CHRIST
UNIVERSITY
BENGALURU, INDIA
Declared as Deemed to be University under Section 3 of UGC Act 1956

IMAGE PROCESSING TO VERIFY RECEIPT OR NOT

DEPARTMENT OF PHYSICS AND ELECTRONICS (CHRIST (DEEMED TO BE UNIVERSITY))

PROJECT BY:

Surya V

2040164

B.Sc. (5CME)

Department of Physics and Electronics

Christ(Deemed to University) Bangalore

Certificate

This is to certify that this is a bonafide record of the project presented by the student and his name is given below during Mid-semester for third-Year student in partial fulfilment of the requirements of the degree of Bachelor of Computer Science and Electronics and Mathematics.

Roll No	Names of the Student
2040164	**SURYA V**

Prof. Vineeth V Physics And Electronics
(Project Guide)

¡Coordinator name here¿
(Course Coordinator)

Date: 19 September, 2022

Abstract:

In this project we mainly divide into four segments such as the Identification of Receipt workflow

Working with images: Import, display and manipulate colour and grayscale images;

Segmenting an Image: Create binary images by thresholding pixel intensity values

Pre- and post-processing Techniques: Improve an image segmentation using common pre- and post-processing techniques

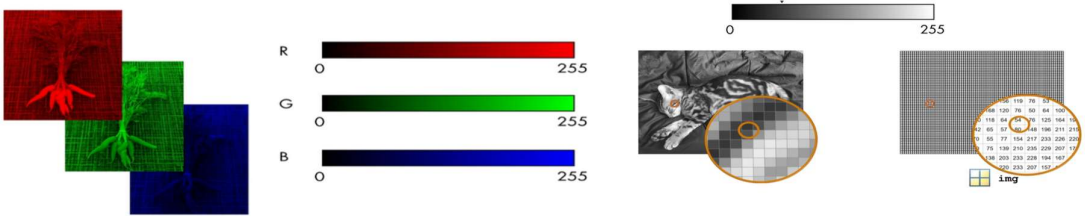
Classification and Batch processing: Develop a metric to classify an image and apply that metric to set of images files

IMPORTING:

The first step is importing images into the MATLAB Workspace. Once in the workspace, you can use MATLAB functions to display, process, and classify the images.

```
I = imread("IMG_002.jpg");
imshow(I)
```

Here we can see the output of image and its code, but here these images are 3D in order to convert them into 2D



Since 2D images have less size and less collar intensity compared to 3D this makes our algorithm to work faster and easier

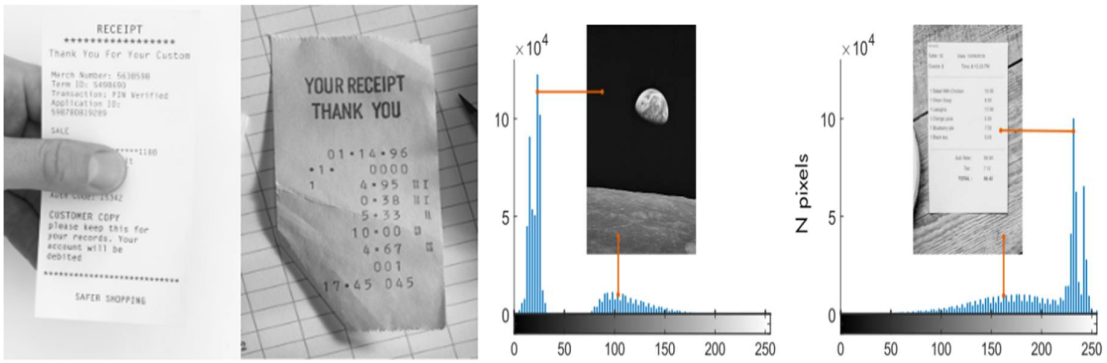
```
img = imread("IMG_001.jpg");
gs = im2gray(img);
gsAdj = imadjust(gs);
imshow(gsAdj)
```

PREPROCESSING:

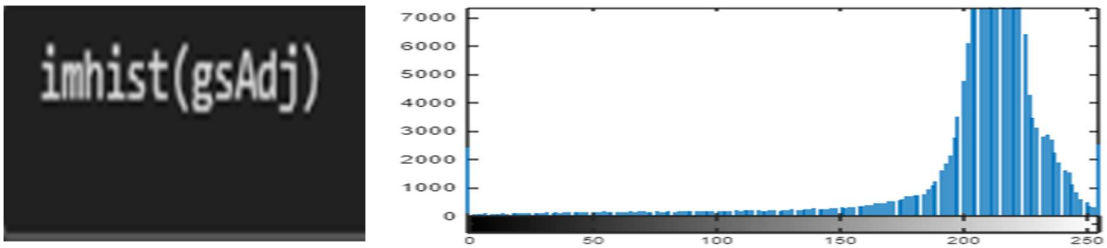
Why is important to convert grayscale?

