



TRIBHUVAN UNIVERSITY  
INSTITUTE OF ENGINEERING  
PULCHOWK CAMPUS

A  
PROJECT REPORT  
ON  
EASY READ

**SUBMITTED BY:**

NIKESH D.C. (PUL075BCT052)  
RAVI PANDEY (PUL075BCT065)  
ROHAN CHHETRY (PUL075BCT066)  
YUKTA BANSAL (PUL075BCT096)

**SUBMITTED TO:**

DEPARTMENT OF ELECTRONICS & COMPUTER ENGINEERING

May, 2023



त्रिभुवन विश्वविद्यालय  
Tribhuvan University  
इन्जिनियरिङ्ग अध्ययन संस्थान  
Institute of Engineering  
पुल्चोक क्याम्पस  
Pulchowk Campus  
इलेक्ट्रॉनिक्स तथा कम्प्युटर इन्जिनियरिङ्ग विभाग  
Department of Electronics and Computer Engineering

553 4070  
552 1260  
Extn: 315  
Fax: 977 1 555 3946  
email: [ece@ioe.edu.np](mailto:ece@ioe.edu.np)  
<http://doece.pcampus.edu.np>

Pulchowk, Lalitpur  
P.O. box-1175

The undersigned certifies that they have read and recommended to the Institute of Engineering for acceptance of a project report entitled "Easy Read" submitted by Nimesh D.C., Ravi Pandey, Rohan Chhetry, Yukta Bansal in partial fulfillment of the requirements for the Bachelor's degree in Electronics & Computer Engineering.

Supervisor

**Bibha Sthapit**

Assistant Professor

Department of Electronics and Computer

Engineering,

Pulchowk Campus, IOE, TU.

External examiner

**Bibek Ropakheti**

Director

Aadhar group Pvt Limited,

Associate Professor

Pokhara University

Date of approval: May 18, 2023

# Copyright

The author has agreed that the Library, Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering may make this report freely available for inspection. Moreover, the author has agreed that permission for extensive copying of this project report for scholarly purposes may be granted by the supervisors who supervised the project work recorded herein or, in their absence, by the Head of the Department wherein the project report was done. It is understood that the recognition will be given to the author of this report and to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering in any use of the material of this project report. Copying or publication or the other use of this report for financial gain without approval of to the Department of Electronics and Computer Engineering, Pulchowk Campus, Institute of Engineering and author's written permission is prohibited.

Request for permission to copy or to make any other use of the material in this report in whole or in part should be addressed to:

Head

Department of Electronics and Computer Engineering

Pulchowk Campus, Institute of Engineering, TU

Lalitpur, Nepal.

# Acknowledgments

We would like to take this opportunity to extend our heartfelt appreciation to our project supervisor, **Bibha Sthapit** Maam, for her invaluable leadership and support throughout our project. Her knowledge, guidance, and support have been crucial to the accomplishment of this endeavor. Our technological expertise and problem-solving skills have been greatly influenced by her breadth of knowledge and expertise. We are appreciative of the time she has invested in teaching us and giving us helpful criticism. She has been incredibly inspirational with her persistence, focus on the little things, and desire to help us through each stage of the process.

We would like to thank **Department of Electronics and Computer Engineering** for providing this opportunity and resources for us to be able to complete this project in time. The knowledge, expertise, and dedication shown towards us have been instrumental in fostering an environment of learning and growth. It has helped us to push our boundaries and strive for excellence in our respective fields of study.

We would also want to express our gratitude to **LIS Nepal** for choosing us to be trained by their extremely talented professionals and having confidence in our abilities. At LIS Nepal, we were inspired and given very diverse perspectives on our concept, which enabled us to improvise our project. We would like to express our gratitude to all the other teachers who have guided us along the road for imparting their wisdom, experience, and insights. Their efforts have been crucial in forming our knowledge of the topic and enhancing our technical proficiency.

Our academic path has reached a major turning point with this endeavor, and we are grateful to have had the chance. We would like to thank everyone else who contributed directly or indirectly to our endeavor from the bottom of our hearts. Their assistance and support have been crucial in assisting us in achieving our objectives.

# Abstract

Most of the information we perceive is through our vision. Visual impairment can hamper day to day task performing capability of an individual and among one of such crucial task is learning. On this note, to provide an aid in independent learning for visually impaired person, the system has been developed. The system captures image of textual documents using a mobile application. As the image is captured by people who are visually challenged, the image is likely to have distortions such as shadows and uneven illuminations. The captured image is sent to a remote server for pre-processing, involving binarization, noise reduction, and layout analysis to make it suitable for OCR processing. The pre-processed image is fed to an OCR engine to obtain textual format. The extracted text is then used to generate speech as final output, enabling people to hear the text in the captured image.

Keywords: *Visually impaired, Learning aid, OCR, Binarization, Noise removal, Layout Analysis*

# Contents

- Page of Approval** ii
- Copyright** iii
- Acknowledgments** iv
- Abstract** v
- Contents** vii
- List of Figures** viii
- List of Abbreviations** ix
- 1 Introduction** 1
  - 1.1 Background . . . . . 1
  - 1.2 Problem statements . . . . . 2
  - 1.3 Objective . . . . . 2
  - 1.4 Scope . . . . . 2
  - 1.5 Application . . . . . 2
- 2 Literature Review** 3
  - 2.1 Binarization . . . . . 3
    - 2.1.1 Otsu’s Thresholding . . . . . 4
    - 2.1.2 Sauvola’s thresholding . . . . . 6
  - 2.2 Integral Image . . . . . 7
  - 2.3 Segmentation . . . . . 7
    - 2.3.1 Connected Component Analysis . . . . . 8
  - 2.4 Noise Reduction . . . . . 9
    - 2.4.1 Morphological Operations . . . . . 10
    - 2.4.2 Border Noise Removal . . . . . 10
  - 2.5 Page Dewarping . . . . . 11

<b>3 Methodology</b>	<b>12</b>
3.1 System Block Diagram	12
3.1.1 User connection	12
3.1.2 Request Handling	13
3.1.3 Core module	13
3.2 Sequence Diagram	14
3.3 Binarization	15
3.3.1 Execution time Improvement	16
3.3.2 Dynamic parameter	16
3.4 Noise Reduction	18
3.4.1 Segmentation	18
3.4.2 Component Filtering	18
3.5 Reading Order Determination	19
3.6 Post Noise Removal	20
3.7 Phone Application for Image Capture	20
3.8 Framework Selection	21
3.9 Server Structure	21
<b>4 Results &amp; Discussion</b>	<b>23</b>
4.1 Binarization	23
4.2 Reading Order Determination	26
4.3 OCR Result	27
<b>5 Future Enhancements</b>	<b>29</b>
References	30
Appendix	32

# List of Figures

2.1	Information present for extracting text is equal in all cases <i>Source: LRDE</i>	
	<i>Dataset</i> . . . . .	3
2.2	Binarization using difeerent thresholding techniques . . . . .	4
2.3	Pixel connectivity . . . . .	8
2.4	Image segmentation by Connected Component Analysis . . . . .	9
2.5	Noise in text images . . . . .	10
3.1	System Block diagram . . . . .	12
3.2	System Sequence diagram . . . . .	14
4.1	Comparison of execution times with and without the use of integral image for	
	binarization . . . . .	23
4.2	A histogram plot for F1 scores obtained by use of predicted 'k' values . . . . .	24
4.3	Noise Reduction . . . . .	25
4.4	Character Error Rates for Noisy and Denoised Inputs . . . . .	25
4.5	Segmenting an article for reading order, emphSource: Gatos P. (II) . . . . .	26
4.6	Segmenting a magazine for reading order, emphSource: LRDE Dataset . . . . .	26
4.7	Comparision of CER for Tesseract and EasyRead . . . . .	27
4.8	Some sample images taken for OCR evaluation . . . . .	28



# List of Abbreviations

**OCR** Optical Character Recognition

**TTS** Text To Speech

**WHO** World Health Organization

**RGB** Red Green Blue

**CER** Character Error Rate

**CCD** Charge Coupled Device

**API** Application Programming Interface

**URL** Uniform Request Locator

**IoT** Internet of Things

**LRDE** Electronics and Radar Development Establishment

**WSGI** Web Server Gateway Interface

**GTTS** Google Text-To-Speech

**DIBCO** Document Image Binarization Competition

**JSON** JavaScript Object Notation

# 1. Introduction

## 1.1 Background

Visual impairment is a broad term describing a wide range of conditions involving loss of visual functions such as visual acuity, accommodation, field of vision, color vision and adaptability to light. It can be caused due to various reasons such as hereditary, congenital, adventitious or disease related (2). People belonging to all age groups can suffer from visual impairment. Visually impaired children often face difficulties in learning which hampers their mental development. According to the WHO, there are about 2.3 billion people have some form of visual impairment worldwide, which represents one-third of the world population.

Humans are fairly adapted to perceiving their environment by eye as it is reflected by the complexity of our vision compared to many other animals. Thus, information available in today's society is mostly centered around having a proper functioning vision and we have evolved to get and provide information opted for our eye. People, who are deprived of the visual cues that normal person uses for completing their day to day task, find it difficult to perform even seemingly easy tasks. Among those are gaining information from books, newspaper or in general any textual content. The advancements in modern technology have produced some products targeted to help visually impaired people but they remain inaccessible by everyone. So, we have proposed a system to aid in this process for the visually impaired people.

Visually impaired people can be allowed to perform tasks normally by providing them the information about things around them in verbal form. The proposed system will partially solve this problem by allowing visually impaired people to "read" things around them. The system will be able to identify textual content in the user's field of vision and then convert it into speech. The conversion of text from pictorial form to speech is a complex task. In the proposed system, we have divided this task into smaller components comprising of text extraction from captured image using various OCR methods and then subsequent conversion of the text into speech using TTS component. The OCR component performs binarization after some-level of preprocessing of the still frames obtained from camera feed and then extracts characters and words by using segmentation techniques. The segmented words

are then identified as proper text by the OCR component. TTS is responsible for speech synthesis.

Some of the earliest known use of optical recognition technologies were for telegraphy and making reading devices for the blind. In 1914 Emanuel Goldberg developed a device that read characters and converted them into telegraph code (3). Around the same time, Edmund Fournier d'Albe designed an analog instrument labelled as optophone that allowed sightless person to “identify” some letters on a printed paper by changing the tone of a siren in response to the interruption of rows of light by the characters (4). In the 1970s, Ray Kruzweil created a reading machine for blind using his omni-font reader, text-to-speech synthesizer and flatbed CCD scanner (5). However, these devices were limited in their application as they required specific configurations for the device itself.

## 1.2 Problem statements

Along with difficulties in performing several day-to-day tasks, learning abilities of visually impaired people are greatly hampered and they are compelled to lean on others for their support. Visually impaired people especially blind people require specially tailored methods for instance, books containing Braille characters for reading. Braille textbooks are heavy and not easily available. Also, the solutions geared to help visually impaired people are not very affordable and widespread. Through this project we aim to enable visually impaired people to learn independently on their own.

## 1.3 Objective

To enable visually impaired people to learn from textbooks or any other textual source of information by developing an easy to use and economical system.

## 1.4 Scope

Though the proposed system is mainly intended for providing an assistance for learning to visually impaired people, similar or slightly modified systems can be developed in the long run, such as currency denomination identifier, real-time language translation.

## 1.5 Application

- Economical learning aid for visually impaired people
- Promoting independence for the visually impaired

# 2. Literature Review

Wide varieties of techniques have been implemented for extracting texts from image documents under OCR technologies. These methods vary on their intended use conditions, performance and so on. The process followed by most OCR technologies can be broadly divided into three steps: pre-processing, segmentation and classification/identification. Pre-processing can have great influence in final OCR result (II). It usually involves several methods and finally produces a binarized output of the input, less informative image having minimal non-textual noises.

## 2.1 Binarization

A image is represented as a matrix of pixels; each pixel being represented as three distinct values for red, green and blue (commonly referred to as RGB) components constituting the color of the pixel. However, for just distinguishing the texts from an image not all of these information is necessary. For reference, a gray-scale image consisting of only one-third of information (having same values for all red, blue and green components) can provide same amount of information as the original image for purpose of text extraction. Thus is why most of OCR techniques start with a gray-scale image as an input implicitly. In fact, we can push this a little further in that all we need for purpose of generating text from an image is that texts and background be clearly separated as distinct values. That is, if the image contains just two color values; possibly black and white then we will have same amount of information as a full colored RGB image or its grayscale variant as far as extraction of text from the image is concerned as shown in figure 2.1. This process of obtaining two value representation for the image is called binarization.

