# Optical Character Recognition

**BITS ZG628T: Dissertation**

by

Dhvani Shah

(2017HT13466)

**Dissertation work carried out at**

**Tata Consultancy Services, Mumbai**

**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**

**PILANI (RAJASTHAN)**

November 2019

# Optical Character Recognition

**BITS ZG628T: Dissertation**

by

Dhvani Shah

(2017HT13466)

**Dissertation work carried out at**

**Tata Consultancy Services, Mumbai**

Submitted in partial fulfillment of M.Tech. Software Systems degree programme

Under the Supervision of
Paresh Shah,
Project Manager
ACE Software Solutions India Pvt Ltd., Mumbai




**BIRLA INSTITUTE OF TECHNOLOGY & SCIENCE**
**PILANI (RAJASTHAN)**

November 2019

**Birla Institute of Technology & Science, Pilani**

**Work-Integrated Learning Programmes Division**

**First Semester 2019-2020**

**BITS ZG628T: Dissertation**

## ABSTRACT


**BITS ID No.** : **2017HT13466**

**NAME OF THE STUDENT** : **Dhvani Kishor Shah**

**EMAIL ADDRESS** : **dhvanishah73@gmail.com**

**STUDENT'S EMPLOYING ORGANIZATION & LOCATION** : **Tata Consultancy Services, Mumbai**

**SUPERVISOR'S NAME** : **Paresh Shah**

**SUPERVISOR'S EMPLOYING ORGANIZATION & LOCATION** : **ACE Software Solutions India Pvt Ltd., Mumbai**

**SUPERVISOR'S EMAIL ADDRESS**: **paresh.shah11@gmail.com**


**DISSERTATION TITLE** : **Optical Character Recognition**

## ACKNOWLEDGEMENTS

I take this opportunity to owe a great many thanks to people who helped and supported me during the accomplishment of this dissertation.

I express my gratitude to my supervisor Mr. Paresh Shah for his exemplary guidance, monitoring and constant encouragement throughout the course of the work.

I would like to thank my additional examiner Mr. Siddhartha Ghosh for providing me an opportunity to carry out this project, along with purposeful guidance and moral support extended to me throughout the duration of the project work.

I am very grateful to my BITS faculty member Mr. Tirtharaj Dash for providing his valuable feedback during the course of work due to which I got the opportunity to embark on this project.

I would like to thank BITS, Pilani for providing me this great opportunity to explore in new areas.

Last but not the least I would like to thank my parents and friends for moral support during the course of work.

Dhvani Shah

# Table of Contents


**1 Introduction ..... 1**

1.1 Challenges ..... 1

1.2 Solution ..... 1

1.3 Objective ..... 2

**2 Literature Review ..... 3**

2.1 OCR Overview ..... 3

2.2 Segmentation ..... 3

2.2.1 Level of Image Segmentation ..... 4

2.2.2 Image Segmentation approaches ..... 4

2.2.2.1 Discontinuity Based Approach ..... 4

2.2.2.2 Similarity Based ..... 11

2.3 Thresholding ..... 12

2.3.1 Different Types of Thresholding ..... 12

2.4 Intoduction of Morphological operators ..... 18

2.4.1 Dilation and Erosion Operators ..... 18

2.4.2 Opening and Closing Operators ..... 20

2.5 Contours ..... 22

2.5.1 Use of Binary Image in Contour Detection ..... 22

2.5.2 Hierarchy ..... 22

2.5.2.1 Hierarchy Representation in OpenCV ..... 23

2.5.3 Contour Retrieval Mode ..... 24

2.5.4 Contour Approximation Method ..... 28

2.6 Overview of Tesseract ..... 29

2.6.1 Tesseract Architecture ..... 29

**3 Implementation Approach .....31**

3.1 Algorithm Overview.....31

3.2 Flowchart displaying Proposed Algorithm work flow.....35

**4 Application Overview .....36**

**5 User Interface Snapshots.....39**

5.1 User Interface depicting the Login/Register Page.....39

5.2 User Interface depicting Application Home Page.....39

5.3 User Interface depicting process completion Dashboard.....40

5.4 User Interface depicting Reports Dashboard.....41

5.5 User Interface depicting Reports generation.....41

**6 Summary .....42**

6.1.1 Techniques used for this process.....42

6.1.2 OCR engine used: Tesseract.....42

**7 Conclusion and Recommendations.....43**

**8 Directions for future work..... 44**

**9 References ..... 45**

**10 Checklist .....46**

# List of Figures

**Figure 1: Image Segmentation approaches.....4**
**Figure 2: Masking process.....5**
**Figure 3: Point detection mask.....5**
**Figure 4: Example of point detection.....6**
**Figure 5: Line detection mask.....6**
**Figure 6: Example of line detection.....6**
**Figure 7: 1st Derivative output graph.....7**
**Figure 8: Example of 1st Derivative.....8**
**Figure 9: 2nd Derivative output graph.....9**
**Figure 10: Sensitivity of derivative to noise.....11**
**Figure 11: Thresholding example.....12**
**Figure 12: Types of Thresholding.....12**
**Figure 13: Example of importance of local Thresholding over global Thresholding.....15**
**Figure 14: Example of local thresholding.....15**
**Figure 15: Comparison of optimal thresholding and global thresholding with threshold value 128.....17**
**Figure 16: Comparison of optimal thresholding and global thresholding with threshold value 200.....17**
**Figure 17: Comparison of optimal thresholding and adaptive thresholding.....17**
**Figure 18: Original Image to Depict E.g. of Dilation and Erosion.....18**
**Figure 19: Result after dilation on an original image.....20**
**Figure 20: Result after erosion on an original image.....20**
**Figure 21: Original image to depict E.g. of opening and closing operator.....20**
**Figure 22: Result after performing opening operation on an original image.....21**
**Figure 23: Result after performing closing operation on an original image.....21**
**Figure 24: Hierarchy example.....23**
**Figure 25: RETR_LIST example.....24**
**Figure 26: RETR_EXTERNAL example.....25**
**Figure 27: RETR_TREE example.....26**
**Figure 28: RETR_TREE graph.....26**
**Figure 29: RETR_CCOMP example.....27**
**Figure 30: RETR_CCOMP graph.....27**
**Figure 31: Output of Chain Approximation None.....28**

**Figure 32: Output of Chain Approximation Simple.....28**
**Figure 33: Tesseract Architecture.....29**
**Figure 34: Tesseract word recognizer.....30**
**Figure 35: Input Image for OCR.....31**
**Figure 36: Table detection from image.....31**
**Figure 37: Binary threshold inverted image.....32**
**Figure 38: Detect vertical lines.....32**
**Figure 39: Detect horizontal lines.....33**
**Figure 40: Intersection of vertical and horizontal lines.....33**
**Figure 41: Bounding boxes from tables.....34**
**Figure 42: Cell detection.....34**
**Figure 43: Flowchart of proposed algorithm.....35**
**Figure 44: Application Overview.....36**
**Figure 45: Screenshot of proposed Application Login/Register Page.....39**
**Figure 46: Screenshot of proposed Application Home Page and Dashboard.....39**
**Figure 47: Screenshot depicting File Upload operation and new Upload ID generation...40**
**Figure 48: Screenshot depicting the completion status and time of individual processes involved in invoice processing.....40**
**Figure 49: Screenshot depicting the percentages of completion of invoice processes of different vendors and report generation.....41**
**Figure 50: Screenshot depicting process completion and reports download.....41**

# List of Tables

**Table 1: Differentiation between Dilation and Erosion.....19**

**Table 2: Differentiation between Opening and Closing.....21**

# Chapter 1

# Introduction

## 1.1 Challenges

An organization receives thousands of invoices from different vendors. The information contained on those invoices can vary based on the needs of the vendor, different types of transaction, and preferences of the purchasing company. Processing and extracting information from all those invoices, is a tedious process.

Moreover, since there are manual interventions it in many cases leads to erroneous data collection. The idea behind this project is to develop an application that extracts data from the invoices and later do a reconciliation with Purchase Orders thereby replacing the process of manual intervention and possibility of capturing erroneous data. So to overcome the burden of manual intervention we use OCR Engine.

## 1.2 Solution

Optical Character Recognition abbreviated as "OCR" which helps to convert non-readable documents into machine readable form. The hardcopy of documents which are scanned are always in non-readable form to machine, so to extract the data from such document is a task. So we manually save the data of scanned document into our database which is a hectic task. So we use OCR Engine to overcome from the manual burden of entering the data in database. Scanned documents passed through recognition engine of the OCR system which convert the scanned pdf into image and then turn images of handwritten or printed characters into machine readable characters which automate data capture from forms and eliminate keystrokes to reduce data entry costs and helps to maintain high level of accuracy.

## 1.3 Objectives

- The objective- behind the application is the user should be able to upload the invoices and get all the necessary information required for reconciliation.
- Scanning printed documents and extracting attributes from them
- Automating data entry, extraction and invoice processing thereby weeding out manual intervention.
- Reconciliation of invoices with Purchase Order.
- Reading scanned invoices and extract information.
- Extracting handwritten information and store them in databases.
- Archiving invoice information for audit purpose.

# Chapter 2

# Literature Review

## 2.1 OCR Overview

Optical Character Recognition (OCR) is method of extracting text from images and also pdf, which are compiled in such a way that extracting or copying information is difficult. Simple forms of OCR scans each letter pixel by pixel to a known database of fonts and decide on the closest match. Smart OCR systems however break down each character into constituent elements like curves and corners and looking for matching physical features and actual letters. OCR software can also make use of a dictionary so as to output words present in the dictionary. With greater processing power and machine learning techniques that allow software to recognize more subtle patterns over time OCR has become versatile enough to recognize harder to read typefaces, inconsistently printed material and even handwriting.

## 2.2 Segmentation

Segmentation is a process of partitioning an image into its constituents parts or objects in the image and analyzing each of this constituents and each object present in the image for certain information retrieval. Segmentation represents the image into a much more simplified and meaningful version that is easier for analysis. The basic idea of segmentation is to separate different objects depending on some properties of those objects. Segmentation partitions an image into meaningful regions and group together pixels with similar properties. Each of this constituents can be analyzed to extract some information, so that those information can be used for certain computer vision problems. Segmentation finds its usefulness in medical imaging, optical character recognition, machine vision problems, object detection, surveillance and many more.

### 2.2.1 Level of Image Segmentation

The subdivision or level of subdivision or level of segmentation is application dependent. Example we want detect the movement of vehicles on a road, so on a busy road we want to find out what is the movement pattern of different vehicles. The image that is given is an aerial image taken either from a satellite or from a helicopter. First level of segmentation should be to extract the road from aerial images. Once we identify the road then we go for further analysis of these road so that we can identify every individual vehicle on the road. Once we have identified the vehicle then we can go for vehicle motion analysis. So subdivision of an image at the first level should stop after we are able to extract or identify the vehicles. We need not go for segmentation of the vehicle in its consequent parts.

### 2.2.2 Image Segmentation approaches

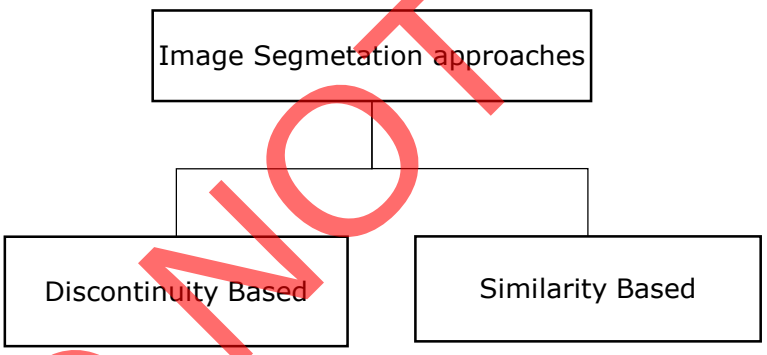



*Figure 1: Image Segmentation approaches*

#### 2.2.2.1 Discontinuity Based Approach

- The partition or subdivision of an image is carried out based on some abrupt changes in intensity levels or some abrupt changes in gray levels of an image.
- Here we identify:
    a) Isolated points in an image
    b) Lines in an image
    c) Edges in an image

- Masking is applied on the image to detect the objects(point,line,edge) from the image.

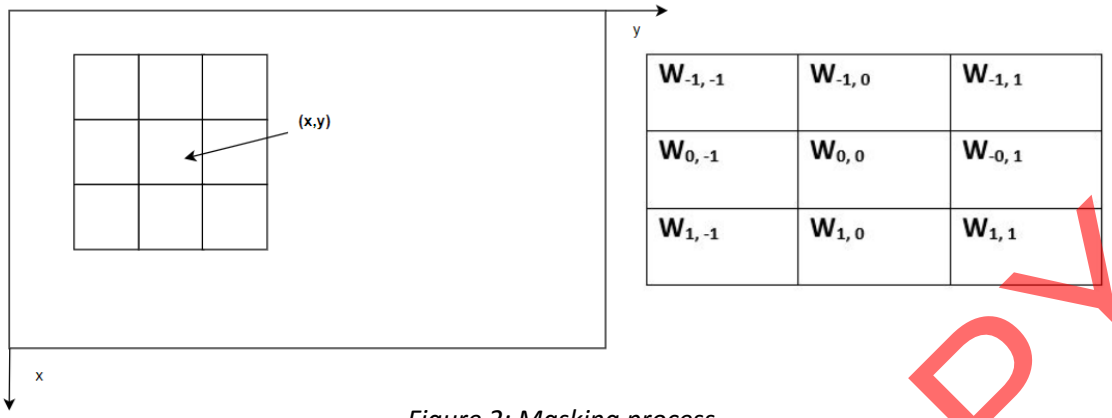



*Figure 2: Masking process*

$$R = \sum_{i=-1}^{1} \sum_{j=-1}^{1} w_{ij}\ f(x+i, y+j)$$

Grayscale image

Mask Coefficient

**A) Point detection:**

| | | |
|---|---|---|
| −1 | −1 | −1 |
| −1 | 8 | −1 |
| −1 | −1 | −1 |

$$|R| \geq T$$

*Figure 3: Point detection mask*

R represent the entire image.