1. When loaded into memory, a grayscale image occupies a third of the space required for an RGB image. Because a grayscale image has a third of the data, it requires less computational power to process and can reduce computation time. A grayscale image is conceptually simpler than an RGB image, so developing an image processing algorithm can be more straightforward when working with grayscale. In receipt images, like the one shown below, little information is lost when converting to grayscale. The essential characteristics, like the dark text and bright paper, are preserved. On the other hand, if you wanted to classify the produce visible in the background, the colour planes would be essential. After converting to grayscale, it's easy to apply the classification method, let alone classify it.

2. Even though these two receipt images are grayscale, the contrast is different. If you are analysing a set of images, normalizing the brightness can be an important pre-processing step, especially for identifying the black and white patterns of text in receipt images. **Here One image is brighter than other to avoid this we can use contrast and intensity Histograms**



An intensity histogram separates pixels into bins based on their intensity values. Dark images, for example, have many pixels binned in the low end of the histogram. Bright regions have pixels binned at the high end of the histogram

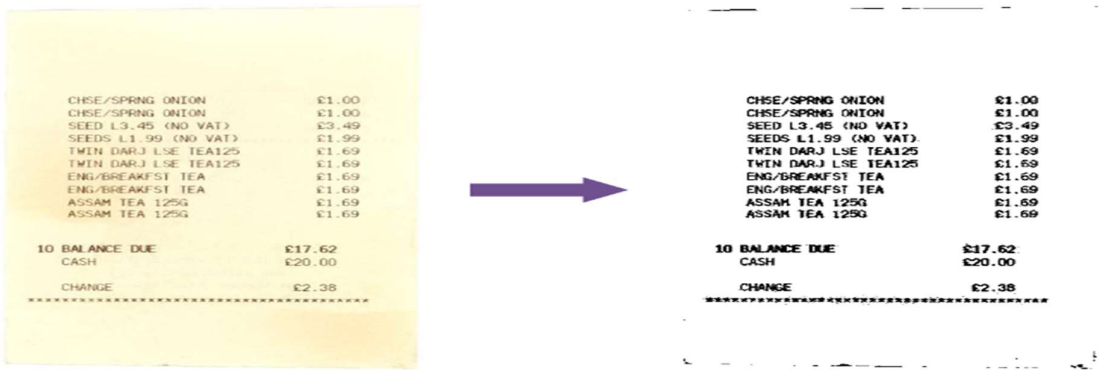


In imhist () it will tell the colour intensity of maximum numbers presented in the image for example this image has a greater number of intensities from 200 to 255.Since we can see the histogram graph now, we want to adjust the histogram graph

In the image of receipt, we can see the left side image is less bright than right side when we use the misadjust it automatically increases the light intensity of the picture.

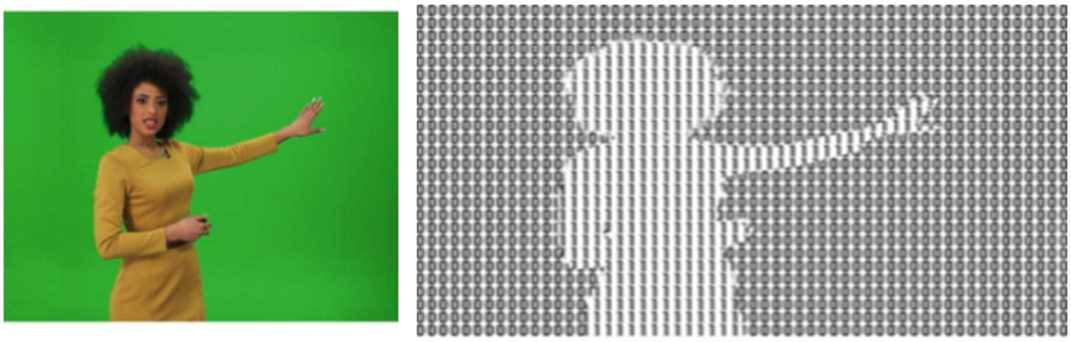
SEGMENTING THE TEXT:

The main feature of receipt images is dark text on a bright background. How can you leverage this? One method is to separate the text from the background. Separating an image into distinct parts is called segmenting an image.



A receipt image has a pattern of horizontal "stripes" of dark text separated by "stripes" of bright paper.