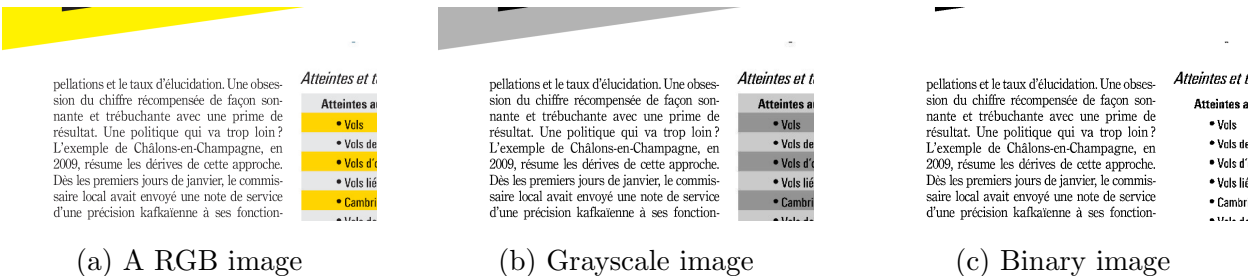


Figure 2.1: Information present for extracting text is equal in all cases  
 Source: LRDE Dataset

das ao primeiro subconjunto ou ao segundo subconjunto, ou seja, qual a causa da existência desses dois subconjuntos.

Voltando ao exemplo dos casos de uso. Não seria mais útil imaginar que seria possível classificar esses casos de uso em, por exemplo, simples, médios e complexos, para obter conjuntos de medidas com variância menor e, portanto, mais previsíveis? E se, por exemplo, os casos de uso simples pudessem ser programados em menos de um dia, os médios entre um dia e uma semana e os complexos em mais de uma semana? Seria mais fácil fazer previsões sobre o tempo que se levaria para desenvolver um sistema. Bastaria contar a quantidade de casos de uso simples, médios e complexos e multiplicar cada quantidade pela média de tempo associada a cada subconjunto. A informação seria mais relevante, possivelmente, do que uma única média aplicada ao conjunto inteiro dos casos de uso, pois com essa abordagem seriam usadas características dos elementos do conjunto para reduzir a incerteza sobre eles.

Porém, essa subclassificação introduz outro fator de incerteza: não se sabe, *a priori*, se a forma de determinar que um caso de uso é simples, médio ou complexo *realmente* classifica os casos de uso em subconjuntos nos quais os valores de tempo de desenvolvimento tenham variância mais baixa.

Nesse ponto poderia ser aplicada a experimentação científica para validar a hipótese de que uma determinada técnica de classificação dos casos de uso efetivamente classifica a complexidade destes adequadamente, ou seja, que esse sistema de classificação colocará no conjunto "simples" os casos de uso que efetivamente sejam mais rápidos de programar, e no conjunto "complexos" os casos de uso mais difíceis, ficando os demais no conjunto "médios".

Para testar essa hipótese é necessário comparar dois conjuntos de valores: o valor dado a um caso de uso pelo método de classificação e o valor do tempo que efetivamente se leva para programar o caso de uso. Para efetuar essa comparação é necessário usar o conceito de *covariância*, ou seja, determinar em que grau os dois conjuntos variam conjuntamente.

(a) A sample image

das ao primeiro subconjunto ou ao segundo subconjunto, ou seja, qual a causa da existência desses dois subconjuntos.

Voltando ao exemplo dos casos de uso. Não seria mais útil imaginar que seria possível classificar esses casos de uso em, por exemplo, simples, médios e complexos, para obter conjuntos de medidas com variância menor e, portanto, mais previsíveis? E se, por exemplo, os casos de uso simples pudessem ser programados em menos de um dia, os médios entre um dia e uma semana e os complexos em mais de uma semana? Seria mais fácil fazer previsões sobre o tempo que se levaria para desenvolver um sistema. Bastaria contar a quantidade de casos de uso simples, médios e complexos e multiplicar cada quantidade pela média de tempo associada a cada subconjunto. A informação seria mais relevante, possivelmente, do que uma única média aplicada ao conjunto inteiro dos casos de uso, pois com essa abordagem seriam usadas características dos elementos do conjunto para reduzir a incerteza sobre eles.

Porém, essa subclassificação introduz outro fator de incerteza: não se sabe, *a priori*, se a forma de determinar que um caso de uso é simples, médio ou complexo *realmente* classifica os casos de uso em subconjuntos nos quais os valores de tempo de desenvolvimento tenham variância mais baixa.

Nesse ponto poderia ser aplicada a experimentação científica para validar a hipótese de que uma determinada técnica de classificação dos casos de uso efetivamente classifica a complexidade destes adequadamente, ou seja, que esse sistema de classificação colocará no conjunto "simples" os casos de uso que efetivamente sejam mais rápidos de programar, e no conjunto "complexos" os casos de uso mais difíceis, ficando os demais no conjunto "médios".

Para testar essa hipótese é necessário comparar dois conjuntos de valores: o valor dado a um caso de uso pelo método de classificação e o valor do tempo que efetivamente se leva para programar o caso de uso. Para efetuar essa comparação é necessário usar o conceito de *covariância*, ou seja, determinar em que grau os dois conjuntos variam conjuntamente.

(b) Global Thresholding

das ao primeiro subconjunto ou ao segundo subconjunto, ou seja, qual a causa da existência desses dois subconjuntos.

Voltando ao exemplo dos casos de uso. Não seria mais útil imaginar que seria possível classificar esses casos de uso em, por exemplo, simples, médios e complexos, para obter conjuntos de medidas com variância menor e, portanto, mais previsíveis? E se, por exemplo, os casos de uso simples pudessem ser programados em menos de um dia, os médios entre um dia e uma semana e os complexos em mais de uma semana? Seria mais fácil fazer previsões sobre o tempo que se levaria para desenvolver um sistema. Bastaria contar a quantidade de casos de uso simples, médios e complexos e multiplicar cada quantidade pela média de tempo associada a cada subconjunto. A informação seria mais relevante, possivelmente, do que uma única média aplicada ao conjunto inteiro dos casos de uso, pois com essa abordagem seriam usadas características dos elementos do conjunto para reduzir a incerteza sobre eles.

Porém, essa subclassificação introduz outro fator de incerteza: não se sabe, *a priori*, se a forma de determinar que um caso de uso é simples, médio ou complexo *realmente* classifica os casos de uso em subconjuntos nos quais os valores de tempo de desenvolvimento tenham variância mais baixa.

Nesse ponto poderia ser aplicada a experimentação científica para validar a hipótese de que uma determinada técnica de classificação dos casos de uso efetivamente classifica a complexidade destes adequadamente, ou seja, que esse sistema de classificação colocará no conjunto "simples" os casos de uso que efetivamente sejam mais rápidos de programar, e no conjunto "complexos" os casos de uso mais difíceis, ficando os demais no conjunto "médios".

Para testar essa hipótese é necessário comparar dois conjuntos de valores: o valor dado a um caso de uso pelo método de classificação e o valor do tempo que efetivamente se leva para programar o caso de uso. Para efetuar essa comparação é necessário usar o conceito de *covariância*, ou seja, determinar em que grau os dois conjuntos variam conjuntamente.

(c) Local Thresholding

Figure 2.2: Binarization using different thresholding techniques

Broadly speaking, binarization encompasses process that transform data features of any entity into vectors of binary numbers usually to make classifier algorithms more efficient. The binary numbers are usually represented as 0 and 1. For image processing, in particular for OCR, it refers to conversion of grayscale (or colored) text containing image into equivalent sized image with two possible values for each of the pixels; 0 if the pixel is likely to be background and 1 if the pixel is likely to represent text component. This is performed in many instances by technique known as thresholding which is simply the process of comparing the image pixel intensity with a threshold value that is likely to separate background and textual pixels. The thresholding scheme may be based on many approaches (5); a broad view is whether a single threshold value is used for comparisons with all pixels also referred to as global thresholding or separate thresholds for each pixel/region; also referred to as local thresholding. Usually local thresholding fares much better compared to global thresholding as it tends to take into account local effects such as spot illuminations as shown by images 2.2. However local thresholding is also slow compared to global threshold as it has to ideally calculate separate threshold values for each pixel of the image and the time taken increases with increasing image size compared to almost constant for global threshold techniques.

### 2.1.1 Otsu's Thresholding

Otsu's thresholding (6) is a global thresholding method based on analysis of histogram of the pixel levels of the image. The histogram is simply the number of pixels corresponding to the pixel levels. For example, a histogram with 50 for pixel level of 129 indicates that there

are 50 pixels in image having gray-scale value of 129. The objective of this method is to separate the pixel levels histogram into two distinct classes corresponding to the foreground and background sets. These classes are separated about a pixel level (i.e. the threshold) such that the classes are maximally separated based on average between-class variance. The between class variance is conjectured to be a measure for “goodness” of thresholding. To select the optimal threshold level every pixel is first set to be the threshold separating the class and then the threshold that gives highest inter-class variance is taken as the optimal threshold. The maximally separating pixel value is the global threshold that is used to compare each pixel for binarization. The interclass variance is given by the equation

$$\begin{aligned}\sigma_B^2 &= \omega_0(\mu_0 - \mu_T)^2 + \omega_1(\mu_1 - \mu_T)^2 \\ &= \omega_0\omega_1(\mu_1 - \mu_0)^2\end{aligned}\quad (2.1)$$

$w_0$  is the proportion or probability of the first class and  $w_1$  is the proportion or probability of second class and is given as

$$\omega_0 = \sum_{i=0}^k p_i = \omega(k) \quad (2.2)$$

$$\omega_1 = \sum_{i=k+1}^L p_i = 1 - \omega(k) \quad (2.3)$$

Here  $p_i$  is the probability of occurrence of  $i^{\text{th}}$  level and  $n_i$  is the number of pixels at level  $i$

$$p_i = n_i/N, p_i \geq 0, \sum_{i=0}^L p_i = 1 \quad (2.4)$$

$\mu_T$  is the mean level of the entire image,  $\mu_0$  is the mean level of first class and  $\mu_1$  is the mean level of the second class

$$\mu_0 = \sum_{i=0}^k ip_i/\omega_0 = \mu(k)\omega(k) \quad (2.5)$$

$$\mu_1 = \sum_{i=k+1}^L ip_i/\omega_1 = \frac{\mu(T) - \mu(k)}{1 - \omega(k)} \quad (2.6)$$

$$\mu_T = \sum_{i=0}^L ip_i \quad (2.7)$$

The optimal threshold  $T$  is given as:

$$\sigma_B^2(T) = \max_{1 \leq k < L} \sigma_B^2(k) \quad (2.8)$$

Otsu thresholding however being a global thresholding technique is not adaptive to changes in images conditions and perform poorly in irregular shadow or illumination conditions. In fact fig [2.2b](#) is generated using Otsu’s thresholding whereas fig [2.2c](#) is generated using Sauvola’s thresholding which is a local thresholding technique.

### 2.1.2 Sauvola’s thresholding

A simple local thresholding method was proposed by Niblack [\(7\)](#). Niblack’s algorithm calculates the threshold ‘T’ for each pixel using the mean ‘m’ and variance ‘s’ over a rectangular window centered around it as

$$T = m + ks \tag{2.9}$$

The value of k is usually negative and depend on the type of documents or external conditions effecting the image. The effect of a negative value for the parameter ‘k’ is such that the threshold value is a value that is more towards the intensity or pixel value of character value i.e. pixel value of black or zero. A more refined approach proposed by Sauvola [\(8\)](#) is to modify the above thresholding value as

$$T = m * (1 + k * (s/R - 1)) \tag{2.10}$$

Here, ‘R’ is the dynamic range of standard deviation usually taken to be 128 and the value of k usually taken in range 0.34 to 0.5 [\(11\)](#).