T represent the non-negative threshold value

Absolute value of R is greater than T, where T is non-negative threshold value then we say an isolated point is detected at location(x,y)

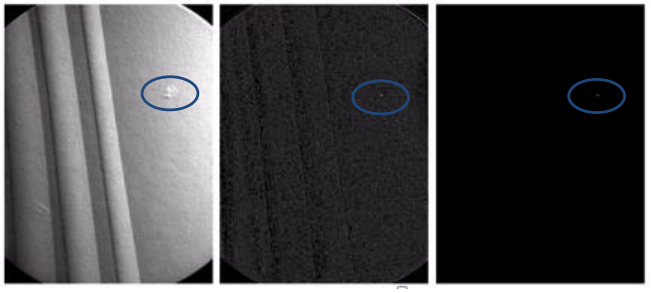


*Figure 4: Example of point detection*

i. Image1 depicts point detection mask.

ii. Image2 depicts X-ray image of a turbine blade with a porosity

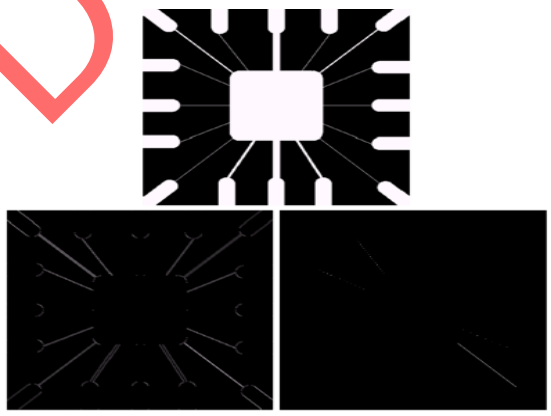
iii. Image3 depicts result of point detection

**B) Line detection:**

- Masks for lines of different directions:


*Figure 5: Line detection mask*

- If we want to detect lines having specific direction (e.g. horizontal direction) then use the mask associated with that particular direction.
- If we want to detect lines having any direction then run all four masks and select the one with highest response.



i. Image1 depicts binary word-bond mask.

ii. Image2 depicts absolute value of result after processing with -45° line detector

iii. Image3 depicts result of the thresholding image (ii)

*Figure 6: Example of line detection*

## C) Edge detection:

- Edge is the boundary between two regions in the image, these two regions have distinct intensity levels.
- When we move from left-to-right we find that we have transition from dark-to-bright-to-dark.
- Works on two approaches:

  a) 1st Derivative

  b) 2nd Derivative

### a) 1st Derivative

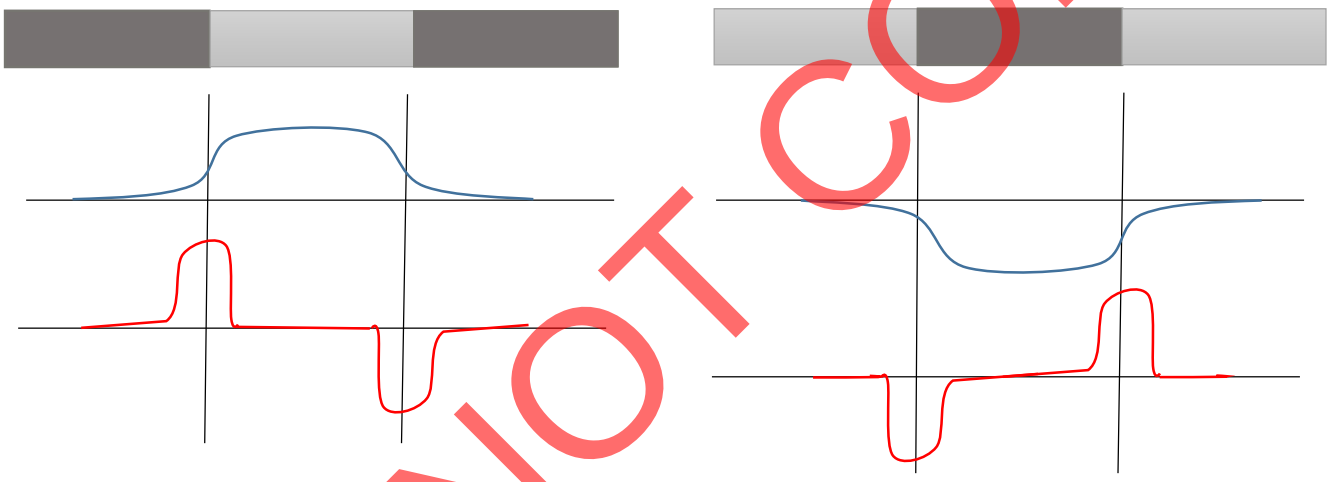


*Figure 7: 1st Derivative output graph*

Red curve depicts the output of 1st Derivative with respect to gray-scale image

I. It responds whenever there is a discontinuity in intensity levels i.e., whenever there is a transition from darker to brighter intensity or vice-versa.

II. For (dark-bright-dark) region, it is positive at the leading edge whereas negative at trailing edge.

III. 1st Derivative it can be computed by using gradient operation

   i. The gradient of an image f(x,y) at location(x,y) is defined as

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} \end{bmatrix}^T$$

$G_x$ represents the partial derivative of *f* along x direction

$G_y$ represents the partial derivative of *f* along y direction

ii. Properties about this gradient vector

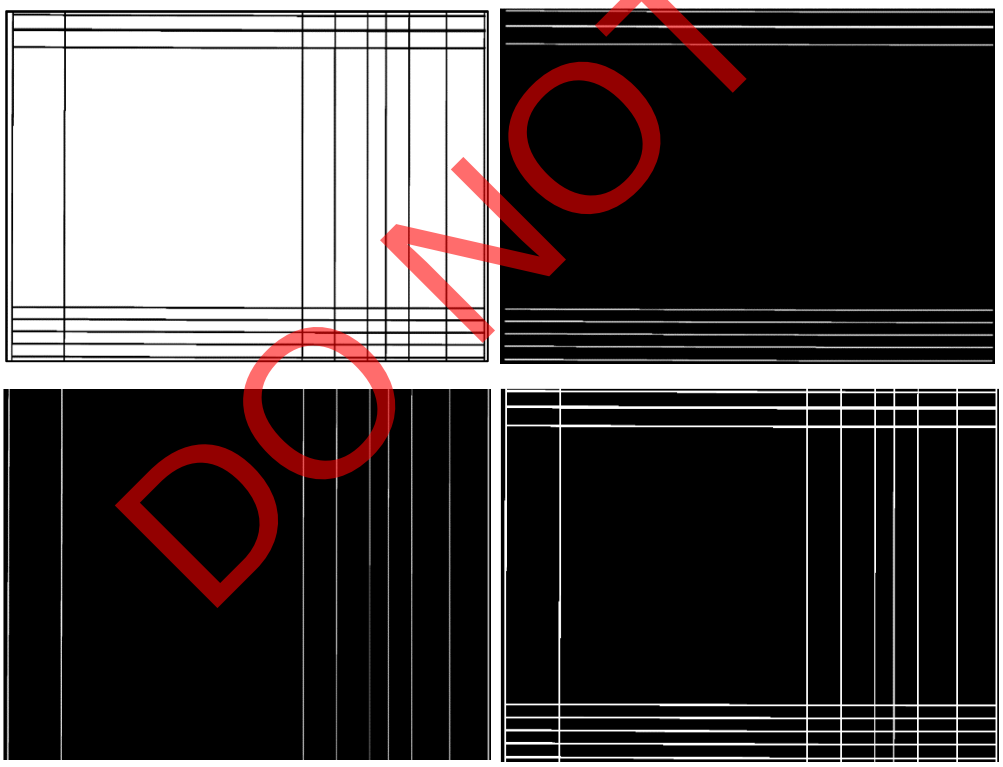
a. It points in the direction of maximum rate of change of image at location(x,y)

b. Magnitude $mag(\nabla f) = \sqrt{G_x^2 + G_y^2} \approx |G_x| + |G_y|$

Magnitude of the gradient tells the strength of the edge at location(x,y)

c. Angle $\psi(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right)$ compute the direction of an edge.

iii. E.g. of 1st Derivative operators are Prewitt and Sobel



a) Original Image

b) $|G_x|$, component of the gradient in the x-direction

c) $|G_y|$, component of the gradient in y-direction

d) Gradient image, $|G_x| + |G_y|$

*Figure 8: Example of 1st Derivative*

**b) 2nd Derivative**

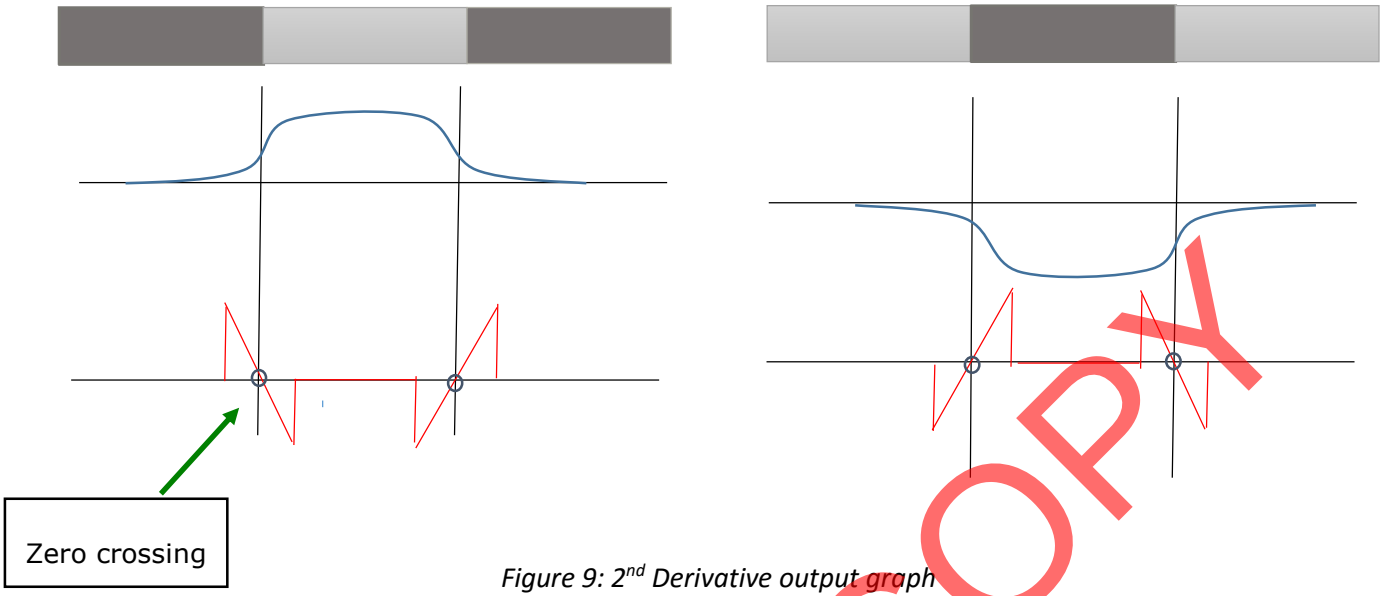



*Figure 9: 2nd Derivative output graph*

Red signal depicts the output of 2nd Derivative with respect to gray-scale image

I. It is positive on the darker side of the edge and it's is negative on the brighter side of the edge.

II. It is basically used for extraction of secondary information that is we can use the sign of 2nd Derivative to determine whether the point lies on darker or brighter side of the edge.

III. Zero crossing of the 2nd derivative is at the midpoint of a gray-level transition, which provides a powerful method for locating the edge.

IV. 2nd Derivative: Laplacian Operator

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$\partial x^2$ represents second derivative along x-axis

$\partial y^2$ represent second derivative along y-axis

Mask given by 2nd Derivative operator is:

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

4-neighborhood

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

8-neighborhood

i. Role of Laplacian Operator:

- Plays secondary role to find out whether a point or a pixel lies on brighter or darker side of an edge.
- Used to accurately locate the location of an edge

ii. Problems with Laplacian Operator:

- Laplacian operator are not usually used for edge detection because it is very sensitive to noise.
- Double edges at every transition.

iii. To reduce effect of noise we will use Gaussian operator

- So the image is first smoother using a Gaussian operator and that smooth image can be operated by the Laplacian operator, this two operator is used together called as Laplacian of Gaussian (LOG) operator.
- $\nabla^2 G$ where $\nabla^2$ is the Laplacian and $G$ is the 2-D Gaussian function
- $G(x,y) = e^{-\frac{x^2+y^2}{2\sigma^2}}$ with standard deviation $\sigma$.
- And we get

$$\nabla^2 G(x,y) = \left[\frac{x^2+y^2-2\sigma^2}{\sigma^4}\right] e^{-\frac{x^2+y^2}{2\sigma^2}}$$

called as Laplacian of Gaussian (LoG).

**c) Sensitivity of edge models:**

Below image depicts sensitivity of edge models corrupted by random Gaussian noise

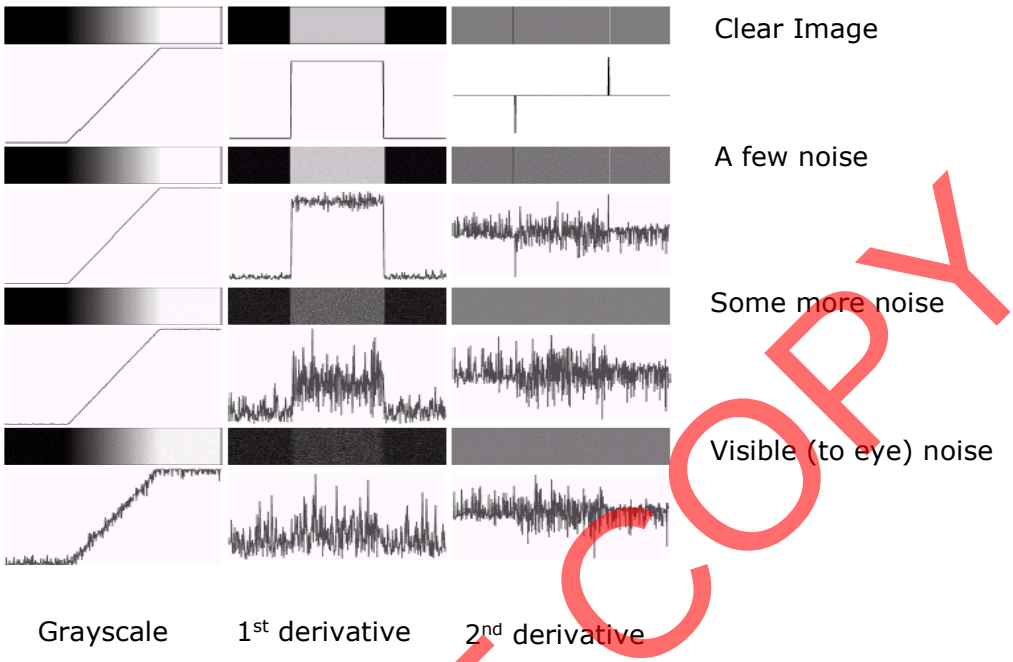



*Figure 10: Sensitivity of derivative to noise*

**2.2.2.2 Similarity Based Approach**

- Group those pixels in an image which are similar in some areas
- Approaches:
  a) **Thresholding:** Partitions the complete image by a threshold value thereafter segmentation is achieved by scanning each pixel and labelling it as background or foreground depending on the gray level of that pixel.
  b) **Region Growing:** Region grows based on connectivity or adjacency and similarity. In this we start from any particular pixel in an image then we group all other pixel which are connected to this particular pixel based on similar intensity value.
  c) **Region Splitting and Merging:** In this we first split the image into number of different components following some criteria. Once we split the image into smaller sub-images then we try to merge that sub-images which are adjacent and are similar in some sense.