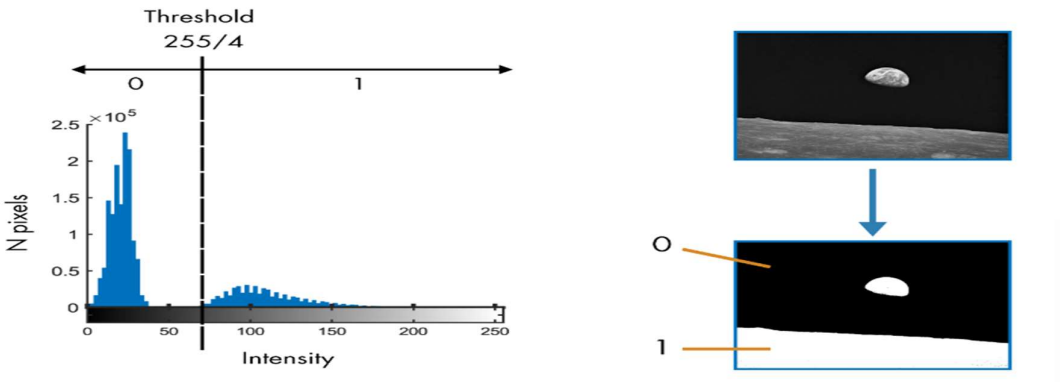
There are different types such as edge detection, texture, shape and size, segmenting by colour, whatever the uses of segmentation are to find out the region of interest which is also know as **Binary mask of image.**You may wonder what is mask lets see what is mask



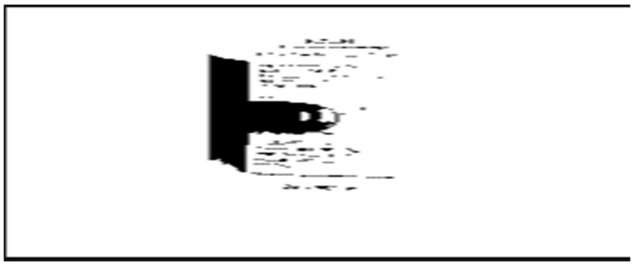
Here the image with black and whites is know as the binary mask, in this binary mark the image which we want keep is marked by "**1**" remaining all are "**0**' in that case we can remove the background filled with zeros to other background. In order to do the binary mask of image we can use the **Intensity Thresholding.**

2. Intensity Thresholding, you can create a binary black and white image from a grayscale image by thresholding its intensity values. Values below the cut-off are assigned the value 0, while those above are assigned the value 1. In the example below, a grayscale image was segmented using a threshold of 1/4 the maximum possible intensity

of 255



```
BW = imbinarize(gsAdj);
imshow(BW)
```

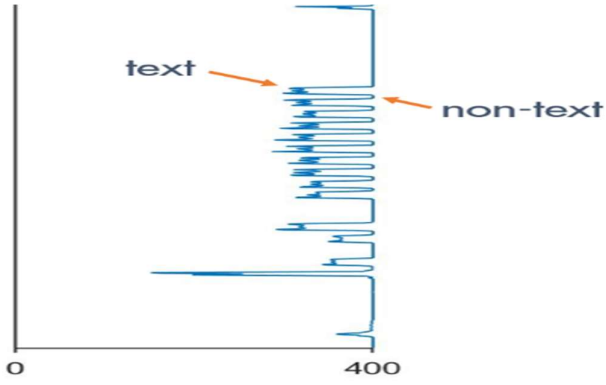


This imbinarize is used to find the automatic threshold value for each with the help of this threshold , this will autmatically in increase the brightness of the images as per needed.

Identifying Text Patterns

Now, You've improved the visibility of the text in a binary image. Now how can you determine if the image is a receipt? The answer lies in the pattern of text in the image. Rows of pixels containing text have more 0 values, and the rows between lines of text have more 1s. If you sum the values across each row, rows with text will have smaller sums than rows without text.

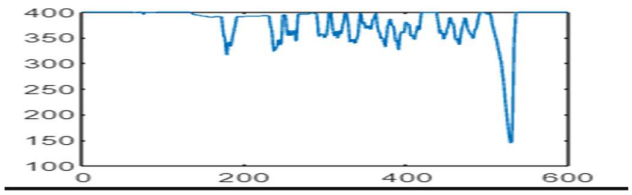
binary image



row sum

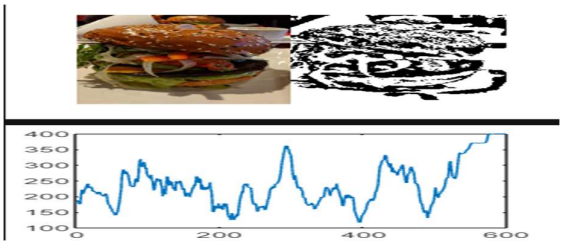
The code for this is to find the sum of the pixel data

```
S = sum(BW,2);
plot(S)
```



