Title Page

Contents

[The Introduction 3](#_Toc56522562)

[Architecture of Tesseract 3](#_Toc56522563)

[Experimental Details 5](#_Toc56522564)

[Thresholding Techniques 5](#_Toc56522565)

[Other preprocessing techniques 6](#_Toc56522566)

[Dilation 6](#_Toc56522567)

[Results and Discussion 7](#_Toc56522568)

[Conclusion 8](#_Toc56522569)

[Bibliography 9](#_Toc56522570)

# 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.

# 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].

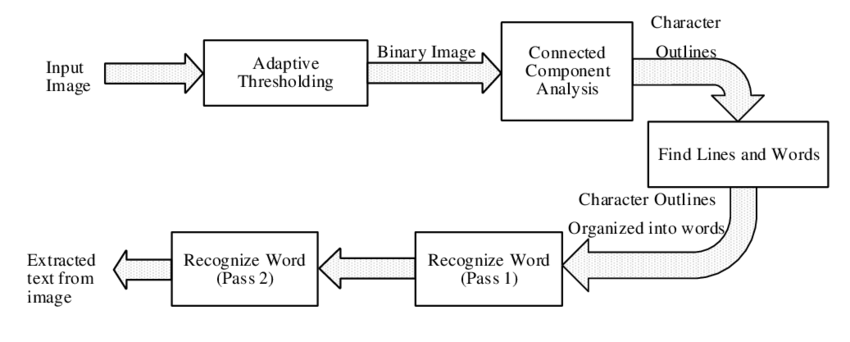


Fig. 1. Architecture of Tesseract OCR

# 3 Experimental Details

## 3.1 Thresholding Techniques

In this experiment, we will be exploring and developing various image preprocessing techniques such as binarization algorithm, 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 pairs of sequences of any type and returns a ratio that represents similarity between the 2 sequences.

The sample images, as in fig. 2 and fig. 3, 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, preprocessing is essential. The focus of the experiment will be thresholding techniques.

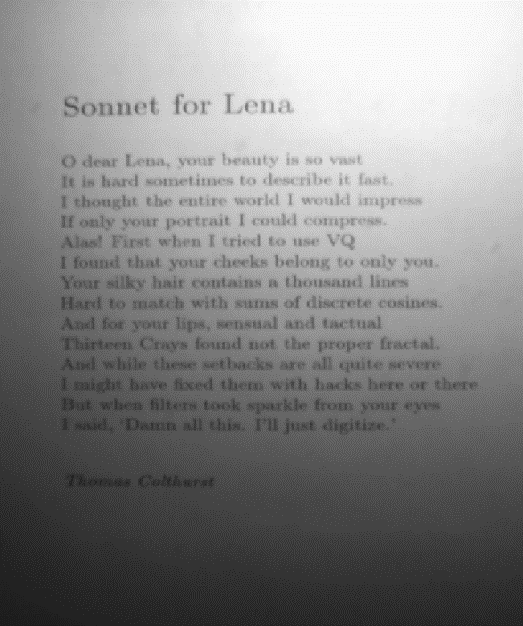
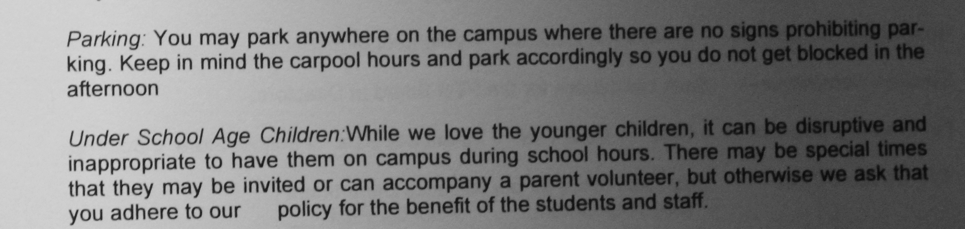


Fig. 2. “sample01.png”

Fig. 3. “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: Binary Adaptive Thresholding, 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 thresholding methods.

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

## 3.2 Other preprocessing techniques

As 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. 4.

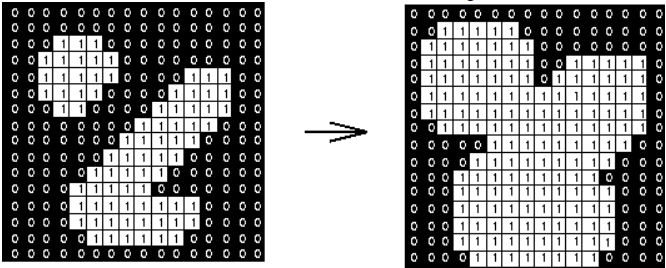


Fig. 4. Dilation: a 3×3 square structuring element  
(www.cs.princeton.edu/~pshilane/class/mosaic/).

### 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. 5, median blur is highly effective against salt-and-pepper noise, which is present in our samples. Hence, it will be used in our experiment.

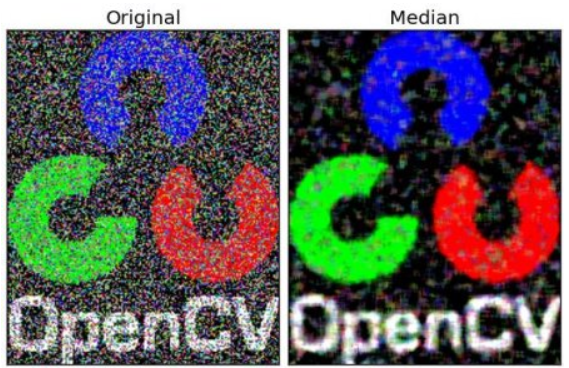


Fig. 5. Median Blur with a 5x5 kernel

### 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. 6, the features of the face are more prevalent as compared to its background after normalization.



Original

Normalized

Fig. 6. Example of normalization on a human face

# 4 Results and Discussion

## 4.1 Simple Thresholding

## 4.2 Adaptive Thresholding

## 4.3 Other image processing techniques

### 4.3.1 Effects of dilation

### 4.3.2 Effects of blur

### 4.3.3 Effects of Normalization

# Conclusion

# 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. |