## 2.3 Thresholding

Segmentation is dividing the image into different regions such that the boundaries are marked. There are various complex algorithm for segmentation. The very basic form of segmentation is thresholding. Thresholding partitions the complete image by a threshold thereafter segmentation is achieved by scanning each pixel and labelling it as background or foreground depending on the gray level of that pixel. The success of the algorithm depends entirely on how the image can be partitioned.


*Figure 11: Thresholding example*

### 2.3.1 Different types of thresholding

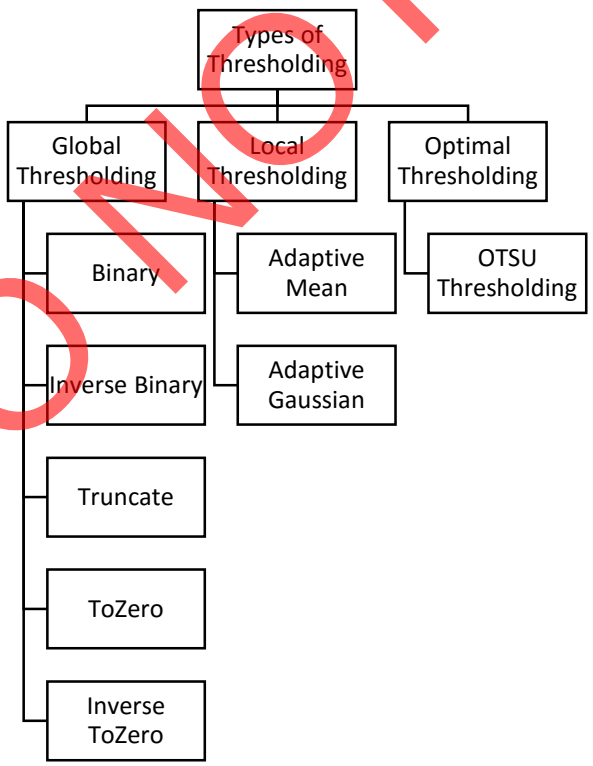



*Figure 12: Types of Thresholding*

## A) Global Thresholding

Global thresholding is the simplest form of thresholding where the image segmentation is done using one single threshold value T for an entire image.

$$\begin{bmatrix} 3 & 0 & 1 & 5 \\ 7 & 6 & 0 & 4 \\ 2 & 7 & 0 & 6 \\ 1 & 3 & 5 & 5 \end{bmatrix}$$

Here we have use 4x4 image so we have 16 pixels

Minimum intensity value is 0

Maximum intensity value is 7

### 1) Binary Thresholding

$$dst(x,y) = \begin{cases} max\ Val, & if\ src(x,y) > thresh \\ zero, & otherwise \end{cases}$$

Let the threshold value be 4 (T=4)

If intensity value of a pixel is greater than the threshold value then in the output intensity of that pixel will have the max value, else it will have zero value.

$$\begin{bmatrix} 3 & 0 & 1 & 5 \\ 7 & 6 & 0 & 4 \\ 2 & 7 & 0 & 6 \\ 1 & 3 & 5 & 5 \end{bmatrix} \xrightarrow{\text{Binary T=4}} \begin{bmatrix} 0 & 0 & 0 & 7 \\ 7 & 7 & 0 & 0 \\ 0 & 7 & 0 & 7 \\ 0 & 0 & 7 & 7 \end{bmatrix}$$

Input Output

### 2) Inverse Binary Thresholding

$$dst(x,y) = \begin{cases} zero, & if\ src(x,y) > thresh \\ max\ Val, & otherwise \end{cases}$$

Let the threshold value be 4 (T=4)

If intensity value of a pixel is less than the threshold value then in the output intensity of that pixel will have the max value, else it will have zero value.

$$\begin{bmatrix} 3 & 0 & 1 & 5 \\ 7 & 6 & 0 & 4 \\ 2 & 7 & 0 & 6 \\ 1 & 3 & 5 & 5 \end{bmatrix} \xrightarrow{\text{Inverse Binary T=4}} \begin{bmatrix} 7 & 7 & 7 & 0 \\ 0 & 0 & 7 & 7 \\ 7 & 0 & 7 & 0 \\ 7 & 7 & 0 & 0 \end{bmatrix}$$

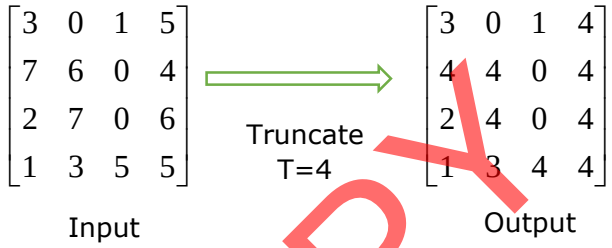
Input Output

### 3) Truncate Thresholding

$$dst(x,y) = \begin{cases} threshold, & if\ src(x,y) > thresh \\ src(x,y), & otherwise \end{cases}$$

Let the threshold value be 4 (T=4)

If intensity value of a pixel is greater than the threshold value then we limit the intensity value of that pixel to the threshold value, else we will keep intensity value of a pixel as it is.



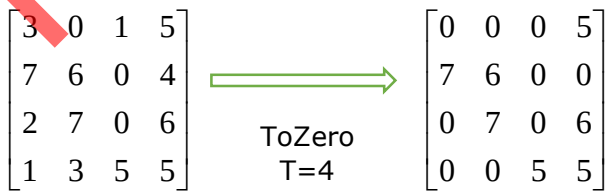

Input Output

### 4) ToZero Thresholding

$$dst(x,y) = \begin{cases} src(x,y), & if\ src(x,y) > thresh \\ zero, & otherwise \end{cases}$$

Let the threshold value be 4 (T=4)

If intensity value of a pixel is greater than the threshold value then we will keep intensity value of a pixel as it is, else it will have zero value.



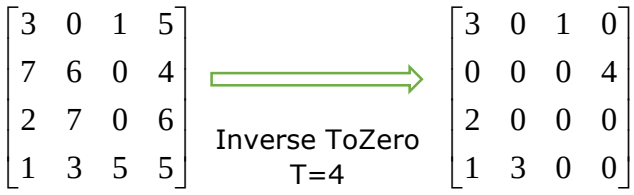

Input Output

### 5) Inverse ToZero Thresholding

$$dst(x,y) = \begin{cases} zero, & if\ src(x,y) > thresh \\ src(x,y), & otherwise \end{cases}$$

Let the threshold value be 4 (T=4)

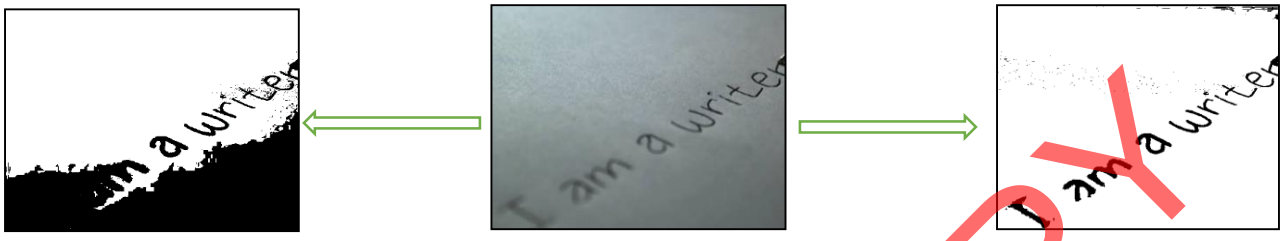
If intensity value of a pixel is greater than the threshold value then make it as zero, else we will keep intensity value of a pixel as it is.




Input Output

## B) Local / Adaptive Thresholding

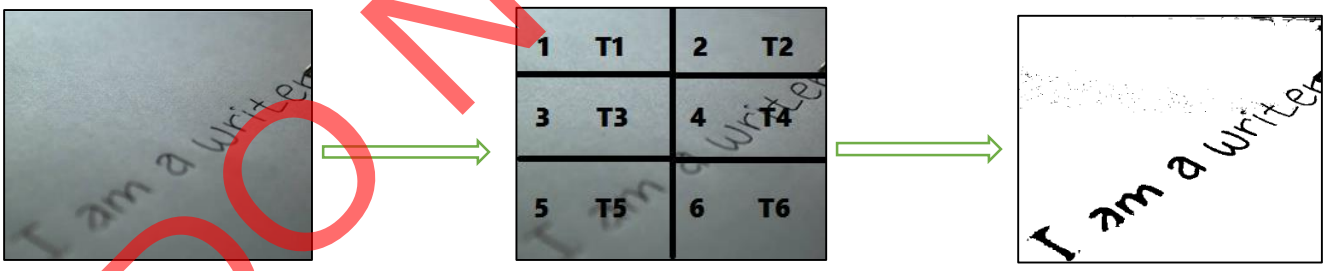
### Importance of Local Thresholding over Global Thresholding



Global Threshold Image | Original Image | Local Threshold Image

*Figure 13: Example of importance of local thresholding over global thresholding*

- In case of images where the intensity level is not consistent throughout the image global thresholding does not perform well and information gets lost due to uneven distribution of intensity. So to overcome the problem of information loss we use local thresholding.
- In global thresholding we apply one threshold to entire image but in local thresholding we split the image into sub-region and apply thresholding to that sub-region part.




Input Image | Intermediate Step of Adaptive Thresholding | Output Image

*Figure 14: Example of local thresholding*

Here input image is divided into six sub-region and apply global thresholding to the sub-region and then combine all them to get the result.

Syntax:

adaptiveThreshold(InputArray src, OutputArray dst, double maxValue, int adaptiveMethod, int thresholdType, int blocksize, double C)

InputArray src - Input Image

OutputArray dst – Output Image

maxValue – The value to limit the intensity value of the pixel (E.g. Suppose for a sub-region threshold value is 100 and pixel have 110 intensity value so in the output image the intensity value of the pixel will be 100)

adaptiveMethod – Specify which kind of adaptive algorithm we are using

OpenCv supports two adaptive algorithm

a) Adaptive Mean: Calculates the threshold value by calculating the mean of the sub-region.
b) Adaptive Gaussian: Calculates the threshold value by calculating the weighted mean of the sub-region.

thresholdType - Type of thresholding to be used (thresh_binary or thresh_binary_inv)

blocksize – size of the sub-region(it is always odd like 3x3 or 7x7) the region for which we calculate the threshold value

C – value subtracted from calculated mean

[**Note:** To get the threshold value we subtract C from the calculated mean (Adaptive Mean) or calculated weighted mean (Gaussian Mean)]



### C) Optimal Thresholding / OTSU Thresholding

Syntax :

threshold(gray_img, op_img, min_val, max_val, CV_THRESH_BINARY | CV_THRESH_OTSU)

- In case of global thresholding min val is threshold value.
- But in case of OTSU Algorithm calculates the min value(threshold value) on it's own so there is no need to specify the min value.

- Case 1: When threshold value is 128

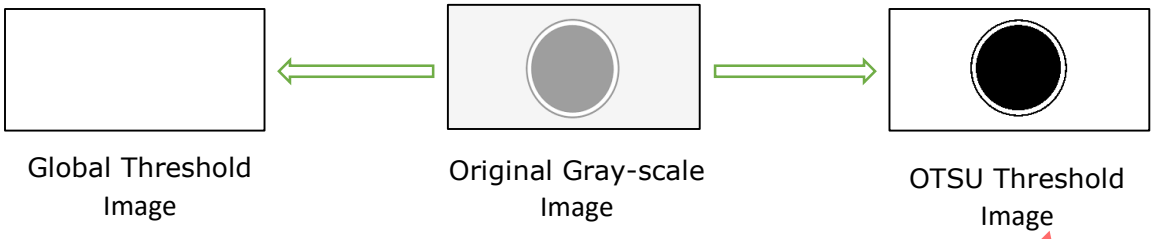



*Figure 15: Comparison of optimal thresholding and global thresholding with threshold value 128*

- In the above image we can see that when we apply global thresholding on an original image it gives no output because 128 act as thresholding value and when we apply OTSU thresholding it calculate it's own thresholding value (won't consider 128 as thresholding value) and gives the output.
- Case 2: When Threshold value is 200

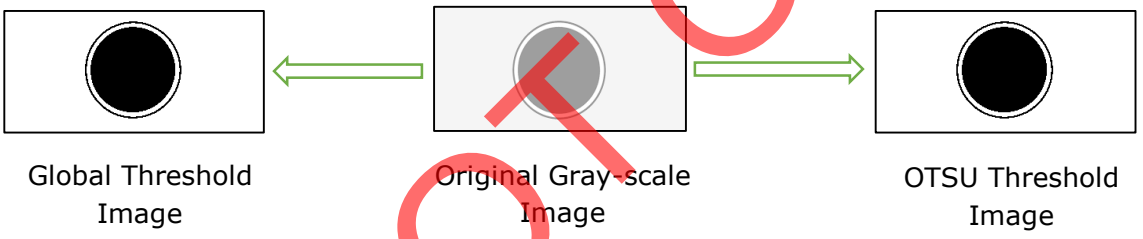



*Figure 16: Comparison of optimal thresholding and global thresholding with threshold value 200*

- In the above image we can see that even if there is change in thresholding value the OTSU thresholding gives the same output as that of previous.