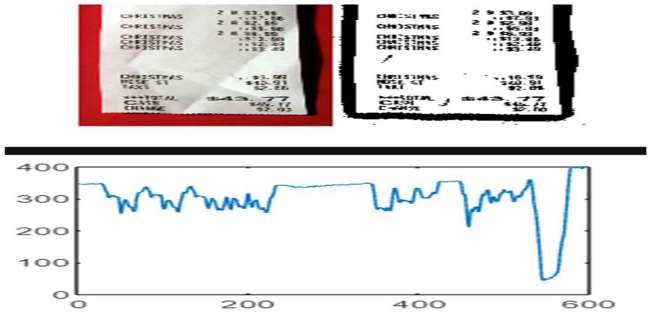
Now we will combine all the perviious code into one selection then we will compare the graphs between one receipt and non- receipt image

```
imgFile = Image 9 ;
I = imread(imgFile);
gs = im2gray(I);
gsAdj = imadjust(gs);
BW = imbinarize(gsAdj,"adaptive","ForegroundPolarity","dark");
imshowpair(I,BW,"montage")
S = sum(BW,2);
plot(S)
```



This is the example for the non- receipt image now will see the receipt image

```
imgFile = Image 3 ;
I = imread(imgFile);
gs = im2gray(I);
gsAdj = imadjust(gs);
BW = imbinarize(gsAdj,"adaptive","ForegroundPolarity","dark");
imshowpair(I,BW,"montage")
S = sum(BW,2);
plot(S)
```



When comparing this two image we can say that the receipt image have more local minima compare to non - receipt image.

Postprocessing to Improve Segmentation:

Segmentation can be improved in two ways: by preprocessing the image before binarizing and by postprocessing the binary image itself. You've already done some preprocessing by converting each image to grayscale and adjusting the contrast. In the next few activities, you'll use three additional pre- and postprocessing techniques.

Noise Removal

Smooth pixel intensity values to reduce the impact of variation on binarization.

Background Isolation and Subtraction:

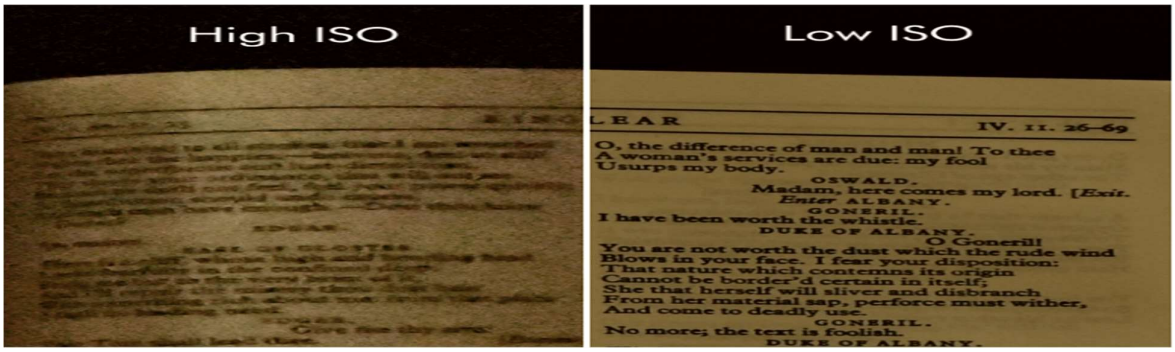
Isolate and remove the background of an image before binarizing

Binary Morphology:

Emphasize particular patterns or shapes in a binary image.

Image noise

You can increase the light sensitivity of a digital camera sensor to improve the brightness of a picture taken in low light. Many modern digital cameras (including mobile phone cameras) automatically increase the ISO in dim light. However, this increase in sensitivity amplifies noise picked up by the sensor, leaving the image grainy (shown on the left).