Thus Sauvola’s approach requires two parameters ‘k’ and ‘w’ (the window size) which both seems to have considerable effect upon the final binarized output [\(9\)](#). In addition, the parameters may depend on the image conditions. Too small window size means that the window might contain mostly of the textual pixels leading to large texts being littered with holes as the algorithm expects window to have minimum contrast in window i.e. some contrast between foreground and background. Too large a window then the window might include objects of different nature leading to irrelevant statistics and thus poor binarization. Lower the value of ‘k’ more easily is the low contrast text regions identified, however unnecessary connections might be made between characters making identification difficult afterwards. Sauvola’s method is very sensitive to ‘k’ and ‘w’ and can perform better if tuned. An adaptive approach might be more suitable such as that proposed by Lazzara [\(9\)](#) where multiple window of different scales are used to take into account variations in text sizes across the image. For a document having somewhat uniform size of characters, the window size, if encompasses enough characters, can be set to a fixed value. The value of parameter ‘k’ however seems to depend largely on image conditions and a fixed value for entire document might not be appropriate for some particular image conditions. So, a more appropriate method would be to be able to replace it with some measure that takes into consideration local features

affecting the parameter.

We propose a novel approach for predicting the value of 'k' based on some properties of the image.

## 2.2 Integral Image

The summed-area table introduced by Crow(1984), and later modified as integral image is a matrix in which each element is the sum of all the all the elements above and to the left of it inclusive of itself. Such a data structure is used to quickly calculate the sum of values in a rectangular subset of a grid. The values of the integral image are calculated in a single pass over the image starting from the top left corner. The value in the integral image for position (x,y) is calculated as shown below:

$$\sum_{\substack{x' \leq x \\ y' \leq y}} i(x', y') = I(x, y) \quad (2.11)$$

The summed-area table can be computed efficiently in a single pass over the image, as the value in the summed-area table at (x, y) as:

$$I(x, y) = i(x, y) + I(x, y - 1) + I(x - 1, y) - I(x - 1, y - 1) \quad (2.12)$$

In order to calculate the sum of values in a rectangular grid in the original image, only four array references and simple addition/subtraction operations are required regardless of the array size. For a rectangular grid ABCD having the diagonal points at A(x<sub>0</sub>,y<sub>0</sub>) and D(x<sub>1</sub>,y<sub>1</sub>), the sum of all values of points in the grid is calculated from the integral image as:

$$\sum_{\substack{x_0 < x \leq x_1 \\ y_0 < y \leq y_1}} i(x, y) = I(D) + I(A) - I(B) - I(C) \quad (2.13)$$

The integral image is widely used in the Viola- Jones Object Detection Framework for calculating summation values needed for Haar-like Features as these features are rectangular. The use of integral image technique in this step greatly contributes to the overall performance of the Viola-Jones algorithm. Similar approach can be taken for calculating variance and standard deviation of any window size. For this we would require a sum of squared pixel values in addition to the basic summed image.

## 2.3 Segmentation

Image segmentation is the process of dividing an image into multiple segments or regions, each of which corresponds to a distinct object or part of the image. Some of these regions



maybe be background or border, some maybe an image within the original image. Since image segmentation is concerned with extracting specific components only, it is performed on a binaried image in order to reduce computation. There are a number of processes to perform image segmentation which work based on a variety of image features such as projection profiles, connected pixel sets, etc. In a textual image, the segmentation process can be done to extract individual paragraphs, lines, words or each individual character. By segmenting an image into smaller, well-defined regions, we can more accurately recognize and interpret the text within each segment. Also, analysis of features of the segmented components can be done to identify usefulness of the component.

### 2.3.1 Connected Component Analysis

Connected Component Analysis is a segmentation approach which identifies and labels connected regions within an image. For textual image segmentation, a connected component is a set of black pixels having at least one black pixel in it's neighbourhood. We can either consider the 4-neighbourhood or the 8-neighbourhood for determining each pixel's connectivity.

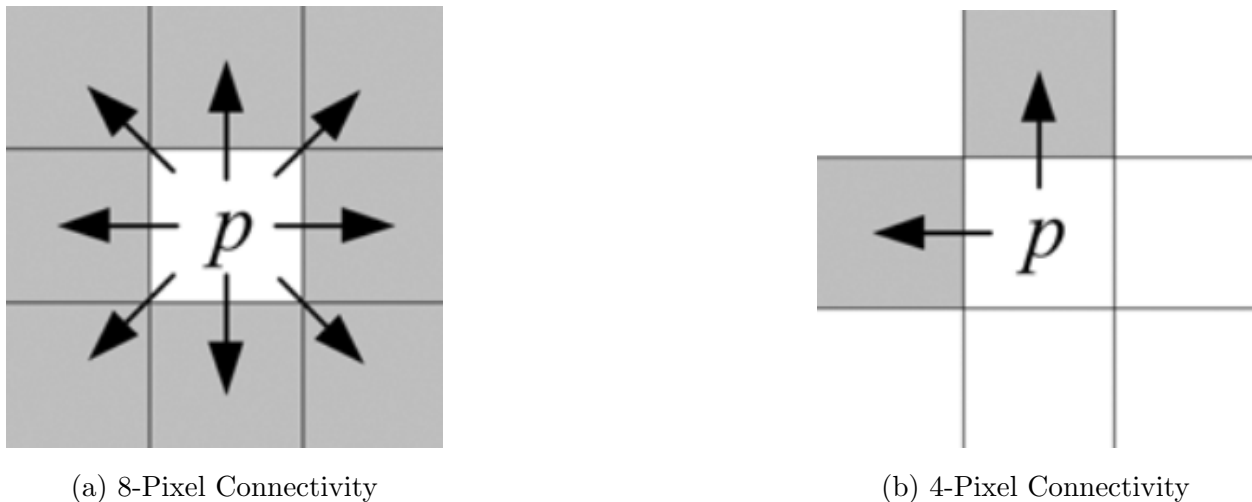
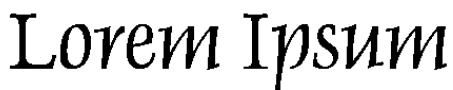


Figure 2.3: Pixel connectivity

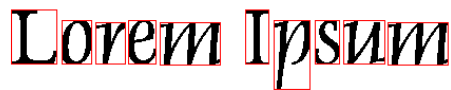
Various approaches based on connected component analysis have been proposed for image segmentation. As explained by Bouman (III), connected component analysis can be done on a binary image by separating each connected component by passing through the image starting from a seed point and following link between previously numbered components to form a connected set. This is known as region growing. In this technique the term “connected” can be defined as required and usually it is in accordance with either 4-point neighborhood system or 8-point neighborhood system. The connected sets are then numbered to extract

the connected components. The output of connected components analysis might contain many small disjointed regions. In order to improve the accuracy of segmentation, nearby regions can be merged into a single region. This is done by merging regions having region distance less than a certain threshold. The region distance function is calculated based on difference between different region features such as color, texture and shape of regions. This merging is done recursively by computing features of merged regions. This approach produces segmented regions with considerable accuracy.

In a segmentation approach presented by Ohlsson (12), the image is pre-processed to remove any skew using Hough Line Transform followed by Affine Transformation and also the contours of blobs are extracted by using Canny Edge Detection. The preprocessed binary image is used to extract the contour of blobs by applying a contour following algorithm as proposed by Satoski (13). The contour of all the blobs are determined by this algorithm and then Bounding Rectangle Algorithm is used to bound each contour into the smallest possible rectangles. The bounding rectangle algorithm used by Ohlsson is OpenCV's bounding rectangle which returns a rectangle for the input point set. The contour following algorithm as mentioned above does not distinguish between text and non-text and thus, bounding boxes for all blobs are obtained at this point. The rectangles which are extremely large or small are considered insignificant. The position of the remaining rectangles are stored and used for further steps as segment positions.



(a) Input Image

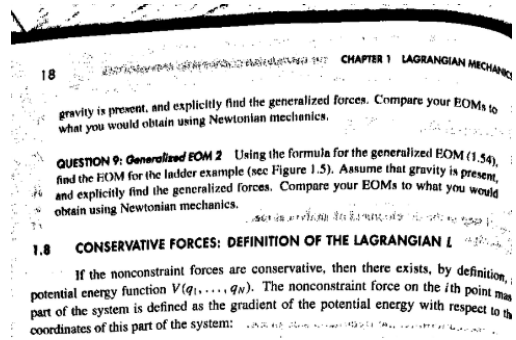
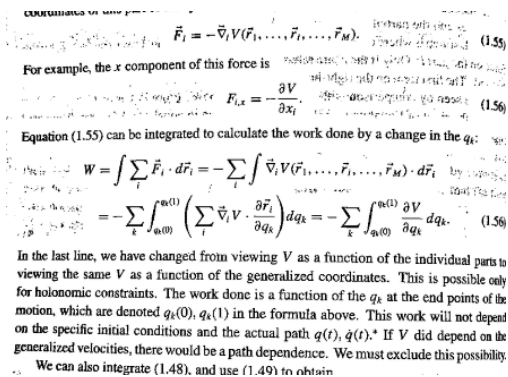


(b) Segmented Image

Figure 2.4: Image segmentation by Connected Component Analysis

## 2.4 Noise Reduction

Binarized image can still contain noise after thresholding as some can exceed threshold and consequently be considered as textual component. For binarized text images, the noise is often present in form of salt and pepper noise, which is due to uneven shadows present in the original image before binarization, and marginal noise, which is due to the separation of text document and background in the original image. Salt and pepper noise is mostly seen scattered as patches in the binarized image. However, these components are usually small compared to the characters and thus can be removed by peeling off layers from boundaries and then growing back the boundary which is referred in image processing as erosion (shrinking) and dilation (growing) respectively (14).



(a) Salt and Pepper Noise

(b) Marginal Noise

Figure 2.5: Noise in text images

## 2.4.1 Morphological Operations

An approach to noise removal by using mathematical morphological operations is suggested by Jamil (I15). Jamil's method suggests using a number of these morphological operations in a sequential combination to reduce noise in binarized images. Opening operation involves erosion followed by dilation while closing involves dilation followed by erosion. An operation to set a pixel as 0 or 1 based on the values of majority of pixels in its neighbourhood is known as majority. A number of combination of these operations were tested to observe the effect on noise in binarized images. This noise removal method however is not suitable for cases where the character size is not large in which case the initial erosion will remove too much information of the character contours and similarly subsequent dilation method will add too much pixel making it even worse. This property of dilation method however can be taken to advantage to allow for word level segmentation. By taking appropriate window size for the dilation operation we can cause nearby characters of the same words which are nearer than the characters of different words to clump together and then finally apply connected component analysis to obtain the bounds for each word. This spatial information can be later used to group the textual contents into words, lines, paragraphs and sections so as to obtain more contextual information which can be used for varied types of contents like that in newspapers and magazines.

## 2.4.2 Border Noise Removal

The binarized image may contain both textual and non-textual noise. Textual noises include undesired or cropped off text from neighboring pages of book. Similarly, non-textual noises include those from photos, background surfaces. These are commonly called border noise.

An approach for removing these border noises is proposed by Bukhari (16). Their method first uses a Gaussian filter to obtain a smoothed or blurred image from the black/white image. Then text lines are detected using some form of ridge detection algorithm. A ridge is considered to be likely a textual noise if it is “very” much near (they used a value of 25 pixels) to the left or right border of image and its length is smaller than 1/5th of the document width. After filtering these small noise ridges starting and ending points of remaining ridges can be determined which respectively give right and left border of the required text region. The starting and ending point may be approximated based on random sample consensus method as remaining ridges can also contain outlier noise. The topmost and bottom-most ridges define the next two page boundaries.

## 2.5 Page Dewarping

Images captured from pages of books or files are rarely completely flat. They are likely to be curved due to the large number of pages stitched together. Such curvature distorts the text characters and thus makes it difficult for the OCR engine to correctly recognize individual characters. If such wrapped images were to be made flat by applying a dewarping algorithm, the accuracy of the character recognition can be improved to a great extent. There are several approaches to page dewarping. A commonly used approach is to gather horizontal lines of text in the warped image and then obtain a warp or coordinate transform to make those lines parallel and horizontal. The horizontal cross-section of the page surface is modelled as a cubic spline whose endpoints are fixed at zero. The obtained texts are then assembled into spans, each span is assigned a number of keypoints. Then the points are projected onto a flat surface after pca analysis with rough initial estimates of required parameters. Through a series of iterations, the parameters are modified so as to improve the reprojction error and when the error is very low ( zero). The final parameters are then used to transform the entire image and a flat image is produced as a result.

# 3. Methodology

The components developed and steps undertaken under this project is explained below.