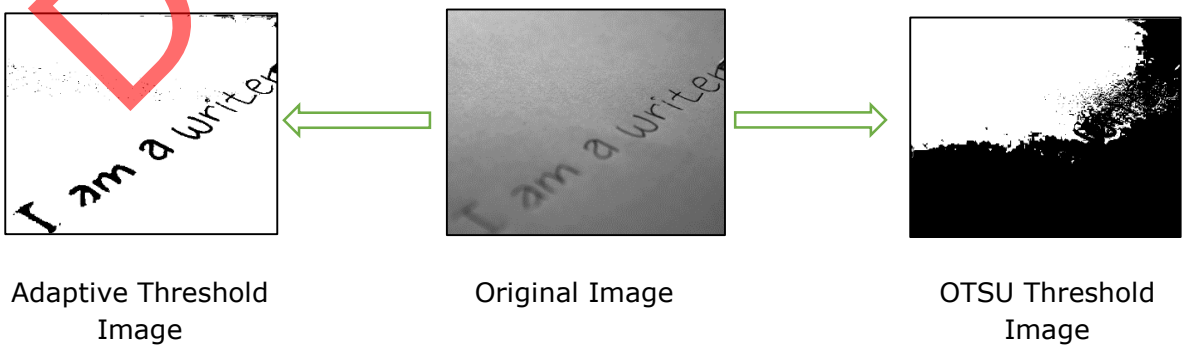



*Figure 17: Comparison of optimal thresholding and adaptive thresholding*

- **Advantage**: Whenever the images are bimodal i.e., we can separate the image into foreground and background OTSU thresholding comes very handy. OTSU method tries to find the threshold such that the in class variability is minimum so that the classes are as compact as possible.
- **Disadvantage**: OTSU thresholding fails when the image is not bimodal. In the Figure 17 we can see that there is loss of information from the image. So to overcome this situation we use adaptive thresholding method.

## 2.4 Introduction of Morphological Operators

In **morphology** we find the shape and size or the structure of the object.

**Structural element** is the mask or window which we place on original image to find the desired output.

There are two main characteristics of structural elements:

**Shape:** Circular, square, rectangle, triangle.

**Size**: varies from 3x3 to 21x21.

**Morphological Operation consist:**

1) Dilation

2) Erosion

3) Opening

4) Closing

### 2.4.1 Dilation and Erosion Operators

A is 4x4 image

A: [1 1 1 1

1 0 0 1

1 0 0 1

1 1 1 1]

B is structuring element

B:[ (1) 1

1 1 ]

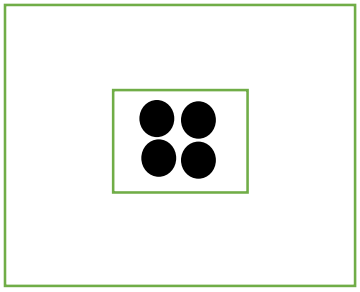


*Figure 18: Original Image to Depict E.g. of Dilation and Erosion*

Original Image with 4 0's in between

| Sr. No. | Dilation | Erosion |
|---|---|---|
| 1 | Expand the image | Shrink the image |
| 2 | Increase white pixel of image and simultaneously decrease black pixel of image | Erode the white pixel so it decrease white pixel of image and simultaneously increase black pixel of image |
| 3 | When we place the Structuring element or mask (B) on the image (A) we get three condition:<br>case 1: all the elements in the B will match perfect (Perfect match then 1)<br>case 2: some element should match (**Some match then 1**)<br>case 3: Nothing is matching (No match then 0) | When we place the Structuring element or mask (B) on the image (A) we get three condition:<br>case 1: all the elements in the B will match perfect (Perfect match then 1)<br>case 2: some element should match (**Some match then 0**)<br>case 3: Nothing is matching (No match then 0) |
| 4 | A=[1 1 1 1<br>1 0 1 1<br>1 1 1 1<br>1 1 1 1]<br>result after dilation | A=[0 0 0 1<br>0 0 0 1<br>0 0 0 1<br>1 1 1 1]<br>result after erosion |

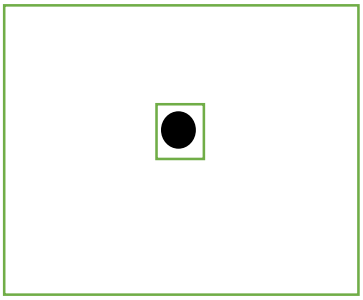
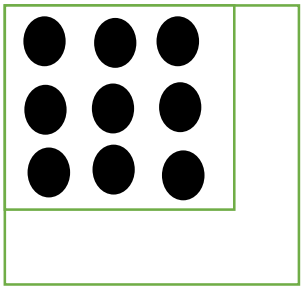
| Sr. No. | Dilation | Erosion |
|---|---|---|
| 5 |  *Figure 19: Result after dilation on an original* |  *Figure 20: Result after erosion on an original image* |
| 6 | Holes reduces:<br>In Original Image there was 4 0's<br>which reduces to 1 0's | Holes Increases:<br>In Original Image there was 4 0's which<br>increases to 9 0's |

*Table 1: Differentiation between Dilation and Erosion*

### 2.4.2 Opening and Closing Operators

A is an image

A= [ 0 0 0 0 0 0 0 0
1 1 1 0 0 1 1 1
1 1 1 0 0 1 1 1
1 1 1 1 1 1 1 1
1 1 1 0 0 1 1 1
1 1 1 0 0 1 1 1]

B is a mask

B= [ (1)
1 ]

Original image were 1 and 2 objects are connected with small gap in between

(All black portion are 0's and white portion are 1's)

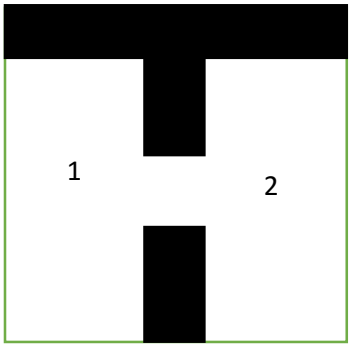
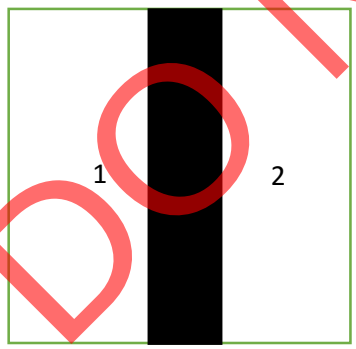
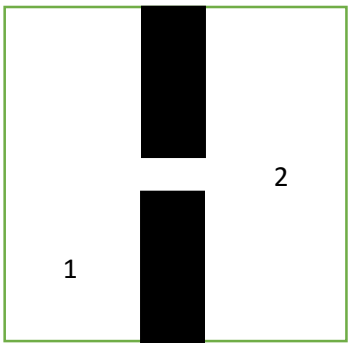



*Figure 21: Original image to depict E.g. of opening and closing operator*

| Sr. No. | Opening | Closing |
|---|---|---|
| 1 | First erosion takes place and then dilation occur. | First dilation takes place and then erosion occur. |
| 2 | After erosion of Matrix A with B as mask<br>C= [ 0 0 0 0 0 0 0 0<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1 ]<br>After dilation of Matrix C with B as mask<br>[ 1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1 ] = > Result | After dilation of Matrix A with B as mask<br>C= [ 1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 1 1 1 1 1<br>1 1 1 1 1 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1 ]<br>After erosion of Matrix C with B as mask<br>[ 1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 1 1 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1<br>1 1 1 0 0 1 1 1 ] = >Result |
| 3 |  <br>*Figure 22: Result after performing opening operation on an original image*<br>All black portion are 0's and white portion are 1's |  <br>*Figure 23: Result after performing closing operation on an original image*<br>All black portion are 0's and white portion are 1's |

| Sr. No. | Opening | Closing |
|---|---|---|
| 4 | Here object are disconnected since there are no gaps, here objects 1 and 2 are separated since there is straight line of 0's in between them.<br>So Opening break the connectivity. | Closing helps to improve the connectivity between two objects 1 and 2 with a thin gap in between. |

*Table 2: Differentiation between Opening and Closing*

## 2.5 Contours

- Contours is curve that joins all the continuous points who have same color or intensity
- Contours can be widely used for shape analysis, object detection and recognition

### 2.5.1 Use of Binary Image in Contour Detection

- OpenCv find contours with the idea of finding white object from black background. So object to be find is white and background should be black.
- Binary image is obtained by converting image to grayscale. To find contours we use binary image of the actual image.
- We apply threshold or canny edge detection technique on grayscale image to get good accuracy and precision.

### 2.5.2 Hierarchy

Hierarchy is representation of the relationship between the objects. It is useful for determining parent-child relationship between contours.

In this example below image depicts nesting of object where each object contains objects of different shapes and sizes. The outer one is represented as parent and inner one is represented as child.

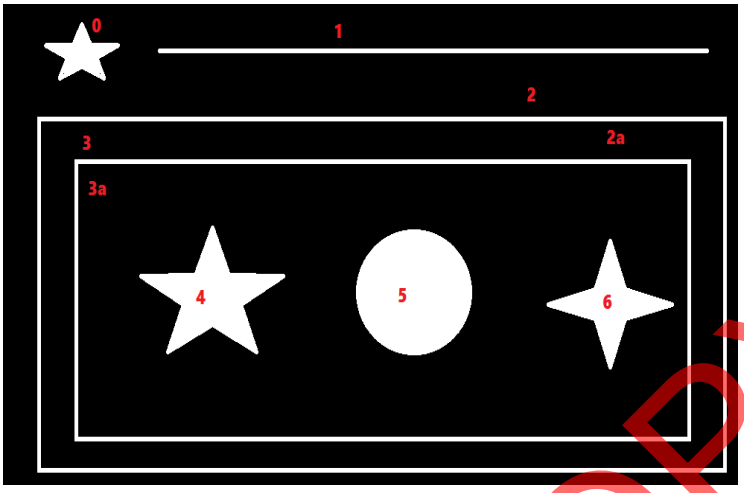



*Figure 24: Hierarchy example*

- Objects are numbered from 0-6
- 2 and 2a denotes external and internal contours
- Hierarchy-0 (0,1,2) [0,1,2 are outermost so they will come in same hierarchy level]
- Hierarchy-1 (2a,3) [since 2a is child of contour 2 so it come in next hierarchy]
- Hierarchy-2 (3a) [child of contour 3]
- Hierarchy-3 (4,5,6) [child of contour 3a and contour 4 can be said as first child of contour 3a]

### 2.5.2.1 Hierarchy Representation in OpenCV:

[Next,Previous,First_Child,Parent]

- Next: Next contour at the same level of hierarchy level [E.g. for contour 0 next is 1]
- Previous: Previous contour at the same hierarchy level [E.g. for contour 0 previous is -1 since no contour previous to 0]
- First_Child: First child contour [E.g. for contour 3a first child is 4]
- Parent : Index of its parent contour [E.g. for Contour 3a Parent is 3]

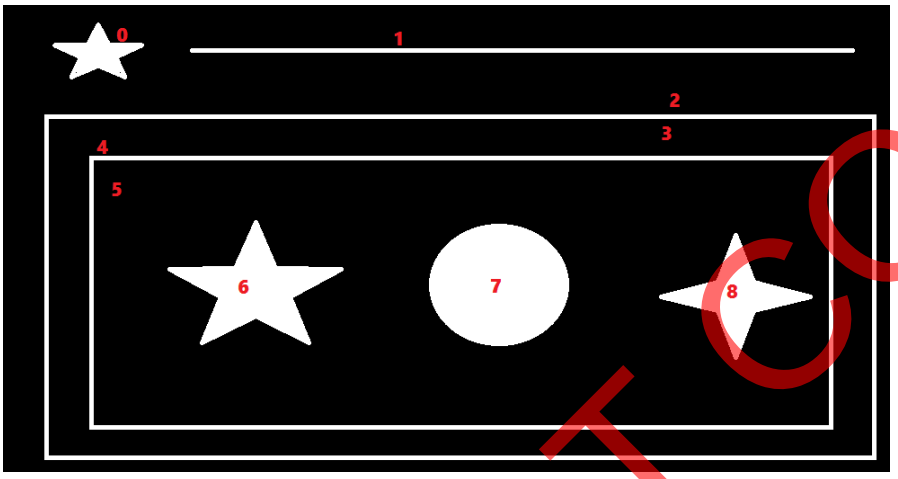
**Note:** If there is no next or previous or parent or child for a contour then field is taken as -1

### 2.5.3 Contour Retrieval Mode

There are four different retrieval mode

**A) RETR_LIST**

- Here all contours are retrieved at same hierarchy level
- No parent-child relationship exits here.
- So here First_Child=-1 and Parent=-1




Output:

```
[[[ 1 -1 -1 -1]
 [ 2  0 -1 -1]
 [ 3  1 -1 -1]
 [ 4  2 -1 -1]
 [ 5  3 -1 -1]
 [ 6  4 -1 -1]
 [ 7  5 -1 -1]
 [ 8  6 -1 -1]
 [-1  7 -1 -1]]]
```

*Figure 25: RETR_LIST example*

For Contour-0:

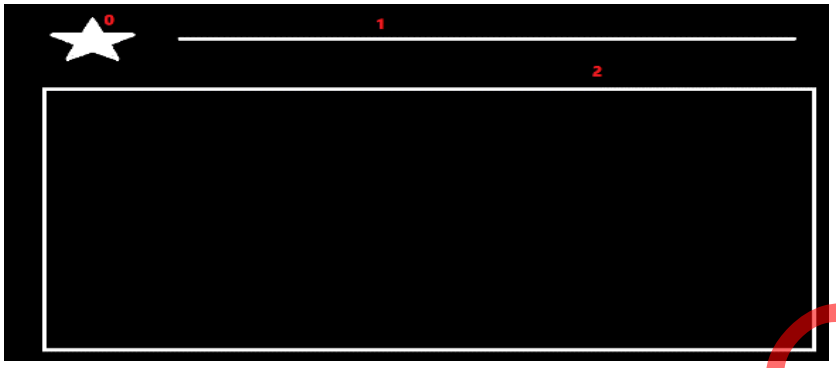
- Next Contour is Contour 1
- No previous contours so -1
- No child
- No parent
- So array is [ 1 -1 -1 -1]

For Contour-1:

- Next Contour is Contour 2
- Previous contour is contour 0
- No child
- No parent
- So array is [ 2  0 -1 -1]

## B) RETR_EXTERNAL

Here all external contours or extreme outer contours are retrieved which is Hierarchy-0 (0,1,2) in our example.




