters for dilation, blur, adaptive threshold]

# 5 Results and Discussion (To edit)

## 5.1 Thresholding (To edit)

### 5.1.1 Simple Thresholding (To edit)

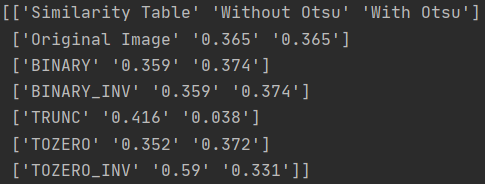
The results in fig 8 shows that the different simple thresholding techniques used does not produce images that has better contrast and less noise. Most of the images are affected by shadow in the original image and the best simple thresholding technique is “Truncate and Threshold” which ????. EDIT for me HAHA



Fig. 8. Results of different simple thresholding techniques



Simple Thresholding with Otsu





Double click to edit excel 😊

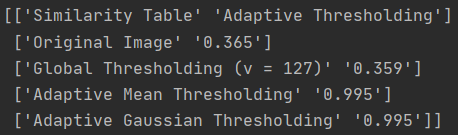
Discuss image outputs visually

Discuss Tesseract output (Character and word detection)

Discuss computerized accuracy output

### 5.1.2 Adaptive Thresholding (To edit)





## 5.2 Other image processing techniques (To edit)

Take the best image from simple and adaptive to work on (STH1, ATH2 and SR)

### 5.2.1 Effects of dilation (To edit)

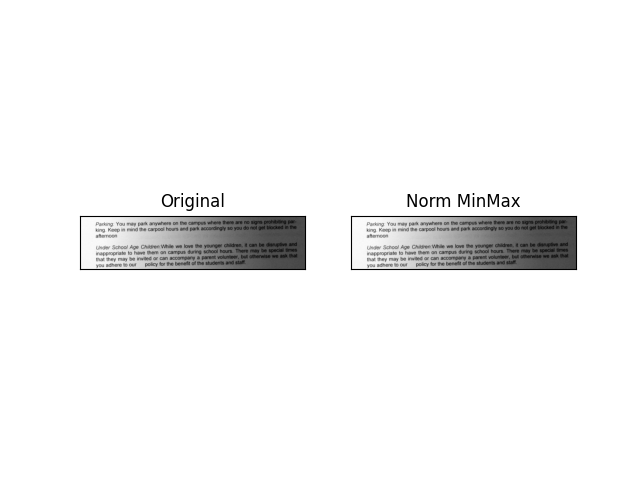
Test before and after thresholding?



### 5.2.2 Effects of blur (To edit)



### 5.2.3 Effects of Normalization (To edit)



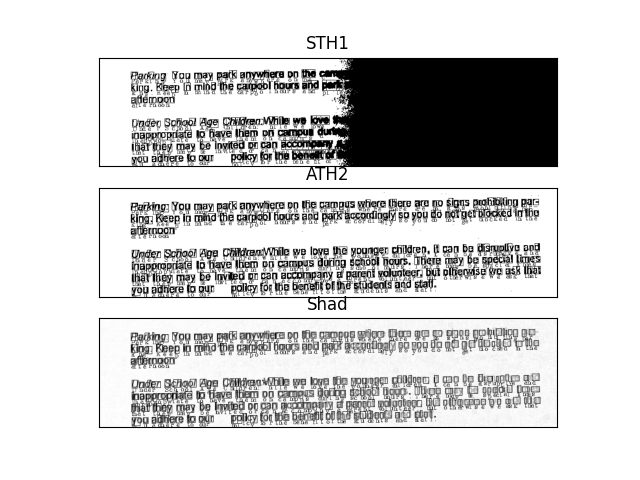
No diff??

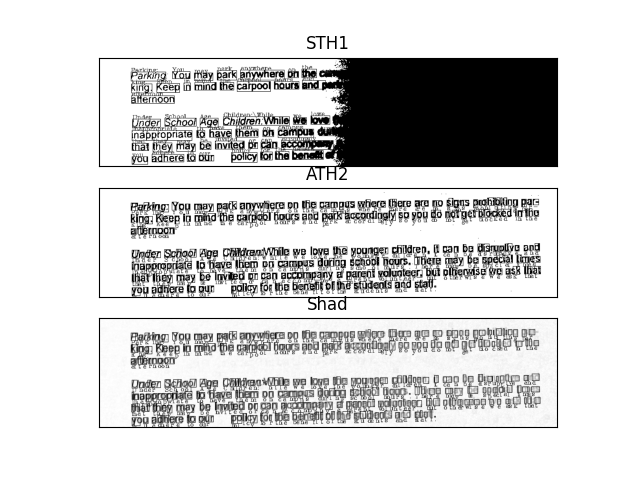
## 5.3 Shadow Removal (To edit)





## 5.4 Character and word visualization (To edit)





# Conclusion (To edit)

# Bibliography

|  |  |
| --- | --- |
| [1] | D. ARCHANA A. SHINDE, "Text Pre-processing and Text Segmentation for OCR," *International Journal of Computer Science Engineering and Technology,* pp. 810-812, 2012. |
| [2] | A. A. Desai, "Gujarati handwritten numeral optical character reorganization through neural network," *Pattern Recognition,* vol. 43, no. 7, pp. 2582-2589. |
| [3] | P. P. R. N. T. J. L. Umapada Pal, "Multi-oriented Bangla and Devnagari text recognition," *Pattern Recognition,* vol. 43, no. 12, pp. 4124-4136, December 2010. |
| [4] | S. N. H. S. A. K. O. Bilal Bataineh, "An adaptive local binarization method for document images based on a novel thresholding method and dynamic windows," *Pattern Recognition Letters,* vol. 32, no. 14, pp. 1805-1813, 2011. |
| [5] | R. SMITH, "An Overview of the Tesseract OCR Engine. In proceedings of Document analysis and Recognition," in *ICDAR, IEEE Ninth International Conference*, Curitiba, Brazil, 2007. |