## 3.1 System Block Diagram

The overall layout of the system is shown in the figure 3.1. The system is constructed as a client server application which allowed for maximum flexibility in choice of platforms and framework for creating the system. Also massive inter-networking in today’s world and the view of increase in IoT devices and applications further strengthened the reason to choose client-server architecture.

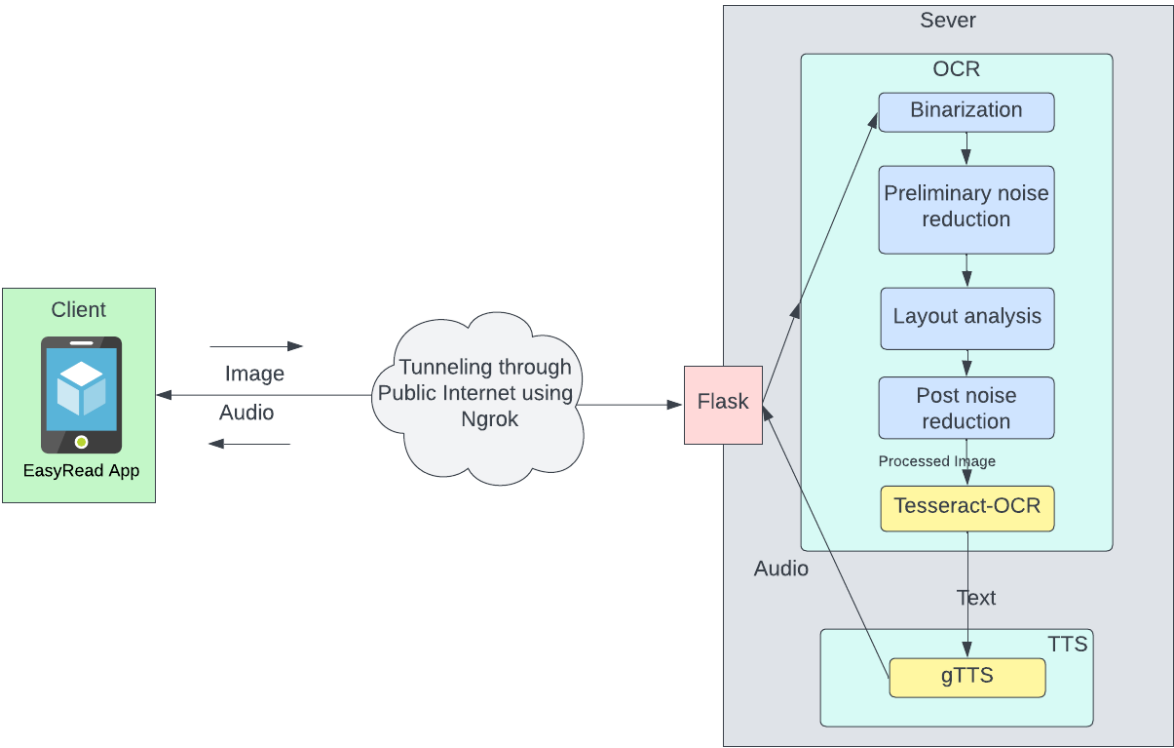


Figure 3.1: System Block diagram

### 3.1.1 User connection

The user is a visually impaired person possibly with loss of vision. They connect to a main processing server through a custom built application for the purpose. The application

enforces easy interface to the user with visual limitations. The application allows the user to capture an image containing a document. The document is then sent as HTTP request to the server application waiting for response in base64 encoded format widely used across the internet. The application will then block for its request to be fulfilled or until its timeout expires. After receiving the reply from the server containing the URL to the fetch its audio from, the application fetches the audio to the local machine and is subsequently played to the user.

### **3.1.2 Request Handling**

Python Flask was selected as the development environment due to its lightweight, user-friendly, and customizable characteristics. Because the default Flask engine is not production-ready, Gunicorn is utilized for the WSGI app. The default Flask engine is unable to process simultaneous multiple user inputs. Due to the developers' decision, Apache was used as the reverse proxy rather than nginx. Several solutions were provided for exporting local hosts to the internet, including Ngrok, Localtunnel, Telbit, Localhost.run, and Cloudflare tunnel. Ngrok was chosen among them since its free tier met the requirements and was therefore integrated into the project.

The server hosts two routes, one for receiving images and the other for sending audio files generated. The server will receive the image from the client in base64 format and process it appropriately using Java-based binarization and segmentation procedures and provide a JSON success reply indicating success receive and process of the sent image, this is done with PUT requests. Another route will return the generated speech output, mp3 files on request with the GET request.

### **3.1.3 Core module**

The flask server application connects to the main module responsible for actual processing of the input. It is composed of a OCR module and a TTS module. Our work focuses on the OCR module so as to make it accurate and robust for images in the wild like that consisting of uneven illuminations, shadows and possibly other defects such as stains. The OCR module further consists of four key sub-modules or steps: binarization, noise removal, layout analysis or reading order determination, and finally the actual conversion of image characters to text. In particular, our work focused on the preprocessing steps of OCR including the binarization, noise removal and layout analysis as openly available OCR; tesseract-OCR already provides large range of language support and pretty accurate text conversions for high quality document images with no further requirement of layout analysis. Thus our work focused on providing high quality binary image generated from documents in the wild to the

tesseract engine in correct order. Thus generated text is then forwarded to gTTS which is a text-to-speech API provided by google for generating high quality audio speech from text input.

## 3.2 Sequence Diagram

The sequence of events between various entities of the system is shown in the figure 3.2. The client initiates the process through the Client App. The request is sent to server from the app and then the server communicates with individual modules to perform their respective tasks. The final response i.e. generated audio is sent to the Client App. The user then interacts with the application to hear the output.

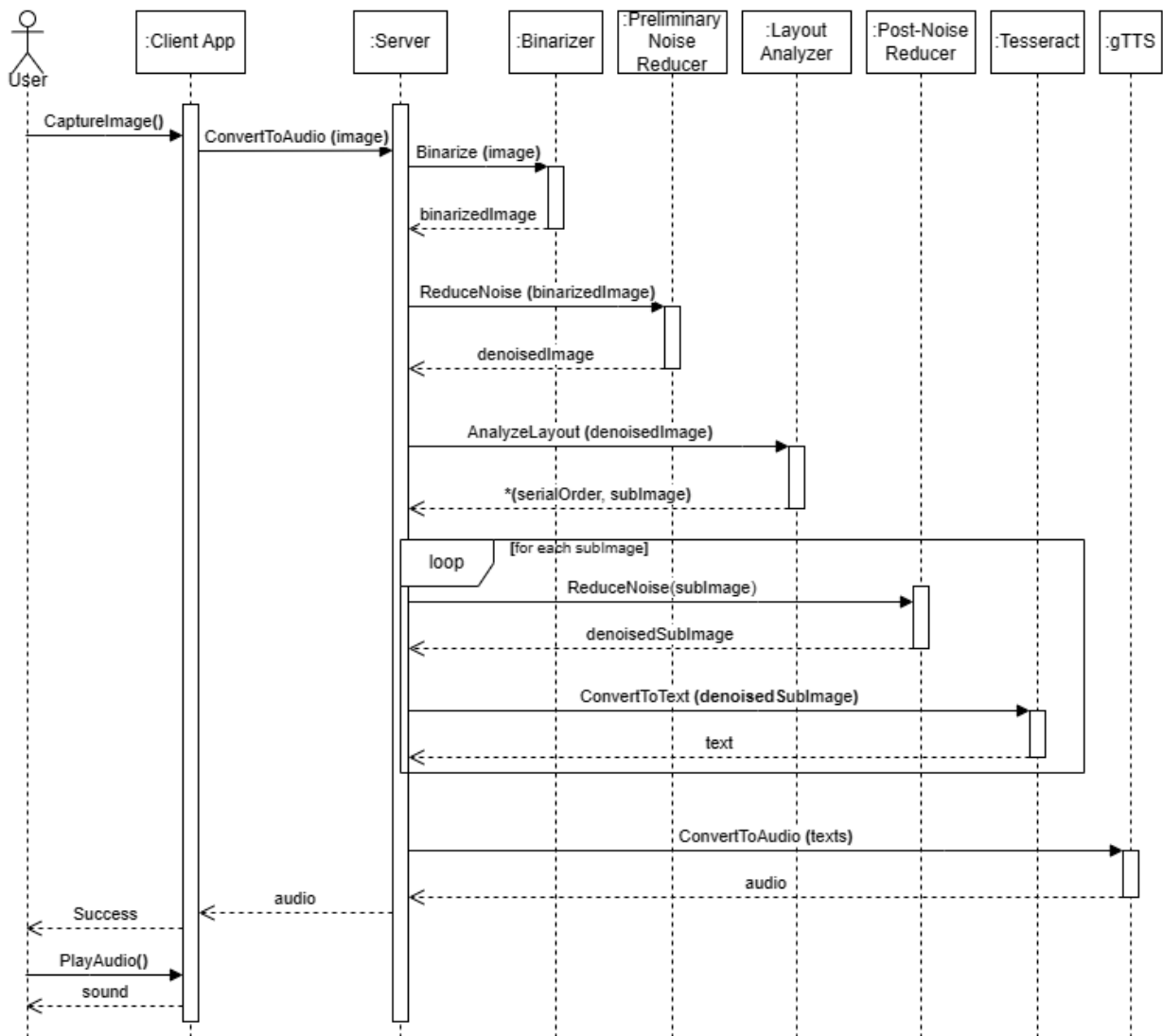


Figure 3.2: System Sequence diagram

### 3.3 Binarization

The binarization procedure implemented in our project is based on some improvements over Sauvola's thresholding algorithm (8). Sauvola's algorithm was improved for our purpose using integral image for fast computation unlike interpolation as proposed in the original paper (8) and similarly use of a regression model to predict proper value of parameter 'k' either for the whole image or for regions in the image was done. As discussed before proper values for the parameters 'k' and 'w' for thresholding is required to be found. However, parameters that are good for some kind of image may not prove to be useful for others. Thus, a simple regression model was used that determines the parameters dynamically for each document image based on some feature extracted from the image that may affect the value parameters to use. Thus selected features need to provide some measure of the document image conditions appropriate for the Sauvola's thresholding method.

```
image ← ImageCapturedFromCamera
bin_image ← BinarizedImage
R ← 128
w ← window_size
k ← predicted_optimal_value_for_k_parameter
i ← 0
while i ≤ image_width − 1 do
  j ← 0
  while j ≤ image_height − 1 do
    m ← MEAN(image, i, j, w)
    sd ← STANDARDDEVIATION(image, i, j, w)
    T ← m * (1 + k * (sd/R − 1))
    if image[i][j] ≤ T then
      bin_image[i][j] ← 1                                ▷ Text pixel
    else
      bin_image[i][j] ← 0                                ▷ Non-Text pixel
    end if
    j ← j + 1
  end while
  i ← i + 1
end while
```



### 3.3.1 Execution time Improvement

Since the local thresholding requires calculating mean and variances for window centered around every pixel, performance can be improved by using integral image which allows calculation of sum and hence mean using constant amount of arithmetic operations. Similarly, for calculating variance we can store the integral image of squares of each pixel value which along with the integral image for original image can be used to calculate the variance within any size of window. Thus used operations are constant i.e. 4 addition/subtraction and a division operations for mean calculation irrespective of the size of the window. Similar holds for variance calculation.

### 3.3.2 Dynamic parameter

As discussed before, a regression model was used for predicting the value of parameter 'k' for a document image provided. For this end, a data-set consisting of image and ground-truth pairs was needed so that evaluation for each image could be done on what is the optimal value of 'k' that gives the most satisfactory result resembling the ground truth the most. To evaluate the resemblance of the binarized image with its ground-truth we used F1 score as accuracy can be misleading (17). The approach taken for building the regression model for dynamic parameter can be divided into two topics feature extraction, data-set preparation, and model building.

#### 3.3.2.1 Feature extraction

It is required to represent the image with some measure(s)/feature(s) for the image condition. For this a range of measures such as mean (m), standard deviation (sd), Otsu's optimal threshold(oT) 2.8, mean of first class from otsu thresholding ( $\mu_0$ ); the classes being separated by the optimal threshold oT, mean of first class from otsu thresholding ( $\mu_1$ ), variance of first class from otsu thresholding ( $\sigma_0$ ), variance of second class from otsu thresholding ( $\sigma_1$ ), entropy(E) and their combinations using multiplication and division such as sd/m, sd/oT ,etc representing the image as a whole were selected as possible candidates for the measures of image conditions. Then based on correlation between the features and optimal-k values possible features were selected as being the most effective ones for predicting optimal-k value.