Output:

```
[[[ 1 -1 -1 -1]
  [ 2  0 -1 -1]
  [-1  1 -1 -1]]]
```

*Figure 26: RETR_EXTERNAL example*



For Contour-0:

- Next Contour is Contour 1 which is in same hierarchy(Hierarchy-0)
- No previous contours
- No child
- No parent
- So array is [ 1 -1 -1 -1]

For Contour-1:

- Next Contour is Contour 2 which is in same hierarchy(Hierarchy-0)
- Previous contour is contour 0
- No child
- No parent
- So array is [ 2  0 -1 -1]

For Contour-2:

- Has no next contour so next contour is -1
- Previous contour is contour 1
- No child

- No child
- So array is [-1  1 -1 -1]

## C) RETR_TREE

- Retrieves all the contours and creates a full hierarchy list
- Here contours are reordered.
- Red letters is the contour number and blue letter is the contour hierarchy

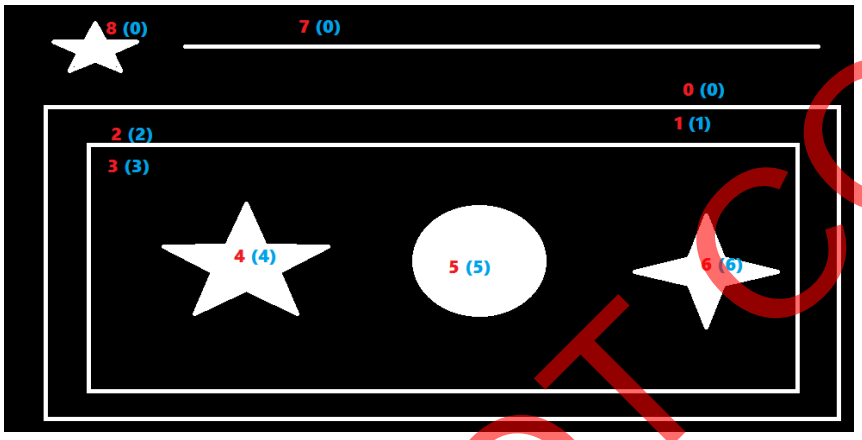



*Figure 27: RETR_TREE example*

Output:

```
[[[ 7 -1  1 -1]
  [-1 -1  2  0]
  [-1 -1  3  1]
  [-1 -1  4  2]
  [ 5 -1 -1  3]
  [ 6  4 -1  3]
  [-1  5 -1  3]
  [ 8  0 -1 -1]
  [-1  7 -1 -1]]]
```

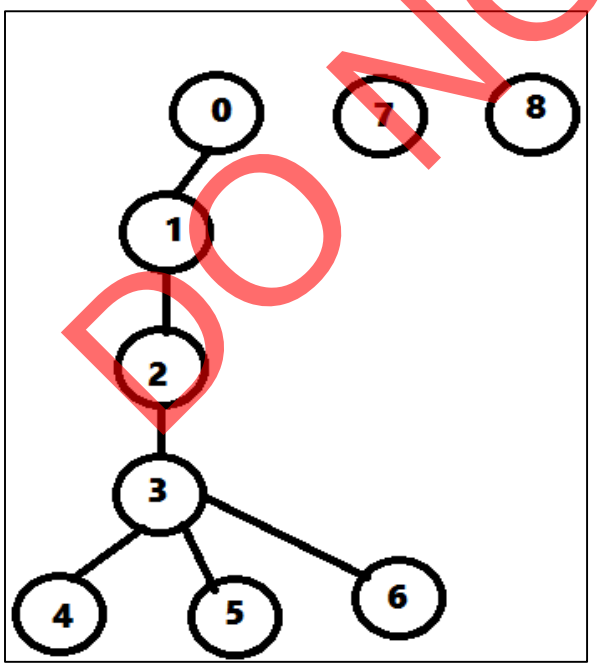



*Figure 28: RETR_TREE graph*

<u>For Contour-0:</u>

- Next Contour is Contour 7 which is in same hierarchy(Hierarchy-0)
- No previous contours
- contour-1 is the child
- No parent
- So array is [7,-1,1,-1]

For Contour-1:

- No one is same hierarchy level so next contour is -1
- No previous contours
- Contour-2 is the child
- Contour-0 is parent
- So array is [-1,-1,2,0]

## D) RETR_CCOMP

- Here it is retrieves all contours and arranges them into two hierarchy level
- Red letters is the contour number and blue letter is the contour hierarchy
- External contours (boundary of objects) belong to hierarchy-1 (E.g. Contour-2)
- Internal contours (Inside portion of object) belong to hierarchy-2. (E.g. Contour-3)
- If any Object is placed inside Internal contours then that object(E.g. Contour-0)belong to hierarchy-1 and its hole in hierarchy-2(E.g. Contour-1)

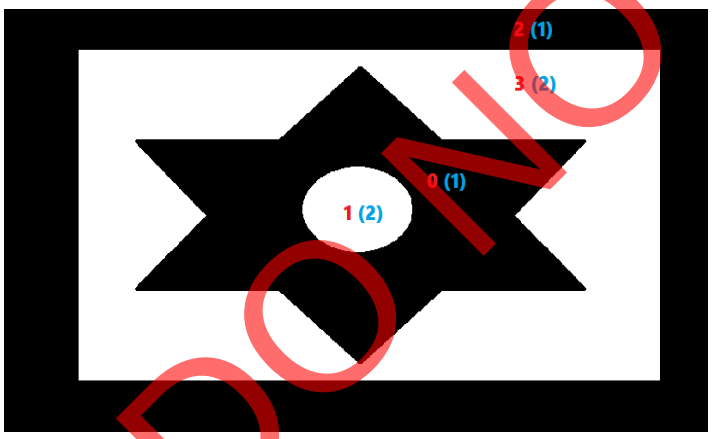



*Figure 29: RETR_CCOMP example*

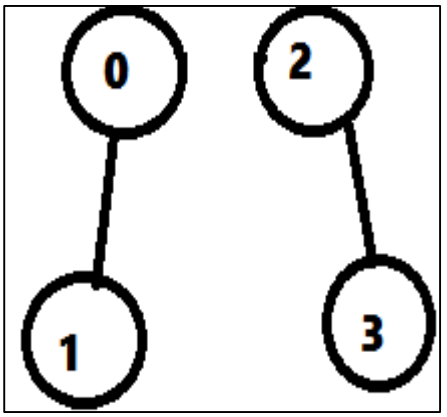



*Figure 30: RETR_CCOMP graph*

Output:

```
[[[ 2 -1  1 -1]
  [-1 -1 -1  0]
  [-1  0  3 -1]
  [-1 -1 -1  2]]]
```

For Contour-0:

- Next Contour is Contour 2 which is in same hierarchy(Hierarchy-1)
- No previous contours
- contour-1 is the child
- No parent
- So array is [2,-1,1,-1]

For Contour-2:

- Has no next so next contour is -1
- Previous contour is 0
- contour-3 is the child
- Has no parent
- So array is [-1 0 3 -1]

For Contour-1:

- Has no siblings so next contour is -1
- No previous contours
- Has no child
- Contour 0 is the parent
- So array is [-1 -1 -1 0]

For Contour-3:

- Has no siblings so next contour is -1
- No previous contours
- Has no child
- Contour 2 is the parent
- So array is [-1 -1 -1 2]

### 2.5.4 Contour Approximation Method

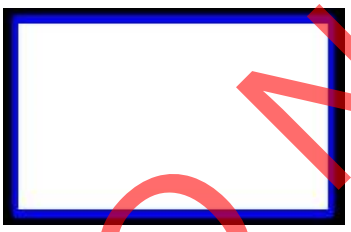


*Figure 31*

*Figure 32*

*Figure 31: Output of Chain Approximation None*

*Figure 32: Output of Chain Approximation Simple*

**1) CHAIN APPROX NONE**

- Retrieves all boundary points to determine the shape of the image
- Disadvantage: Conserve Memory (Since all boundary points are not required since we need only the corners)

**2) CHAIN APPROX SIMPLE**

- Retrieve the required boundary points (example 4 points in Figure 32) to determine the shape
- Advantage: Saves the memory

## 2.6 Overview of Tesseract

Tesseract is an open-source OCR engine. It is a powerful optical character recognition (OCR) engine which supports more than 100 languages. The engine is highly configurable and the detection algorithms can be tuned to obtain the results with high accuracy. The Tesseract engine uses language-specific training data for recognizing words. Like human brain, the OCR algorithms bias towards words and sentences that frequently appear together in a given language. Therefore using training data in the correct language fetches the best results. The accuracy of the OCR process however depends largely on the quality of the input image. Tesseract works best for scaled images. We can improve results by properly scaling the image, removing noise and artifacts or cropping the area where the text exists or use any other pre-processing techniques before feeding it to the OCR engine. It also supports hundreds of control parameters which can customize the OCR engine. It also has a handy filter arguments to quickly find parameters that match a particular string.

### 2.6.1 Tesseract Architecture

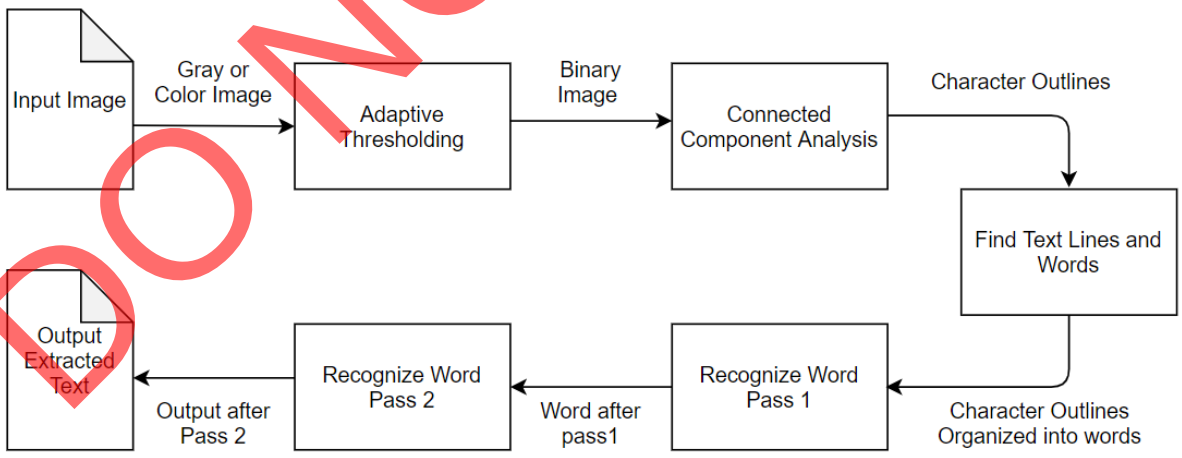



*Figure 33: Tesseract Architecture*

**Different steps of Tesseract Architecture**

1) Convert the image into binary image by applying adaptive thresholding that minimize within-group variance so we can easily identify background and foreground images.

2) Character component analysis is done on the image which finds out the connected objects from image, this objects are converted into blobs.

3) Find lines and words: Blobs are organized as text lines and from the text lines we find out the words which is separated by definite spaces and fuzzy spaces.

4) Text recognition takes place into two passes:

a) In first pass: each word is recognized from text and passed to adaptive classifier as training data

b) In second pass: adaptive classifier is sensitive to different types of fonts and also help to distinguish between upper and lower case letters and results out the text as the output.

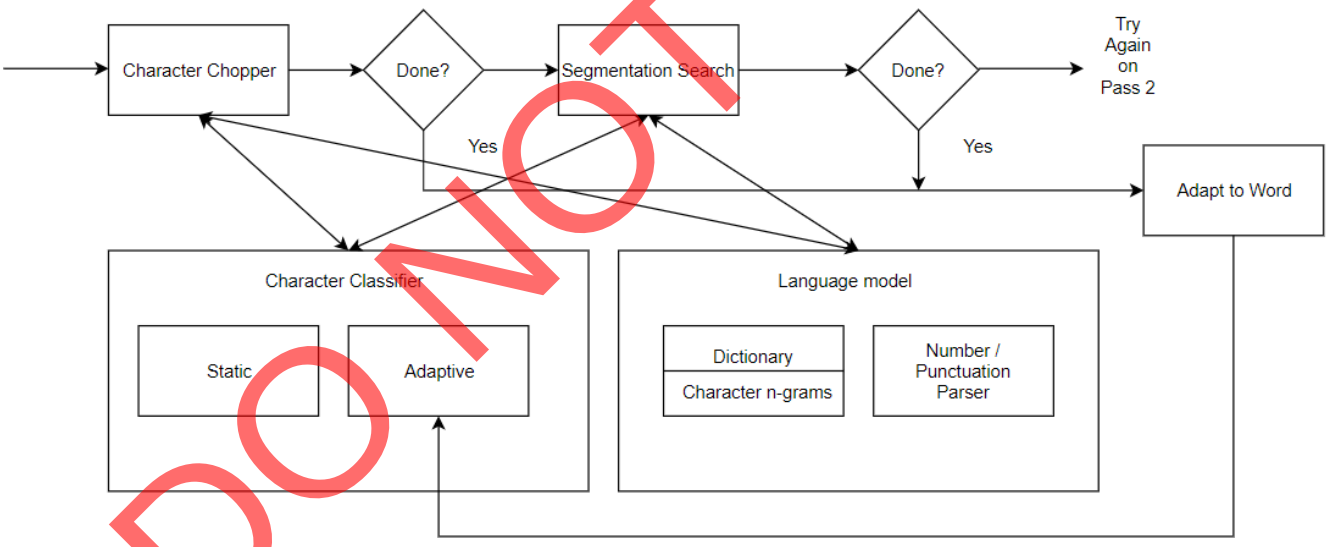



*Figure 34: Tesseract word recognizer*

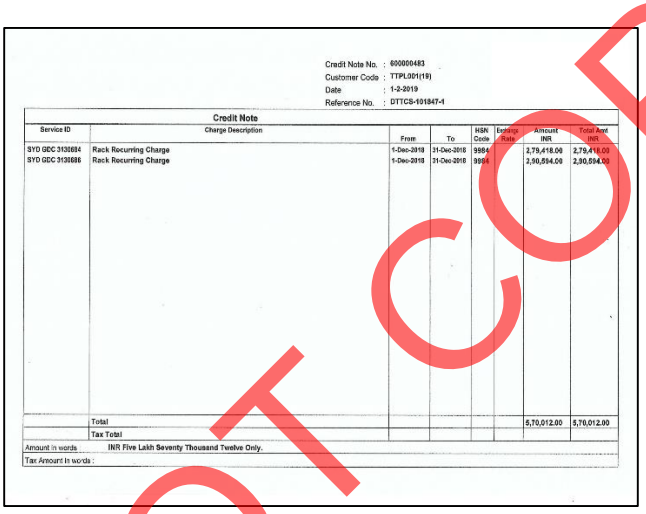
# Chapter 3

# Implementation Approach

## 3.1 Algorithm Overview

**Step 1:** Read pdf file.

**Step 2:** Convert the scanned pdf file into images.



Credit Note No. : 600000483
Customer Code : TTPL001(19)
Date : 1-2-2019
Reference No. : DTTCS-101847-I

**Credit Note**

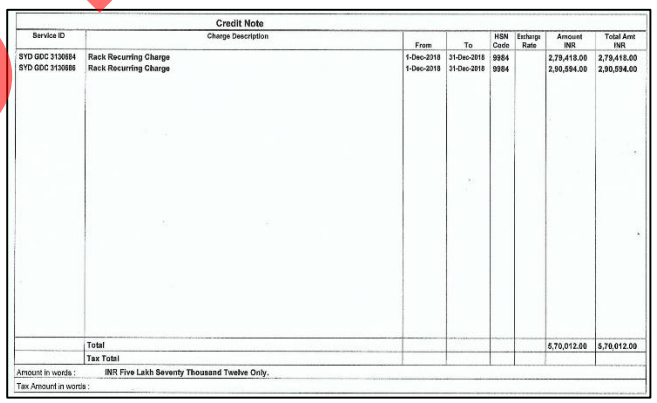
| Service ID | Charge Description | From | To | HSN Code | Exchange Rate | Amount INR | Total Amt INR |
|---|---|---|---|---|---|---|---|
| SYD GDC 3130684 | Rack Recurring Charge | 1-Dec-2018 | 31-Dec-2018 | 9984 | | 2,79,418.00 | 2,79,418.00 |
| SYD GDC 3130686 | Rack Recurring Charge | 1-Dec-2018 | 31-Dec-2018 | 9984 | | 2,90,594.00 | 2,90,594.00 |
| | Total | | | | | 5,70,012.00 | 5,70,012.00 |
| | Tax Total | | | | | | |

Amount in words : INR Five Lakh Seventy Thousand Twelve Only.
Tax Amount in words :

*Figure 35: Input Image for OCR*

**Step 3:** Pre-process and enhancing the quality of the image (image resizing, noise removal)

**Step 4:** Identifying the Table from the image.



**Credit Note**

| Service ID | Charge Description | From | To | HSN Code | Exchange Rate | Amount INR | Total Amt INR |
|---|---|---|---|---|---|---|---|
| SYD GDC 3130684 | Rack Recurring Charge | 1-Dec-2018 | 31-Dec-2018 | 9984 | | 2,79,418.00 | 2,79,418.00 |
| SYD GDC 3130686 | Rack Recurring Charge | 1-Dec-2018 | 31-Dec-2018 | 9984 | | 2,90,594.00 | 2,90,594.00 |
| | Total | | | | | 5,70,012.00 | 5,70,012.00 |
| | Tax Total | | | | | | |

Amount in words : INR Five Lakh Seventy Thousand Twelve Only.
Tax Amount in words :

*Figure 36: Table detection from image*

**Step 5:** Information retrieval from Tabular Structure

a) Thresholding the image. : Helps to create Binary form of image from a colour or grayscale images. In this technique image partition into two sections foreground and background and removes background (noise) section from image which helps for better image analysis.

b) Invert the image. : Colours of image are replaced to their colour complements which make image much contrast used for image enhancement.

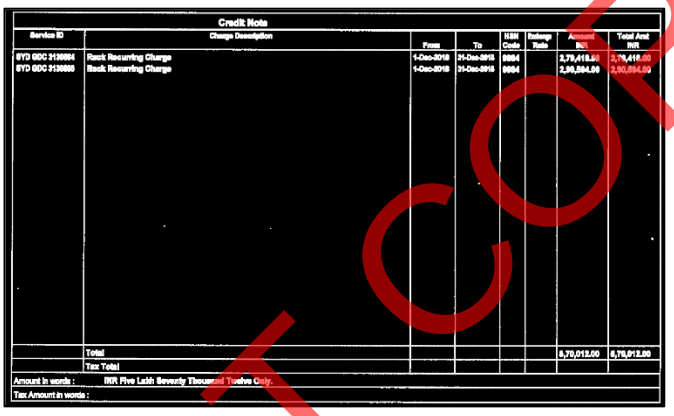



*Figure 37: Binary threshold inverted image*

c) Apply morphological operation to detect vertical lines.

*Figure 38: Detect vertical lines*

d) Apply morphological operation to detect horizontal lines

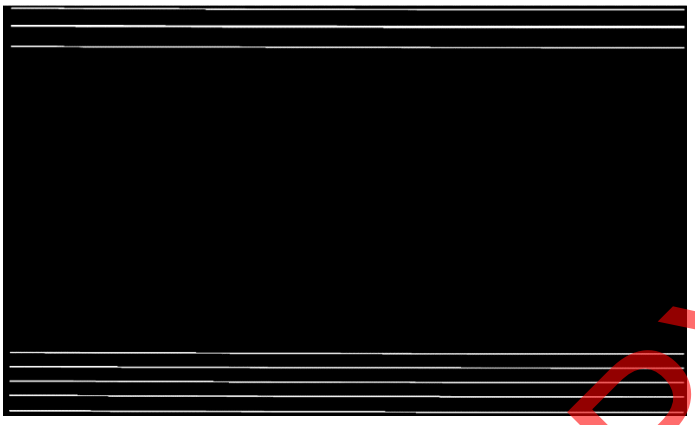


*Figure 39: Detect horizontal lines*

e) Superimpose both images into a third one which will have intersecting horizontal and vertical lines.

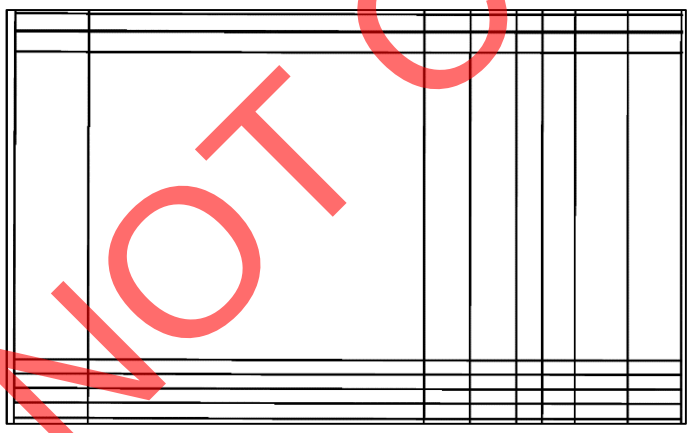


*Figure 40: Intersection of vertical and horizontal lines*

f) Find bounding boxes or contours for final image, which will detect all the boxes.

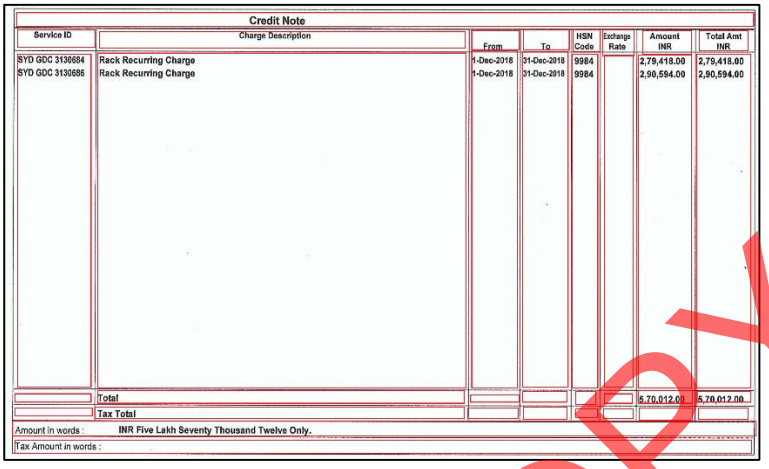



*Figure 41: Bounding boxes from tables*

g) Sorting the boxes based on top to bottom / left to right.