#### 3.3.2.2 Dataset Preparation and modeling

The goal of preparing data-set was to obtain the (image, optimal-k) pair. The image is represented by its features explained above. The optimal value of 'k' is the one that gives the highest F1 score when compared to the ground truth of binarization for given image.

The process of obtaining dataset can be described as follows:

```

for all (image, ground_truth) in train_set do
  optimal_k ← 0
  k ← 0
  w ← 40
  while k ≤ 0.9 do
    while w ≤ 90 do
      bin_image ← BINARIZE(image, k, w)
      score ← F1SCORE(bin_image, ground_truth)
      if max_score ≤ score then
        optimal_k ← k
      end if
      k ← k + 0.05
      w ← w + 10
    end while
  end while
  feature ← FEATURE(image)
  PRINT (optimal_k, feature)
end for

```

The (optimal-k, feature) pairs were obtained for training sample of images by trying all values with small increments (0.05) within range of 0 to 0.9 and then selecting the optimal as the one that gave the highest F1-score. Though value of window size 'w' is also varied albeit in small range, its effect is ignored.

With these pairs, finally the thus selected features and optimal-k value were fitted with a best fit line; a line was chosen to minimize over-fitting on the training data as very few were available. The (image, ground-truth) pairs data set required for this purpose was made available by DIBCO(2009-2018) (I8); a total of 116 images and their ground-truths. Similarly, some modification to this data set images were done artificially to create more data sets. The final linear model used for predicting the optimal-value of k for given document-image is:

$$k = 0.15 \times \sqrt{\frac{\mu_0}{\sigma_1}} + 0.025 \times E - 0.117 \quad (3.1)$$

Though the given approach focuses on describing the whole image by these features, smaller regions can be selected for better result with trade-off of execution time. The training however can be done with global feature selection as local feature selection requires large

amount of time.

## 3.4 Noise Reduction

The noise present in text images that we will be dealing with is mostly scattered patches of salt and pepper noise throughout the image caused due to uneven shadows in the original image and marginal noise, as a result of background separation from the text document. In order to remove such noise, we apply a score-based filtering technique to filter useful i.e. text-only components from the components obtained after connected component analysis.

### 3.4.1 Segmentation

The segmentation approach we have used is based on connected component analysis using the 4-connectivity of each pixel. Connected component analysis is done to identify each continuous blob of black pixels as an individual component. For the analysis, we label each connected set with a unique label (component index) and then use the component indices of connected sets to extract individual components from the image. The labelling is done by assigning an incremental label to the black pixels belonging to a connected set by iterating through the image and checking for any labelled neighbours in second quadrant of the pixel's neighbourhood. If a labelled neighbour is found, the current pixel is assigned the label of the neighbour, else it is assigned a new label. The labelling process is very similar to the Cluster identification approach suggested by Aldridge, which is based on Hospen-Kopelmann algorithm (19). After one iteration, all the continuous blobs i.e. components are labelled with their unique labels.

After labelling the components, their co-ordinates are used to form a minimum bounding upright rectangle enclosing the labelled components. These bounding rectangles are the output of segmentation process. The result of segmentation completely depends on the clarity of the binarized image as the segmentation process does not consider if the connected component is text or noise. To extract text-only components from the binarized document, the obtained list of components is filtered using component filtering technique to remove noisy components.

### 3.4.2 Component Filtering

A component property based filtering technique can be used to classify individual components obtained by connected component analysis as text or non-text components. The non-text components can then be discarded as unwanted noise. In order to filter the components, each component is filtered based on its properties such as border proximity, count of black pixels, density of black pixels in bounding rectangle, relative size of component with

respect to image and several other metrics. The count of black pixel can help filter extremely small components as salt and pepper noise. A text character image needs to have a certain minimum number of pixels for it to be recognizable as a character. Some of the noise is seen in form of diagonal streaks of black pixels which form a large bounding rectangle, thus having a very low density of black pixels. Marginal noise is often seen as large rectangular patches of dark pixels caused by image background or page borders. the height and width of components is used to decide if it is noise or text.

A combination of these metrics are used with respective score thresholds to classify a component as a text or non-text. As our goal is to filter out the non-text components, we use this filtering technique with initial thresholds to obtain a collection of components having majority of text components by discarding any components in direct contact of the image border regions. A second phase of filtering is done with consideration of other metrics such as component height, width, etc. with their respective thresholds to obtain text components and thus the noise present in the image is reduced. This filtering technique can be useful to filter out a significant amount of components which are salt and pepper noise or marginal noise present in the binarized image.

### 3.5 Reading Order Determination

Tesseract assumes that its input is a binary image with optional polygonal text regions defined. Also the tesseract didn't require its own layout parsing as it used proprietary layout determination technique of Hewlett-Packard (20). For our purpose, it was required to have a layout parsing that was robust and adapted to wide ranges of document layouts. There can be wide range of document layouts ranging from that in newspapers, magazines, articles to books. The main goal of layout analysis for our purpose was simply for determining the proper reading order of texts. For this we don't require the full sophistication of layout determination. For most of the documents, white spaces are used to separate various sections of text and the white spaces are proportional in that large white spaces usually signify more important separations. Taking this observation into account, a novel reading order determination technique was employed.

- First any bordering white-spaces are removed from the document image.
- The white-space that spans along the whole document's height or width is found.
- The image is then separated into two 'sub-images' based on white-space.
- If multiple such white-spaces spanning the whole document's height or width exists then the largest one is selected for bi-parting the image into two sub-images.

- The reading order is as follows: If the image is separated horizontally then read the top sub-image first then bottom sub-image else if the image is separated vertically then read left sub-image first then right sub-image.
- The aforementioned processes are recursively applied until at least columns of text are clearly as remaining text sections can easily be ordered based on horizontal line-spans.

For white-space detection projection profile method (21) was used with some margin in threshold (instead of zero sum over projection) to account for possible noise.

## 3.6 Post Noise Removal

The individual sections obtained after parsing the document's layout are further subjected to noise removal process. Information about the text characters present in the document sections is used to refine the component filtering process. Our assumption is that, each section has a majority of text components and a small number of noise, which could not be removed by initial noise removal process. The size of text fonts may be varying in different sections, therefore a single threshold can not be used to separate noise and text throughout the document. We localize the value of threshold for each section and use it to remove components with properties significantly different from the majority of components in that section. The process of noise removal after layout parsing is as follows:

- Segmentation is applied to individual document sections to get list of components.
- The obtained list is used to obtain bounding box for each component.
- The average width and height of components is calculated for each section.
- Components having dimensions significantly higher than the average dimensions are discarded as noise.

The residual components are used to construct a clean image of the document section and then passed into the recognition model.

## 3.7 Phone Application for Image Capture

Utilizing the Expo React Native framework, a program was created. Installing the React native camera component through expo-camera allowed us to use it because it performs well with text recognition, bar-code scanning, and facial recognition, all of which seem to be relevant for our purposes. Expo-camera was used by us for photography purposes. We used the state features and await the camera-related asynchronous calls. We need permissions in order to use native features, and as our demos depend on the camera, the permissions setup was done accordingly. For the audio permission, no explicit requests were required.

Accordingly, the embedded styling was completed. The mobile application will act as our client and take care of clicking the photo and sending it to a remote server via a retrieve PUT request because we have used a client-server architecture. The audio file, which is playable in the program, will then be provided by the server in mp3 format.

The mobile application will send two requests as per a single-use image input one for sending the image in base64 format and another for downloading the mp3 audio file. The audio file is downloaded and stored in the application cache memory and played.

To obtain an apk, we had to alter the JSON format and obtain the necessary the.apk file for demonstration reasons. Building the apk required the usage of eas-cli, which required registering in the official expo.io. The builder was default set to.abb format, which is the android play store format. We customized it for our purpose.

### **3.8 Framework Selection**

To choose the ideal foundation for creating the system, a great deal of research was done. Given that the system's objective is to be as space-efficient and size-efficient as possible, this was necessary. Given that each programming language has advantages and disadvantages, we have chosen to employ a combination of python and java to optimize the server-side code. To create the application, we used React Native, a javascript-based framework that borrows ideas from the well-known web development framework React. The client-server architecture is the route we have taken. As the client is solely responsible for low-level computing activities, performance is not dependent on client processing power. The atomic nature of the client and server has also reduced compatibility issues.

### **3.9 Server Structure**

Python Flask was selected as the development environment due to its lightweight, user-friendly, and customizable characteristics. Because the default Flask engine is not production-ready, Gunicorn is utilized for the WSGI app. The default Flask engine is unable to process simultaneous multiple user inputs. Due to the developers' decision, Apache was used as the reverse proxy rather than nginx. Several solutions were provided for exporting local hosts to the internet, including Ngrok, Localtunnel, Telbit, Localhost.run, and Cloudflare tunnel. Ngrok was chosen among them since its free tier met the requirements and was therefore integrated into the project.

The server hosts two routes, one for receiving images and the other for sending audio files generated. The server will receive the image from the client in base64 format and process it appropriately using Java-based binarization and segmentation procedures and provide a

JSON success reply indicating success receive and process of the sent image, this is done with PUT requests. Another route will return the generated speech output, mp3 files on request with the GET request.

The noise-reduced image is sent into the open-source OCR library tesseract, which requires highly noise-free inputs as it is designed to deal with digital inputs. Our project use case will mostly contain real-world noisy images so we have employed several preprocessing to decrease the noise as much as possible. We have used a text-to-speech library for the text-to-speech feature, GTTS which works with sending API requests to google TTS.

## 4. Results & Discussion

The results we have achieved from this project are explained below:

### 4.1 Binarization

By use of Integral image a huge performance improvement was achieved as shown in figure 4.1. The execution time also includes the time taken to create integral image. As can be observed the time taken while not using integral image computation increases in squared fashion over window size. This can be confirmed by the fact that number of iterations or pixels increases in squared power when window size (height or width) increases.

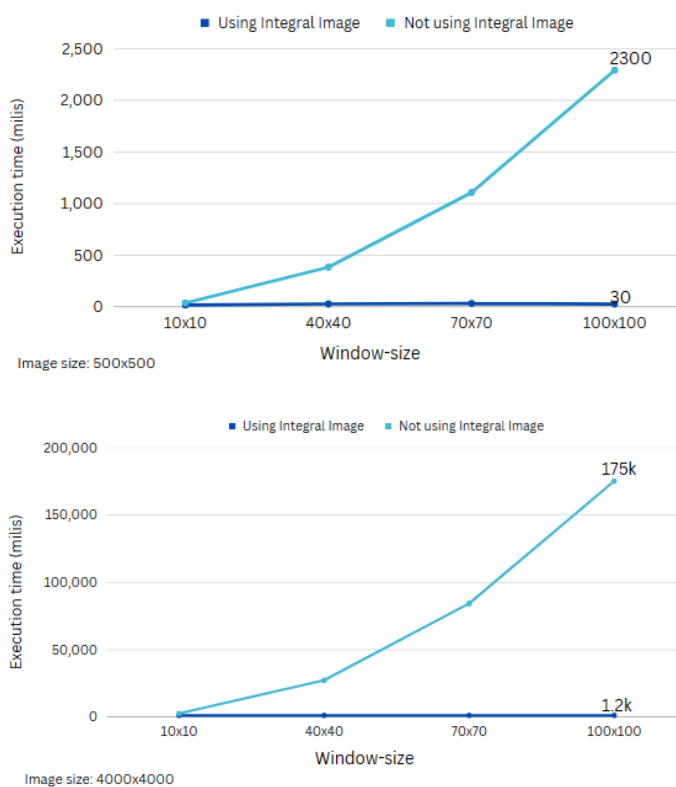


Figure 4.1: Comparison of execution times with and without the use of integral image for binarization

For the dynamic parameter problem, a data set of 116 different images from DIBCO(2009-2018) were used. They were split randomly into nearly equal test and training samples



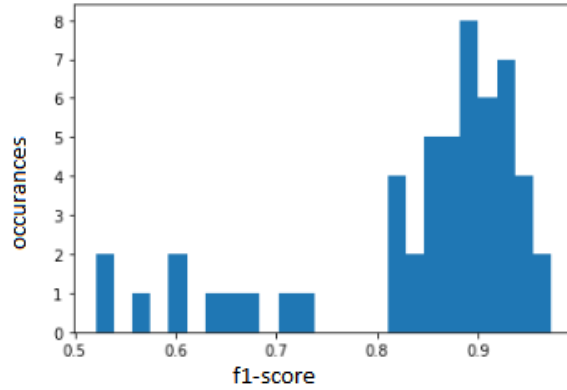


Figure 4.2: A histogram plot for F1 scores obtained by use of predicted 'k' values

of 53 and 63 respectively. An average f1 score of 0.84 compared to 0.72 while using a constant value of 0.34 for the 'k' parameter was observed. Also it was observed as low of performance as 0.02 f1-score while using constant value whereas while using that predicted by the linear model lowest performance of 0.6 f1-score was observed. This suggests that although the simple regression model is far better and stable than simply using a constant value from experimentation for kinds of documents under consideration, the performance can be improved by increasing data set size and model complexity appropriately. Also, the parameter 'k' in Sauvola's thresholding will be better if replaced by some other factors rather than just a constant value. Those factors may be obtained by using measures from global thresholding methods such as Otsu's thresholding highlighted in this paper. The distribution of f1scores is shown by taking a histogram with intervals of 0.04 in figure [4.2](#).

## Noise Reduction

The component filtering technique is able to remove the noise present in the binarized image to a considerable extent. In order to test the accuracy improvement by noise reduction, we calculated the CER ([22](#)) for the output text generated for the binarized image and ground truth image and then for the denoised image and ground truth image. An improvement was seen in the CER after reducing noise by filtering out the non-text components as OCR systems tend to infer noisy blobs as random text characters.

### 1.2 USING VIRTUAL WORK TO SOLVE THE SAME PROBLEM

What are the natural variables in the problem solved in the previous section? One useful choice is  $X$ , the horizontal position of the inclined plane, and  $d$ , the distance of the small block from the top of the inclined plane. Notice that  $d$  is a coordinate defined with respect to a noninertial reference frame, since the point  $d = 0$  is accelerated. The first task is to find expressions in terms of these variables for the kinetic energies of the block and the plane. Throughout this book, we will denote kinetic energies by the letter  $T$ . As

(a) Before Noise Reduction

### 1.2 USING VIRTUAL WORK TO SOLVE THE SAME PROBLEM

What are the natural variables in the problem solved in the previous section? One useful choice is  $X$ , the horizontal position of the inclined plane, and  $d$ , the distance of the small block from the top of the inclined plane. Notice that  $d$  is a coordinate defined with respect to a noninertial reference frame, since the point  $d = 0$  is accelerated. The first task is to find expressions in terms of these variables for the kinetic energies of the block and the plane. Throughout this book, we will denote kinetic energies by the letter  $T$ . As

(b) After Noise Reduction

Figure 4.3: Noise Reduction

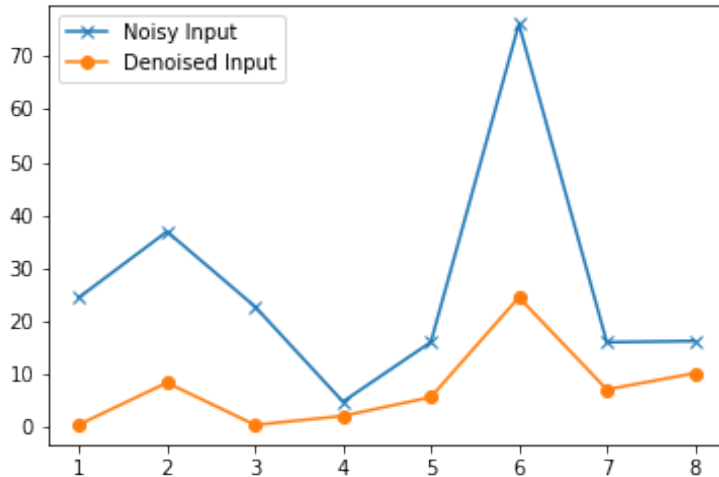
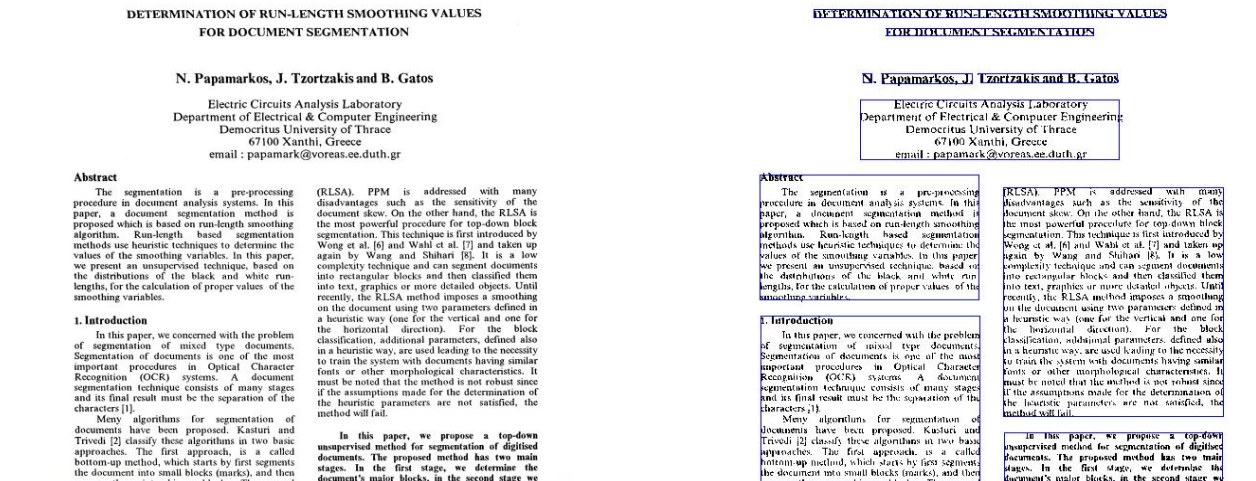


Figure 4.4: Character Error Rates for Noisy and Denoised Inputs

## 4.2 Reading Order Determination

The aforementioned technique for reading order is able to cope with varying layouts such as that of articles and magazines and the sub-images or bounding boxes generated by it for those documents is shown below [4.5](#) and [4.6](#).



(a) Sample image

(b) Segmented result

Figure 4.5: Segmenting an article for reading order, emphSource: Gatos P. (II)



(a) Sample image

(b) Segmented image

Figure 4.6: Segmenting a magazine for reading order, emphSource: LRDE Dataset

### 4.3 OCR Result

A comparison of the tesseract OCR and the proposed system based on CER is presented in the figure [4.7](#). As show by the graph the proposed system performs superior to the tesseract system. This can be attributed to the fact that tesseract OCR on its own employs Otsu’s method of binarization; a global method that cannot cope properly with changing image conditions such as uneven illumination and shadow. The proposed system utilizes Sauvola’s thresholding; a local method and improvement over it using a prediction model for 'k' parameter from some image properties extracted from the Otsu’s method. Noise reduction process also removes large blob like elements enhancing the result further. A sample of 17 images were evaluated for the accuracy comparison. Some sample data set used for the comparison is shown below [4.8](#).

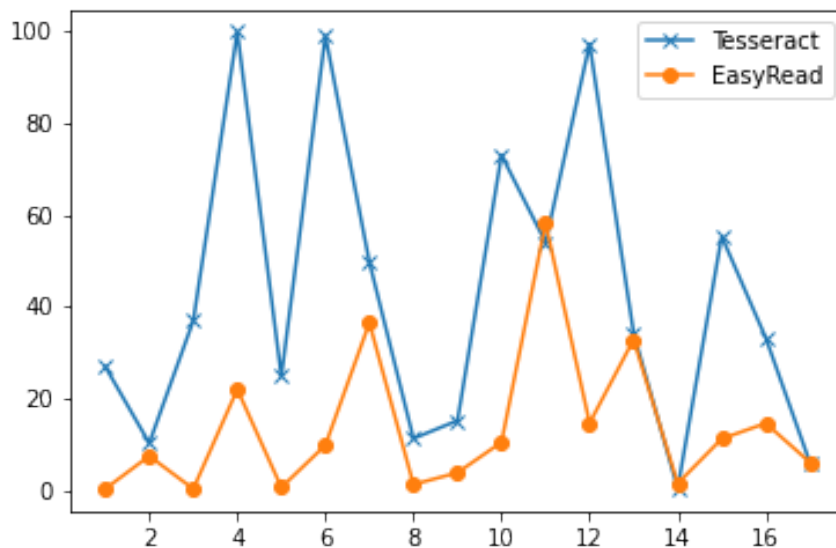


Figure 4.7: Comparison of CER for Tesseract and EasyRead

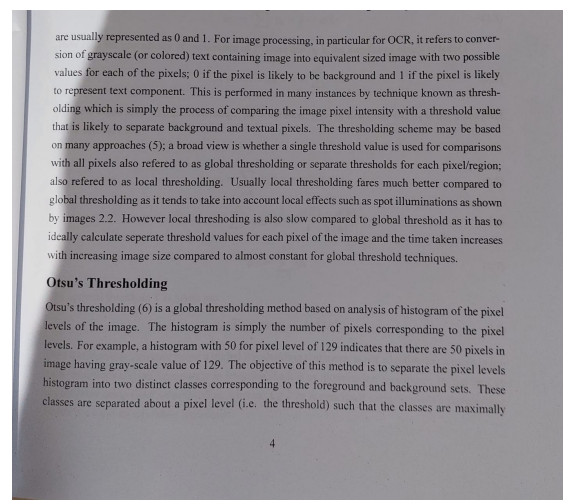
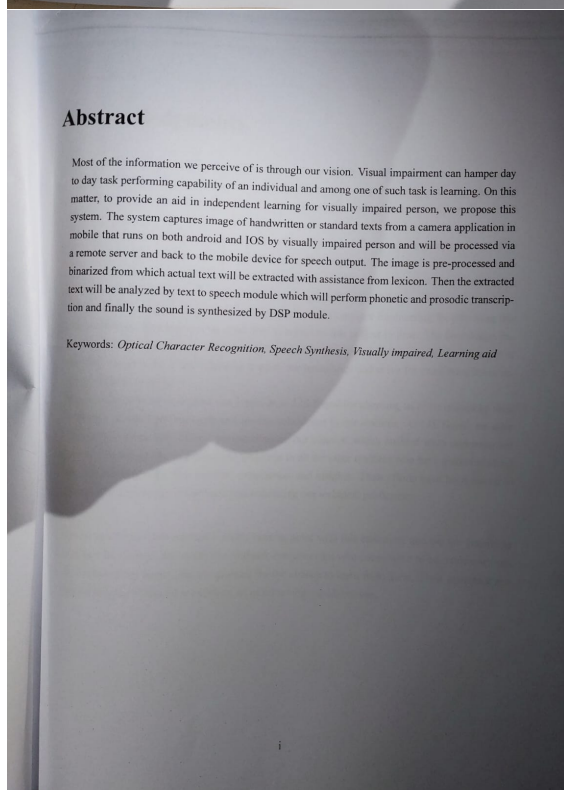
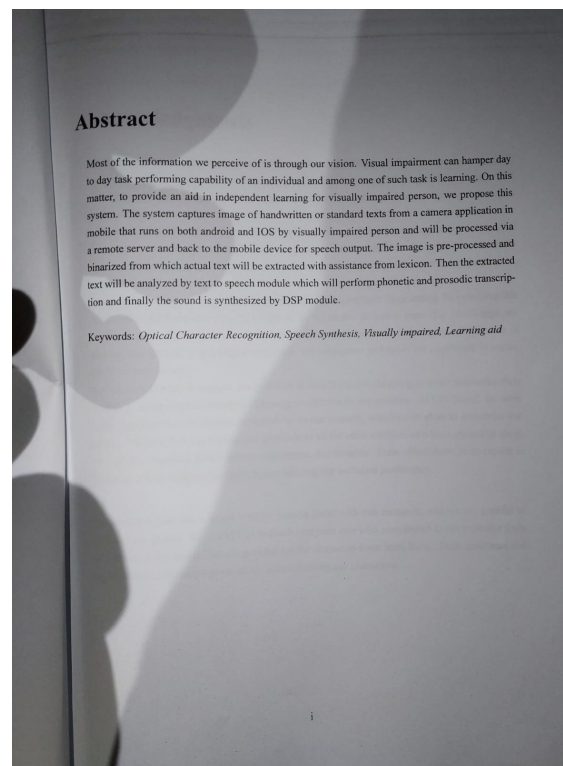
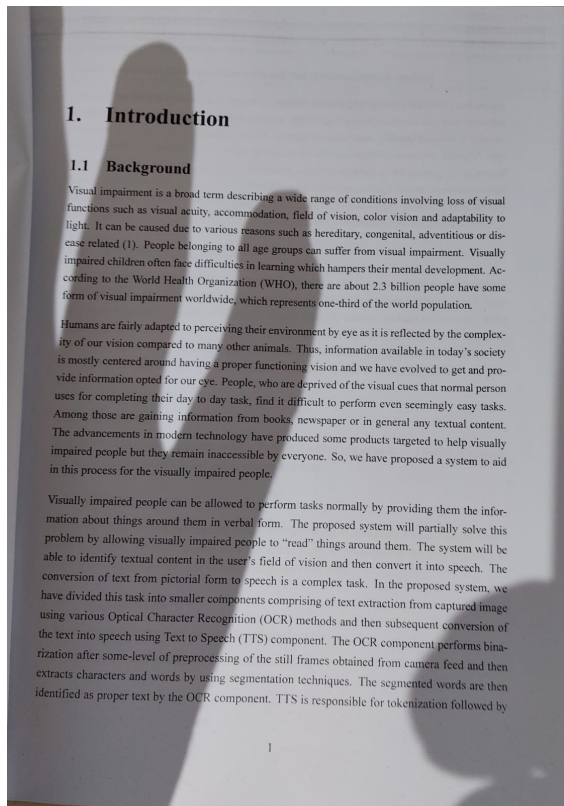