Generated boxes as shown below

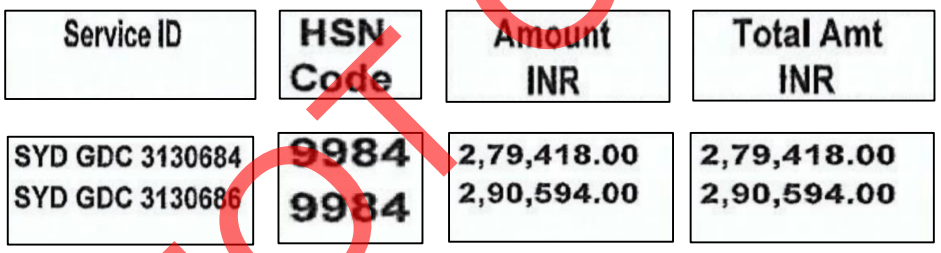



*Figure 42: Cell detection*

h) Loop over all the contours

i. find the location of all the boxes

ii. crop the part which has a rectangle

iii. save cropped images in a folder

**Step 6:** Loop over all the cropped images.

**Step 7:** Pre-process and enhancing the quality of the image (image resizing, noise removal)

**Step 8:** Run OCR engine (Tesseract OCR engine) to recognize text from the images.

**Step 9:** Process the extracted text and filter out the required data.

**Step 10:** Repeat steps 5 to 8 until all the images are processed.

**Step 11:** End.

## 3.2 Flowchart displaying Proposed Algorithm work flow

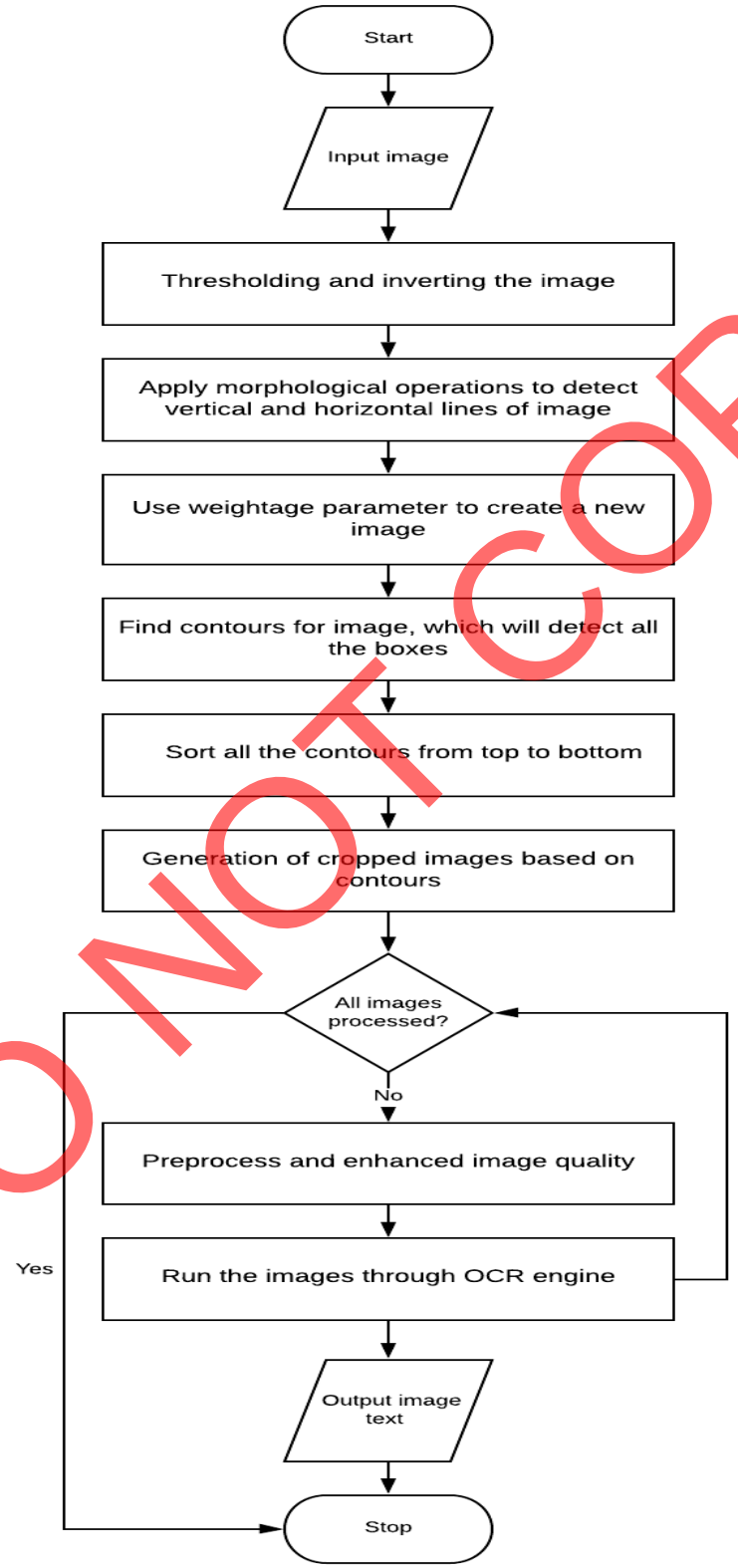



*Figure 43: Flowchart of proposed algorithm*

# Chapter 4

## Application Overview

Proposed Application workflow is depicted in the diagram

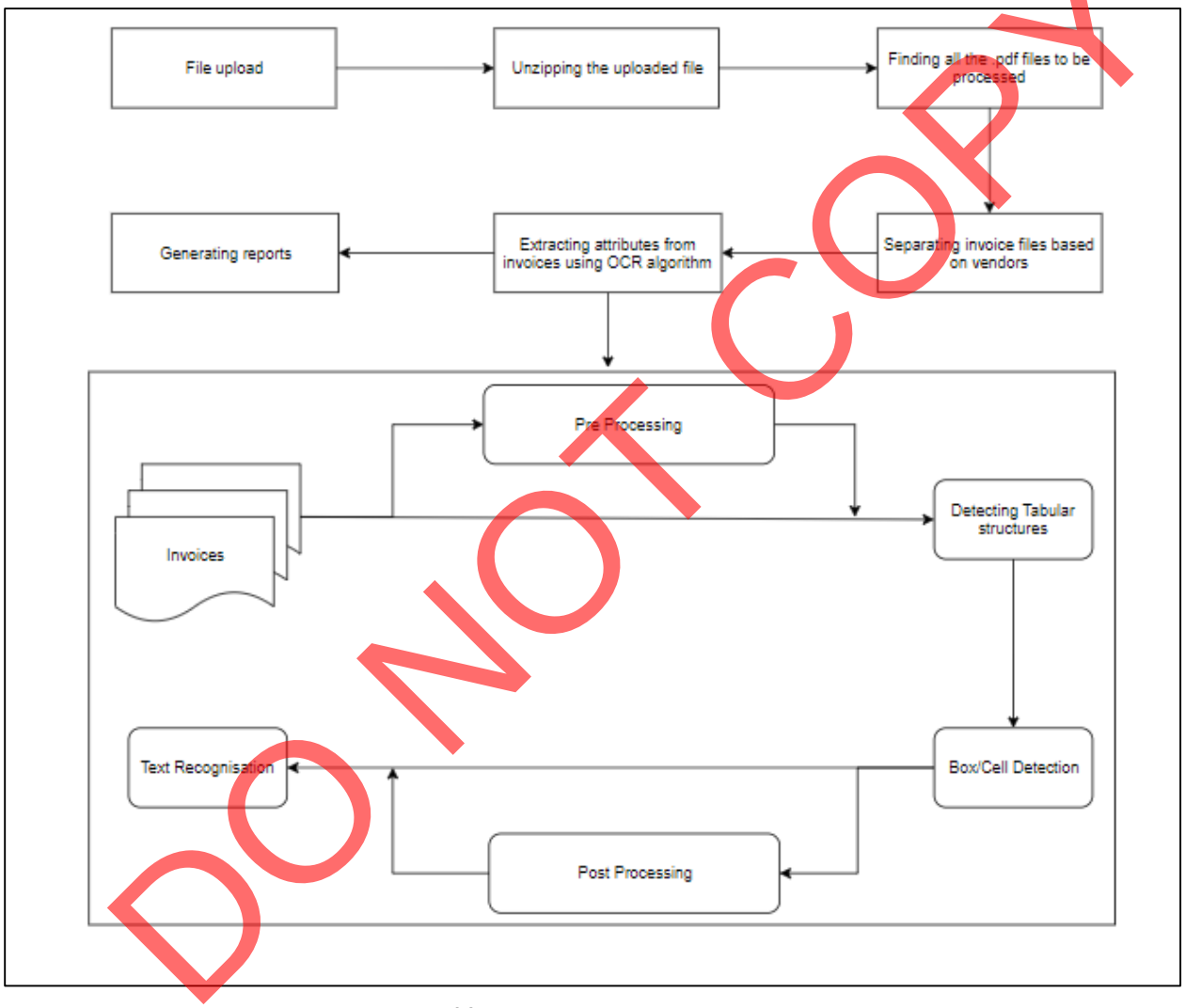





*Figure 44: Application Overview*

Invoice processing application leverages the potential of Optical character resolution for maintaining and archiving information pertaining to invoices. The proposed application follows microservice architecture and is designed to host two different services and provisions have been made to accommodate more services when needed. The application uses the concept of message passing between two services as a way of communication. The application is designed to follow producer/consumer model where producer application publishes messages which are subscribed by the consumer. Since the application follows microservice architecture, multiple instances of the services can be deployed and multiple producers and consumers can be configured to fulfil the purpose of bulk invoice processing. The application has the flexibility to add more processing capability and can process different sets of invoices. We can create additional services for other operations as well without impacting the core application.

The application displays real time processing information which gives the user idea about the current status of the process. It also provides user to upload multiple invoices and process all of them in a single go. User can start multiple jobs simultaneously and can review the status of each job from time to time. The application performs different image processing techniques and certain complex calculations to extract tabular structure from scanned documents with utmost accuracy. It is intelligent enough to determine the type of invoices and apply appropriate techniques to extract required information from them. It performs coordinate based information retrieval from different sections of the image. The proposed application has the capability to generate consolidated reports specified by the user. It is flexible enough to generate specific audit reports, reconciliatory reports and other user defined reports. Moreover, the application provides features like access management and identity management and is flexible enough to provide user specific roles. The proposed application has the capability to accommodate a greater number of users and assign them different levels of roles. It can be easily integrated with other enterprise application and serve large set of clients. It is a single page application and requires minimum time to load and requires minimal resources to process.

There are different steps involved in processing the application. Since the application follows microservice architecture, different services are responsible for performing different operation.

Steps performed by first service are:

1. **File Upload:** End user upload a compressed .7z file which contains all the invoices to be processed. User has the flexibility to upload file as large as 60MB.
2. **Unzipping and Validating Files:** The input file is unzipped and validated for any errors.
3. **Finding PDF documents to be processed:** Application crawls through all the files and folders present in the upload file and selects a consolidated list of files to be processed.

After all the three steps have been performed the first service acts as a producer and sends all the necessary information required for processing to the second service. After consuming the information, the second microservice starts with its processing.

Steps involved with the second service are:

4. **Post Invoice Upload:** User has the flexibility to upload additional files pertaining to invoices for archival and reconciliatory purpose.
5. **Invoice Segregation:** The application extracts specific information from the invoices based on certain coordinates and segregates invoices based on the information.
6. **Extracting Raw Data:** The application performs raw data extraction from the invoices using coordinated based data retrieval.
7. **Extracting Invoice Attributes:** The applications performs certain logical operations and image processing techniques to extract invoice attributes required for reconciliation.
8. **Comparison with PO:** User can perform certain reconciliatory operation on the invoices with Purchase Order file. It helps to weed out all the discrepancies present in the invoices.
9. **Reports Generation:** The final step involves generating different types of reports.

# Chapter 5

# User Interface Snapshots

## 5.1 User Interface depicting the Login/Register Page

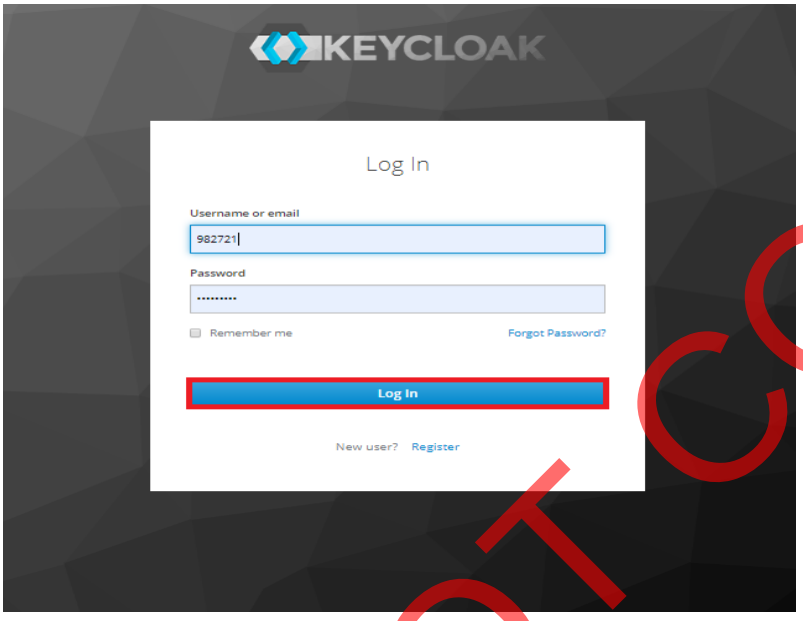



*Figure 45: Screenshot of proposed Application Login/Register Page*

## 5.2 User Interface depicting Application Home Page

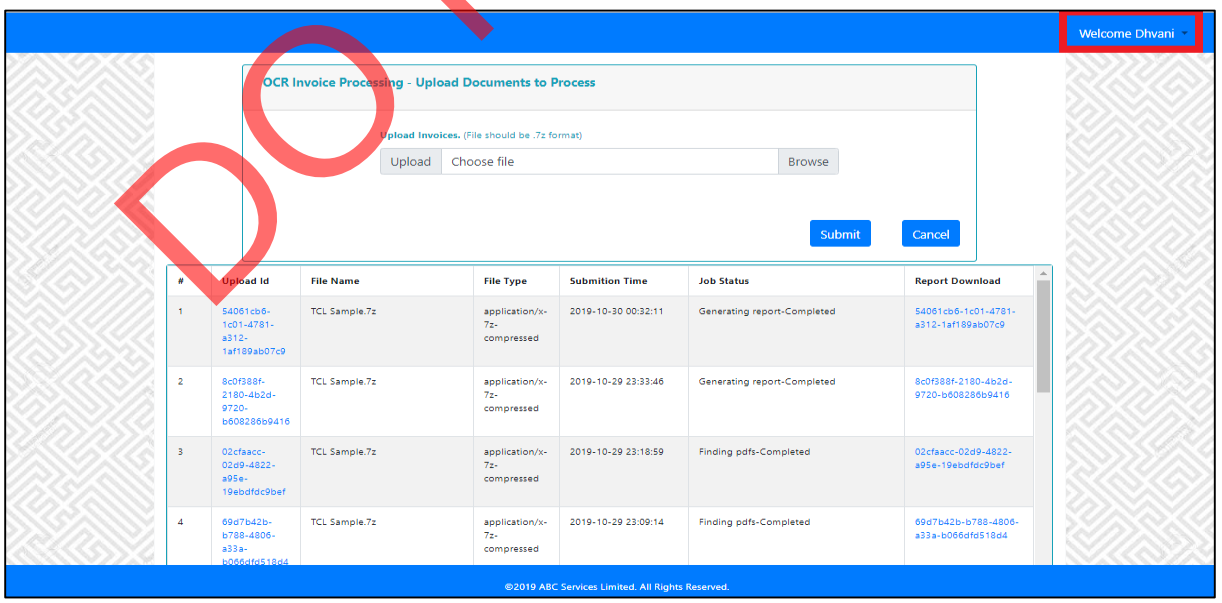



*Figure 46: Screenshot of proposed Application Home Page and Dashboard*

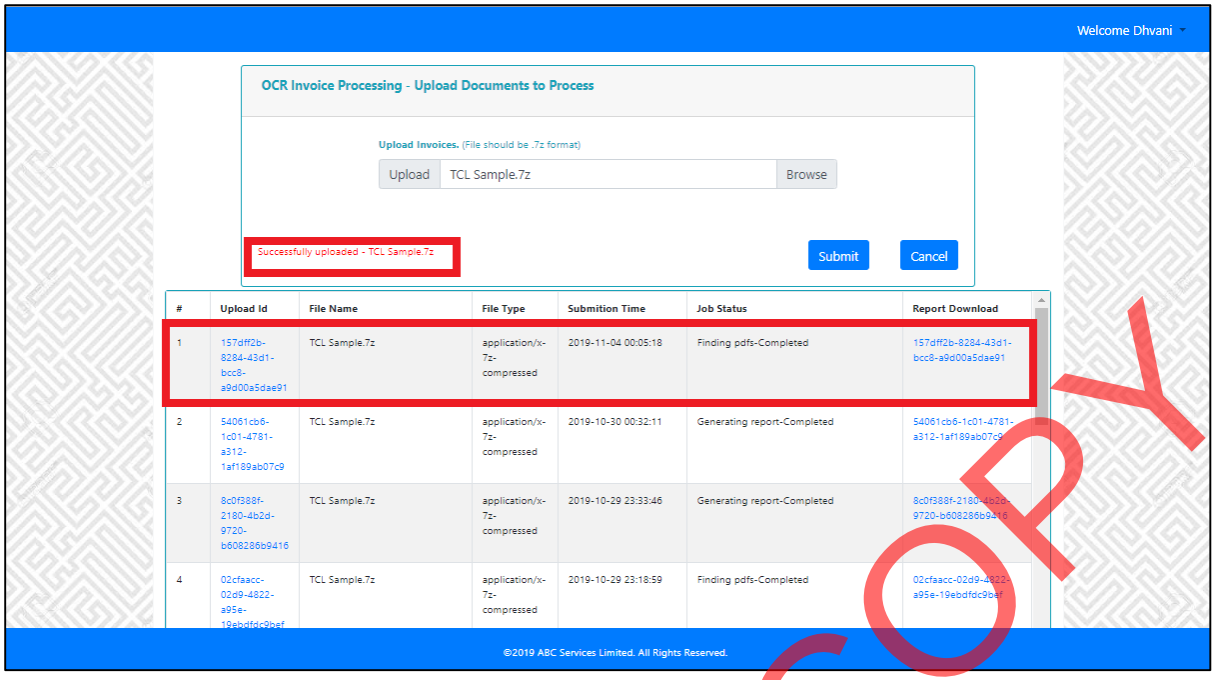



*Figure 47: Screenshot depicting File Upload operation and new Upload ID generation*

## 5.3 User Interface depicting process completion Dashboard

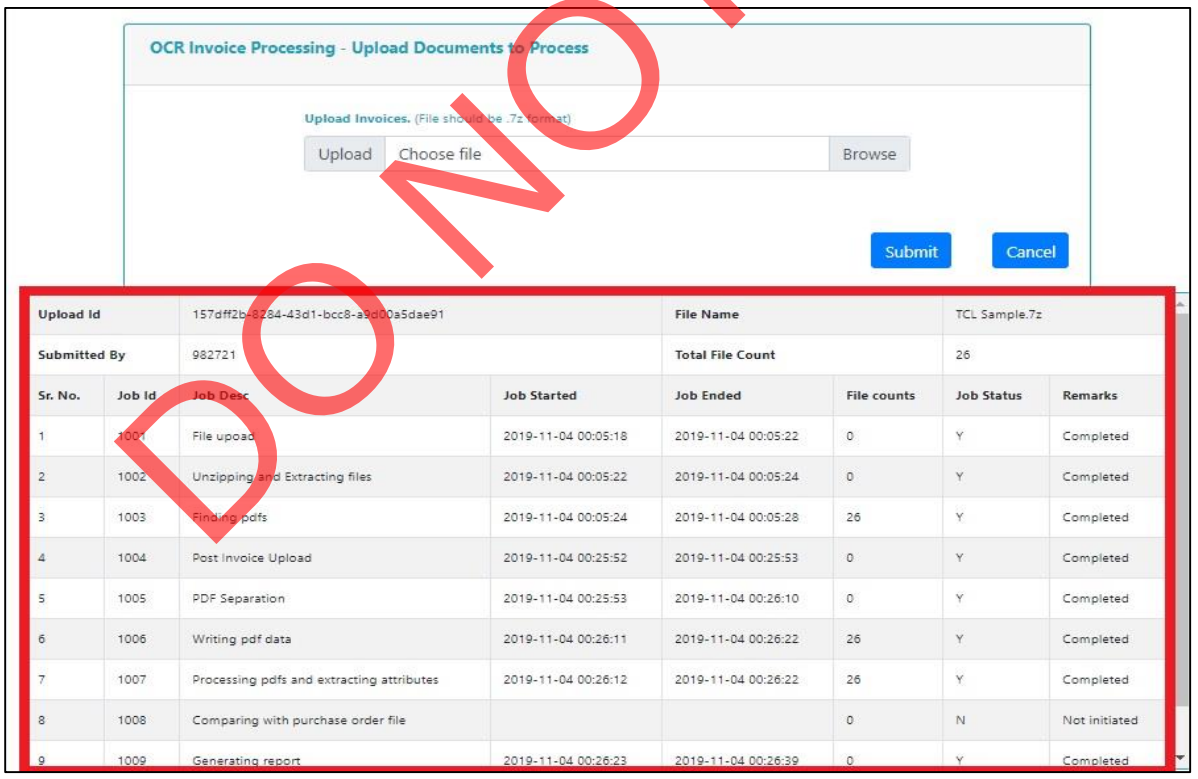



*Figure 48: Screenshot depicting the completion status and time of individual processes involved in invoice processing*

## 5.4 User Interface depicting Reports Dashboard

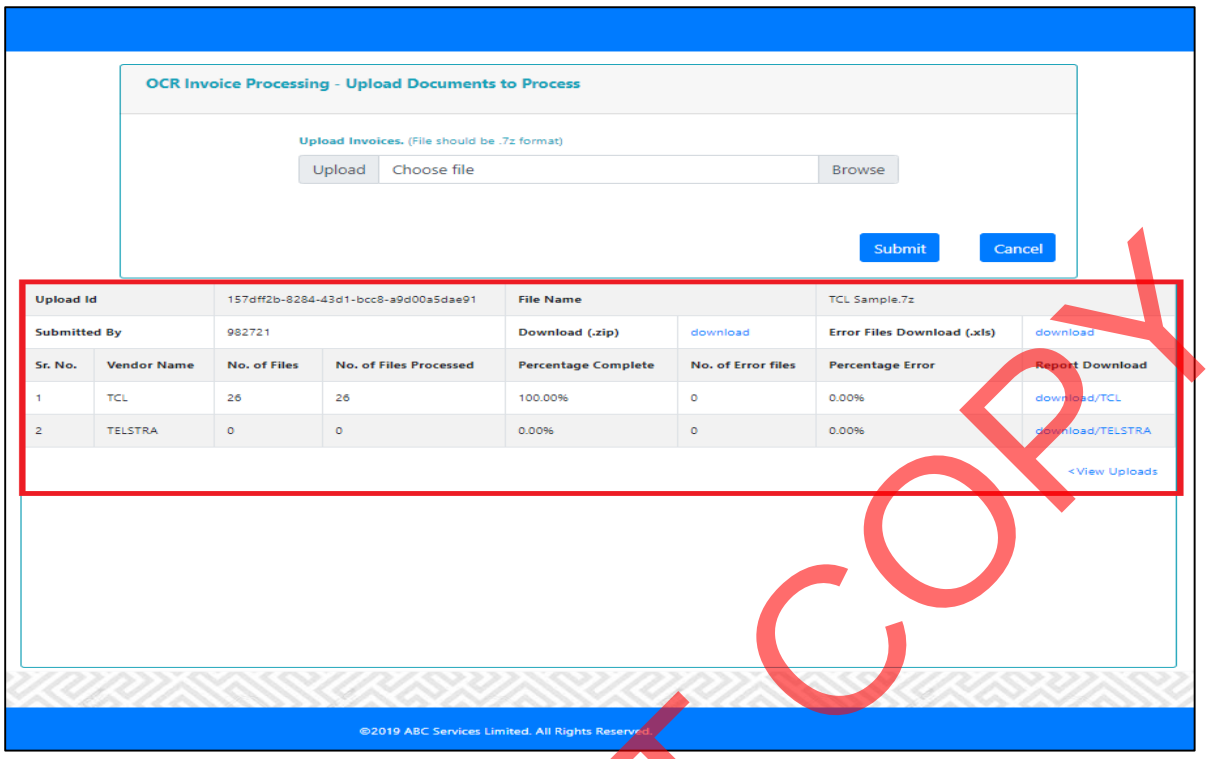



*Figure 49: Screenshot depicting the percentages of completion of invoice processes of different vendors and report generation*

## 5.5 User Interface depicting Reports generation

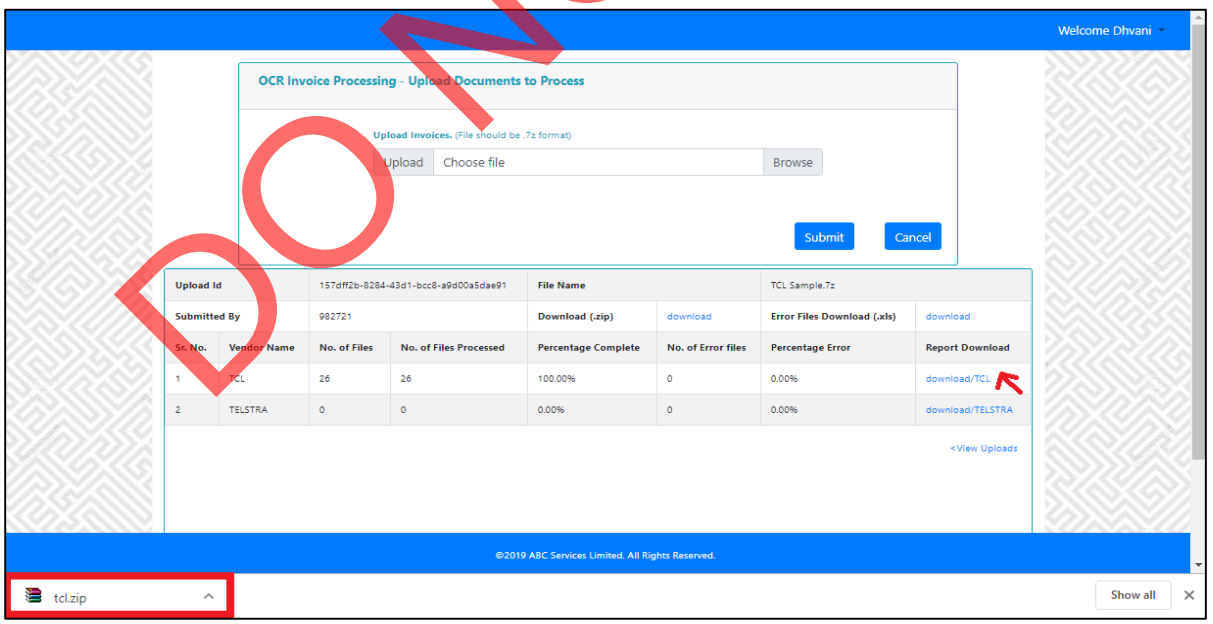



*Figure 50: Screenshot depicting process completion and reports download*

# Chapter 6

## 6.1 Summary

The idea behind this project is to help users replace manual invoice processing and provide a dynamic solution wherein the user can upload invoices and thereby get information from those invoices as needed. This should be a cost-effective solution for invoice processing where the correctness can be achieved at a maximum level. The application should be able to extract information from readable pdfs as well as electronic conversion of images of typed, handwritten or printed text. It has the capability to extract tabular information from different parts of pdfs and present the same in a format prescribed by the user. The application is able to compute various mathematical computations and thereby segregating the erroneous invoices. The application uses Optical Character recognition and is able to read data from scanned images and provide the use with correct information for reconciliation or comparison purpose. It is a holistic approach for processing everything associated with an invoice.

### 6.1.1 Techniques used for this process:

1) Segmentation: The idea of segmentation revolves around partitioning an image into meaningful regions and group together pixels with similar properties.
2) Thresholding: Thresholding partitions the complete image by a threshold thereafter segmentation is achieved by scanning each pixel and labeling it as background or foreground depending on the gray level of that pixel.
3) Morphological Operators: find the shape and size or the structure of the object.

### 6.1.2 OCR engine used: Tesseract

Tesseract is an optical character recognition engine helps to convert scanned non-readable documents into machine readable form. It accept any image format(.jpg,.png,.tiff) Tesseract can read the sentences from left to right for the languages like English, Spanish and also read the text from right to left for the languages like Hebrew or Arabic.

# Chapter 7

## 7.1 Conclusion

Invoice Processing is one of the major challenges that most of the organization has to deal with. Manual intervention in invoice processing often leads to erroneous data entered in the system resulting in incorrect report generation. Optical Character Recognition tries to address this issue to a great extent. Pre-processing the image plays vital role in data extraction. Various image enhancement and noise removal techniques are used for cleaning and enhancing image quality. Image segmentation combined with various morphological operations helps extract data from images where there are tabular structures present. The proposed algorithm leverages all the techniques to extract invoice information and generate specific reports for auditing purpose. Data from complex tabular structure can be easily extracted. Also, it has the capability to extract information from different coordinates of an image. The OCR engine can be further trained to detect complex characters as well.

## 7.2 Recommendations

- The scanned copies which are used for processing should be of good quality with low blurriness.
- Skewed document may lead to inaccurate result.

# Chapter 8

## Directions for future work

1) **More Accuracy:** The proposed algorithm can be improved further to provide more accurate results.
2) **Faster Processing:** Speed of recognizing characters in scanned documents can be improved further.
3) **Text reconstruction and detection:** The proposed algorithm can be extended to reconstruct and detect broken characters and poor quality texts.
4) **Detection of different fonts:** The application can be further trained to detect diverse set of characters in an image.
5) **Detection of complex tabular structure:** Improvements can be done to detect more complex tabular structures in scanned text documents.
6) **Handwriting Detection:** The proposed algorithm can be extended to detect handwritten characters.

# Chapter 9

**References:**


[1] Neetu Bhatia (2014).Optical Character Recognition Techniques: A Review. In International Journal of Advanced Research in Computer Science and Software Engineering, Volume 4, Issue 5, May 2014

[2] Gidi Shperber (2018). A gentle introduction to OCR. Retrieved from https://towardsdatascience.com/a-gentle-introduction-to-ocr-ee1469a201aa

[3] Kanan Vyas (2018). A Box detection algorithm for any image containing boxes. Retrieved from https://medium.com/coinmonks/a-box-detection-algorithm-for-any-image-containing-boxes-756c15d7ed26

[4] Hiral Modi, & M. C. Parikh, PhD (2017).A Review on Optical Character Recognition Techniques. In International Journal of Computer Applications (0975 – 8887) Volume 160 – No 6, February 2017

[5] Adrian Rosebrock (2015). Sorting Contours using Python and OpenCV. Retrieved from https://www.pyimagesearch.com/2015/04/20/sorting-contours-using-python-and-opencv/

[6] Tesseract OCR. Retrieved from https://github.com/tesseract-ocr/tesseract

[7] R. Zanibbi, D. Blostein & J.R. Cordy (2003). A Survey of Table Recognition: Models,Observations, Transformations, and Inferences.

[8] Dharampal, Mutneja V (2015). Methods of Image Edge Detection: A Review. In Journal of Electrical & Electronic Systems DOI: 10.4172/2332-0796.1000150.

[9] Dr. S.Vijayarani1 & Ms. A.Sakila. Performance Comparison of OCR Tools. In International Journal of UbiComp (IJU), Vol.6, No.3, July 2015

[10] Chirag Patel, Atul Patel, & Dharmendra Patel (2012). Optical Character Recognition by Open source OCR Tool Tesseract: A Case Study. In International Journal of Computer Applications (0975 – 8887) Volume 55– No.10 October 2012

[11] Selcuk Algun (2018). Review for Tesseract and Kraken OCR for text recognition. Retrieved from https://medium.com/datadriveninvestor/review-for-tesseract-and-kraken-ocr-for-text-recognition-2e63c2adedd0

[12] Jonathan Chung (2018). Handwriting OCR: handwriting recognition and language modeling with MXNet Gluon. Retrieved from https://medium.com/apache-mxnet/handwriting-ocr-handwriting-recognition-and-language-modeling-with-mxnet-gluon-4c7165788c67

[13] Graves, A., & Schmidhuber, J. (2009). Offline handwriting recognition with multidimensional recurrent neural networks. In Advances in neural information processing systems (pp. 545–552).

[14] Graves, A., Fernández, S., Gomez, F., & Schmidhuber, J. (2006, June). Connectionist temporal classification: labelling unsegmented sequence data with recurrent neural networks. In Proceedings of the 23rd international conference on Machine learning (pp. 369–376). ACM.