Figure 4.8: Some sample images taken for OCR evaluation

## 5. Future Enhancements

There are many potential enhancements that could be made to our project. Here are a few ideas for possible future enhancements:

- **Construct a Device Representation**

The Raspberry Pi, which runs the program, a camera, and a speaker are needed for the hardware variant.

- **Performance optimization:**

By identifying alternative strategies to lessen the number of calculations necessary to process the data, we can improve the efficiency of our project.

- **Ensure Safety**

To ensure that the handling of the private or sensitive data is safe and risk-free, we can implement better verification measures, enhance data protection, or carry out regular security assessments.

- **Increase the Audience**

If our endeavor is a success, there may be chances to increase the audience by focusing on new populations or geographical areas. This could entail creating marketing campaigns that are aimed at particular demographics, translating material into other languages, or collaborating with other organizations to reach new audiences.

# References

- [1] B. Gatos, I. Pratikakis, and S. J. Perantonis, “Adaptive degraded document image binarization,” *The Journal of the Pattern Recognition Society*, 2005.
- [2] G. Douglas and M. McLinden, “Visual impairment,” *Special Teaching for Special Children?*, p. 26, 2005.
- [3] F. Schantz, H. *The History of OCR, Optical Character Recognition*. Manchester Center, Vt: Recognition Technologies Users Association, 1982.
- [4] E. E. Fournier D’Albe, “On a type reading optophone,” *Proceedings of the Royal Society A*, pp. 373,375, 1914.
- [5] M. Sezgin and B. Sankur, “Survey over image thresholding techniques and quantitative performance evaluation,” *Journal of Electronic Imaging*, vol. 13, pp. 146–168, 01 2004.
- [6] N. Otsu, “A threshold selection method from gray-level histograms,” *IEEE TRANSACTIONS ON SYSTEMS, MAN, AND CYBERNETICS, VOL. SMC-9, NO. 1*, pp. 62–66, 1979.
- [7] W. Niblack, *An Introduction to Digital Image Processing*. Englewood Cliffs: Prentice Hall, 1986.
- [8] J. Sauvola and M. Pietikäinen, “Adaptive document image binarization,” *Pattern Recognition*, vol. 33, no. 2, pp. 225–236, 2000.
- [9] G. Lazzara and T. Géraud, “Efficient multiscale sauvola’s binarization,” *International Journal on Document Analysis and Recognition*, pp. 105–123, 2014.
- [10] F. C. Crow, “Summed-area tables for texture mapping,” 1984.
- [11] C. A. Bouman, “Digital image processing,” 2022.
- [12] V. Ohlsson, “Optical character and symbol recognition using tesseract,” 2016.
- [13] S. Suzuki and K. Abe, “Topological structural analysis of digitized binary images by border following,” *Comput. Vis. Graph. Image Process.*, vol. 30, pp. 32–46, 1985.



- [14] W. Burger and M. J. Burge, “Digital image processing,” p. 181, 2016.
- [15] N. Jamil, T. Sembok, and Z. Bakar, “Noise removal and enhancement of binary images using morphological operations,” *Int Symp Inf Technol 2008 ITSIm*, vol. 3, pp. 1 – 6, 09 2008.
- [16] S. S. Bukhari, F. Shafait, and T. M. Breuel, “Border noise removal of camera-captured document images using page frame detection,” in *Camera-Based Document Analysis and Recognition* (M. Iwamura and F. Shafait, eds.), (Berlin, Heidelberg), pp. 126–137, Springer Berlin Heidelberg, 2012.
- [17] M. Sokolova, N. Japkowicz, and S. Szpakowicz, “Beyond accuracy, f-score and roc: A family of discriminant measures for performance evaluation,” *AI 2006: Advances in Artificial Intelligence, Lecture Notes in Computer Science*, vol. 4304, pp. 1015–1021, 2006.
- [18] I. Pratikakis, K. Zagoris, and I. Santaniello, “Icdar 2019 competition on document image binarization (dibco 2019),” *2019 International Conference on Document Analysis and Recognition (ICDAR)*, 2017.
- [19] M. A., “A finite state machine approach to cluster identification using the hoshenkopelman algorithm,” p. 6, 2008.
- [20] S. R., “An overview of the tesseract ocr engine,” *Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, 2007.
- [21] B. C. M. Javed, P. Nagabhushan, *Extraction of Projection Profile, Run-Histogram and Entropy Features Straight from Run-Length Compressed Text-Documents*. India: Computer Vision and Pattern Recognition, 2008.
- [22] C. Neudecker, K. Baierer, M. Gerber, C. Clausner, A. Antonacopoulos, and s. Pletschacher, “A survey of ocr evaluation tools and metrics,” 2021.



# Appendix

das ao primeiro subconjunto ou ao segundo subconjunto, ou seja, qual a causa da existência desses dois subconjuntos.

Voltando ao exemplo dos casos de uso. Não seria mais útil imaginar que seria possível classificar esses casos de uso em, por exemplo, simples, médios e complexos, para obter conjuntos de medidas com variância menor e, portanto, mais previsíveis? E se, por exemplo, os casos de uso simples pudessem ser programados em menos de um dia, os médios entre um dia e uma semana e os complexos em mais de uma semana? Seria mais fácil fazer previsões sobre o tempo que se levaria para desenvolver um sistema. Bastaria contar a quantidade de casos de uso simples, médios e complexos e multiplicar cada quantidade pela média de tempo associada a cada subconjunto. A informação seria mais relevante, possivelmente, do que uma única média aplicada ao conjunto inteiro dos casos de uso, pois com essa abordagem seriam usadas características dos elementos do conjunto para reduzir a incerteza sobre eles.

Porém, essa subclassificação introduz outro fator de incerteza: não se sabe, *a priori*, se a forma de determinar que um caso de uso é simples, médio ou complexo *realmente* classifica os casos de uso em subconjuntos nos quais os valores de tempo de desenvolvimento tenham variância mais baixa.

Nesse ponto poderia ser aplicada a experimentação científica para validar a hipótese de que uma determinada técnica de classificação dos casos de uso efetivamente classifica a complexidade destes adequadamente, ou seja, que esse sistema de classificação colocará no conjunto “simples” os casos de uso que efetivamente sejam mais rápidos de programar, e no conjunto “complexos” os casos de uso mais difíceis, ficando os demais no conjunto “médios”.

Para testar essa hipótese é necessário comparar dois conjuntos de valores: o valor dado a um caso de uso pelo método de classificação e o valor do tempo que efetivamente se leva para programar o caso de uso. Para efetuar essa comparação é necessário usar o conceito de *covariância*, ou seja, determinar em que grau os dois conjuntos variam conjuntamente.

Figure 5.1: Fig: Sample Input to binarize

mais interessante pode ser o fenômeno. A pesquisa em engenharia de software, por exemplo, tenta descobrir formas de estimar quanto tempo um programa levaria para ser desenvolvido. É uma medida difícil porque mesmo que se tenha uma descrição detalhada de cada função ou caso de uso a ser desenvolvido, o tempo que o desenvolvedor levaria para programar cada um desses elementos poderia variar de alguns minutos a várias semanas.

Saber o tempo esperado (ou o tempo médio) ao criar os programas necessários para realizar um caso de uso<sup>2</sup> pode até ser útil ao fazer previsões para conjuntos de casos de uso, mas médias aritméticas de um conjunto de valores só costumam ser boas estimativas quando uma quantidade significativa de valores está em jogo. Saber o tempo médio que se gasta para programar um caso de uso não permite prever quanto tempo vai-se levar para programar um dado caso de uso, tomado individualmente. Similarmente, mesmo sabendo que em 1.000 jogadas de uma moeda, aproximadamente 500 serão cara e aproximadamente 500 serão coroa, não há meios de saber qual resultado será obtido em uma jogada específica.

No caso de conjuntos que variam muito, como do tempo que se leva para programar casos de uso, será que não se trataria de considerar que não se tem um único conjunto mas sim um certo número de subconjuntos, cada um dos quais com características distintas? Cada um com sua própria média e variância?

Voltando ao conjunto de maior variância do exemplo anterior, {1, 20, 2, 22}. Observando-se esse conjunto, não seria possível concluir que talvez se trate de dois subconjuntos distintos? Ou seja, {1, 2} e {20, 22}. Nesse caso, o que se tem são dois subconjuntos com médias distintas, e cada um dos quais com variância bem menor do que a do conjunto original.

Seria necessário, então, construir uma teoria para determinar

Figure 5.2: Fig: Binarized Output

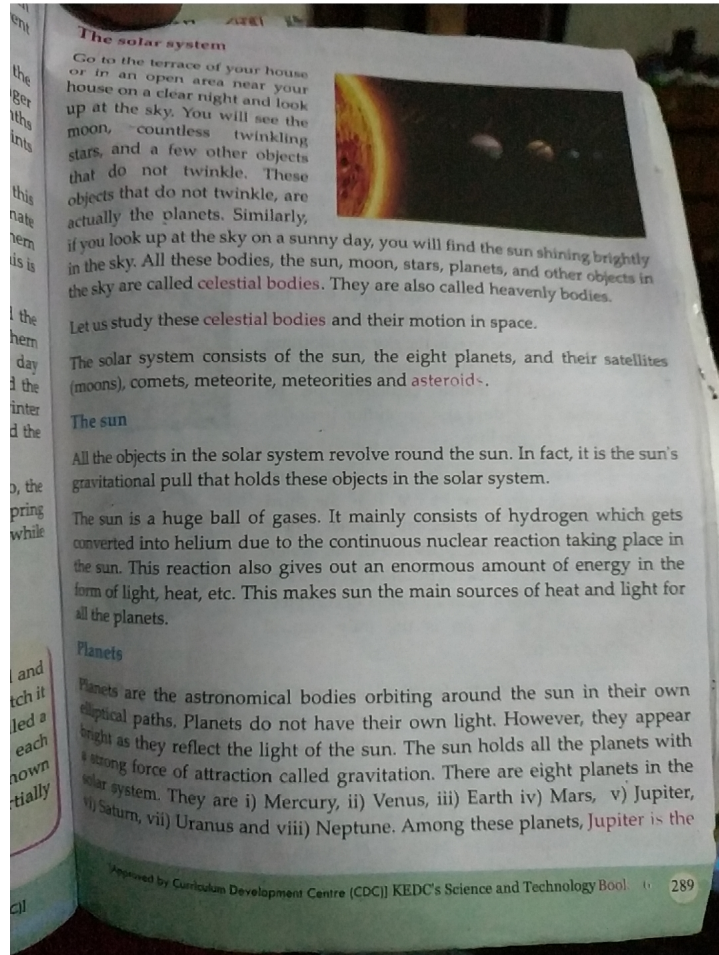


Figure 5.3: Fig: Screenshot of Demonstration camera UI

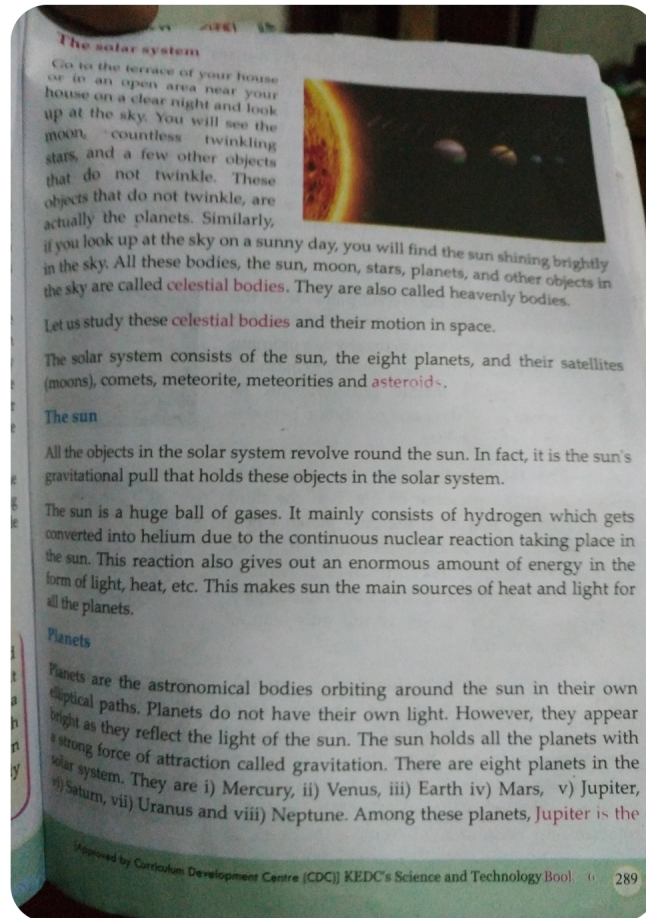


Figure 5.4: Fig: Screenshot of Demonstration clicked picture

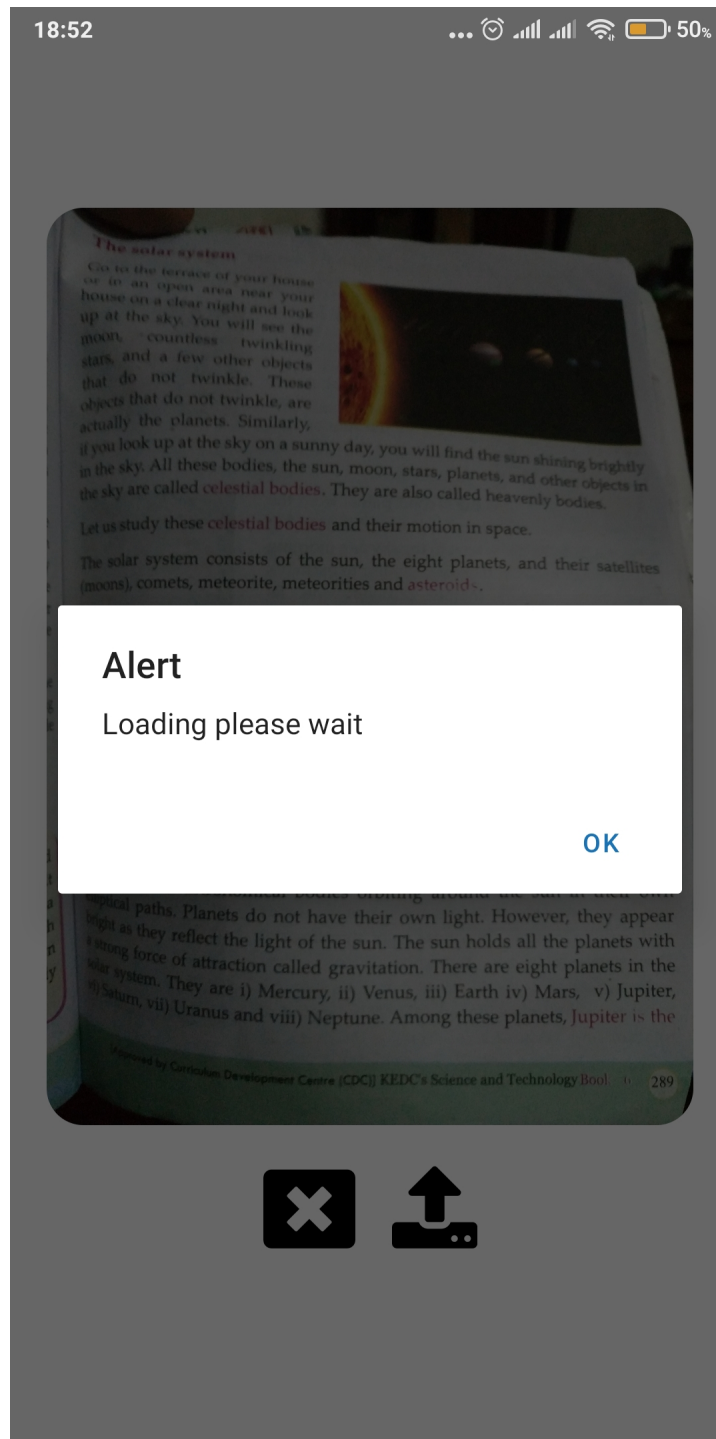


Figure 5.5: Fig: Screenshot of Demonstration waiting message

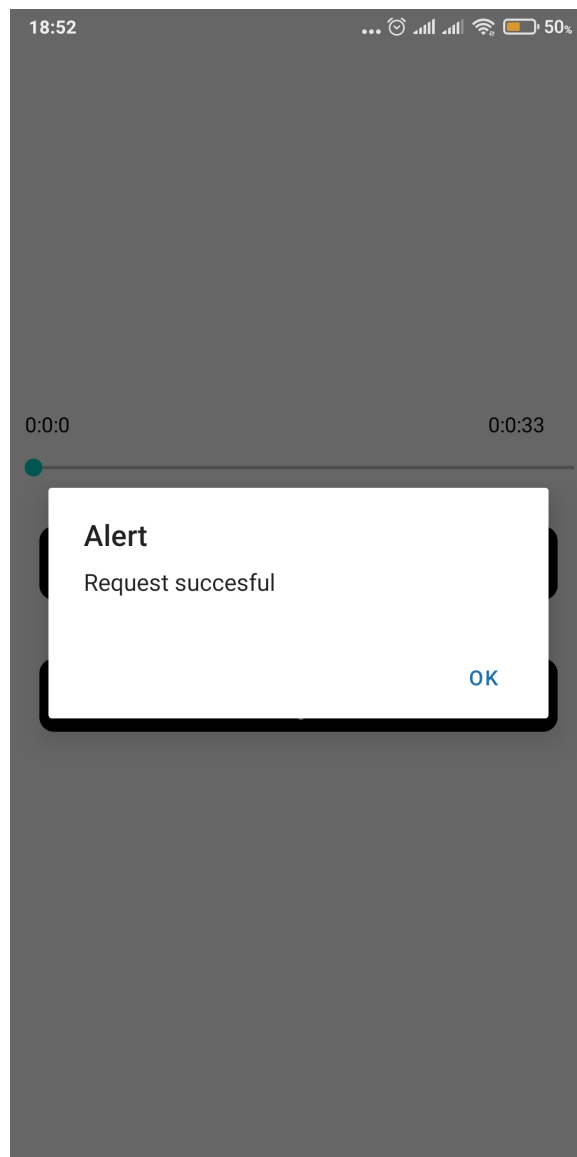


Figure 5.6: Fig: Screenshot of Demonstration success message

18:52

... 🕒 📶 📶 📶 🔋 49%

0:0:0

0:0:33



Figure 5.7: Fig: Screenshot of Demonstration audio play

18:52

... 🕒 📶 📶 📶 🔋 49%

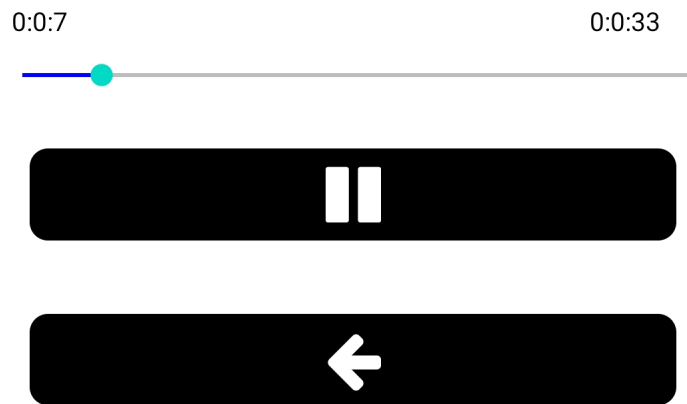


Figure 5.8: Fig: Screenshot of Demonstration audio pause



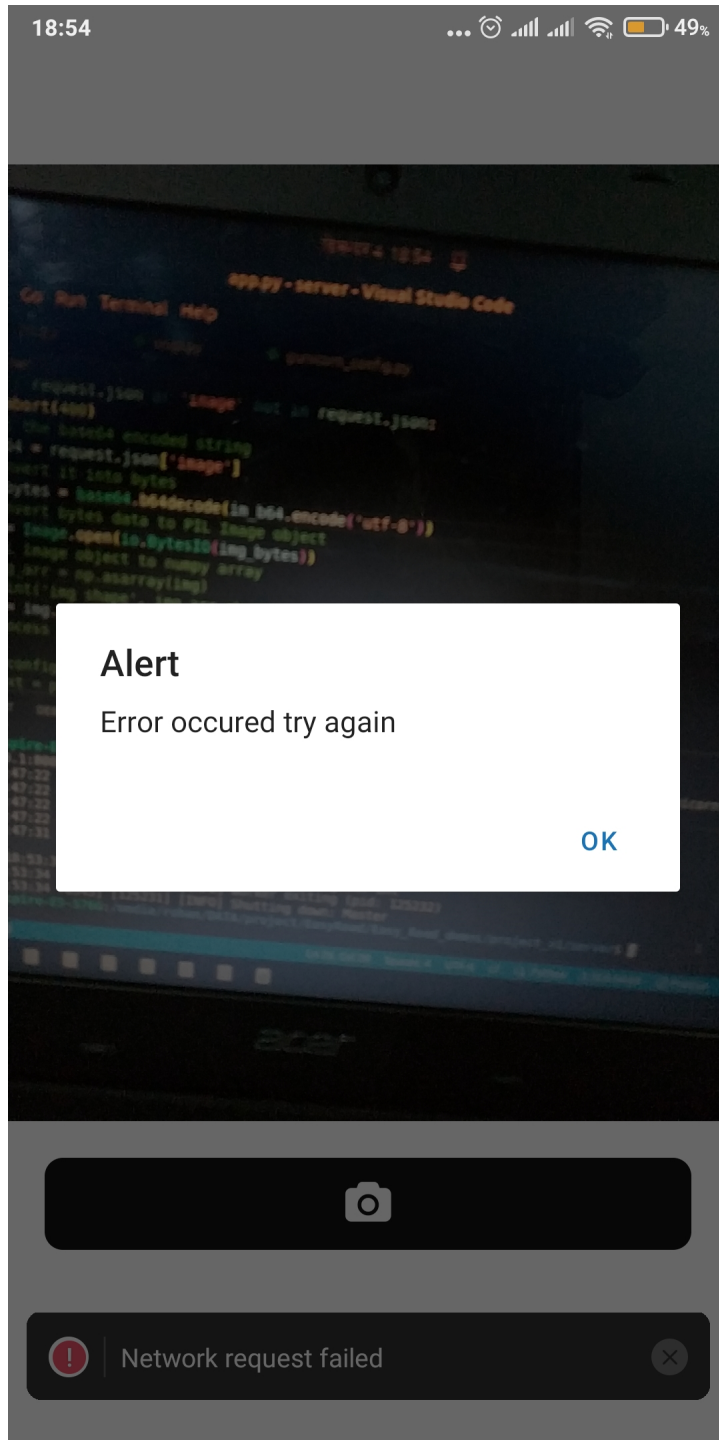


Figure 5.9: Fig: Screenshot of Demonstration error message