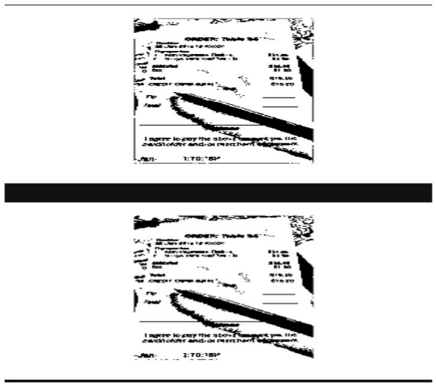
```
BWsmooth = imbinarize(gssmooth,"adaptive","ForegroundPolarity","dark");
imshow(BWsmooth)
```

Noise ClearImage

```
gssmooth = imfilter(gs,H,"replicate");
```

Without clearing noise

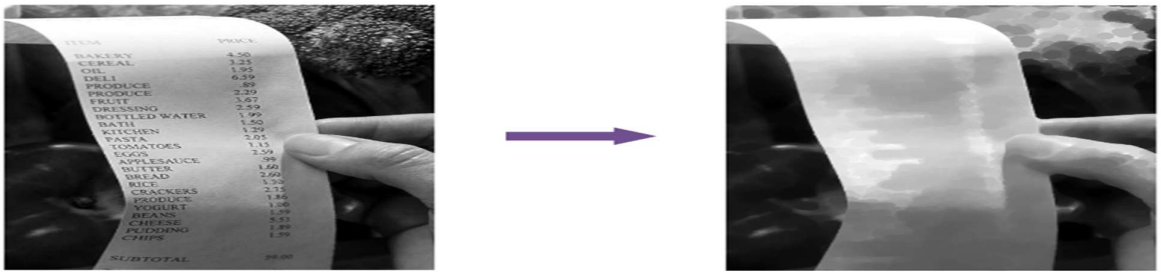
```
BWsmooth = imbinarize(gssmooth,"adaptive","ForegroundPolarity","dark");
imshow(BWsmooth)
```



This frist image is without clearninf the noise and second image is cleanring the noise

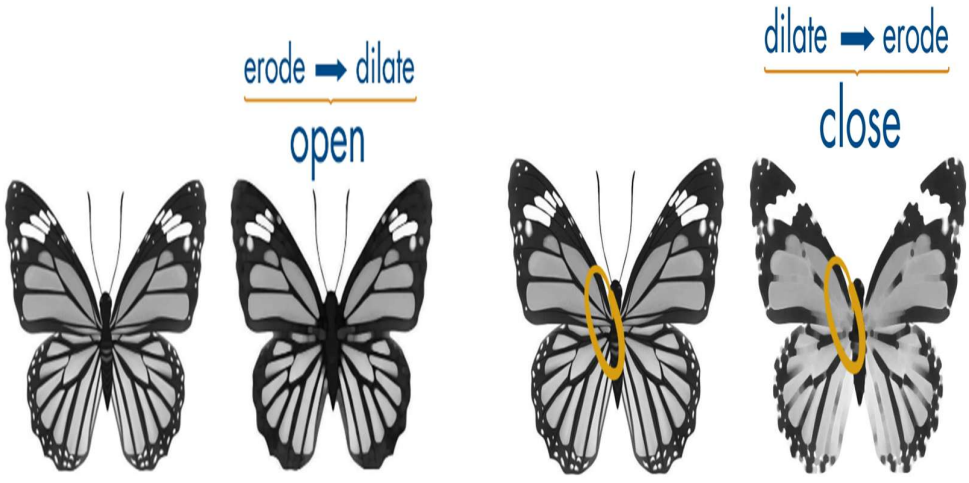
Isolating the Image Background:

Receipt images that have a busy background are more difficult to classify because artifacts pollute the row sum. To mitigate this issue, you can isolate the background and then remove it by subtraction.In a receipt image, the background is anything that is not text, so isolating the background can be interpreted as removing the text. One way to remove text from an image is to use morphological operations



A morphological closing operation emphasizes the bright paper and removes the dark text.

In the close its removes the darker part of image by and keeping the lighter intensity and **Open is** similar opposite that removes the darker part of image by and leaving the lighter intensity

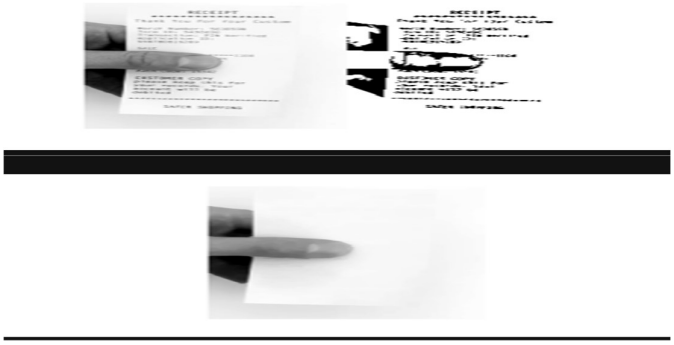


Selecting the shape

```
SE = strel("disk",8);
```

Back Ground

```
Ibg = imclose(gs,SE);
imshow(Ibg)
```

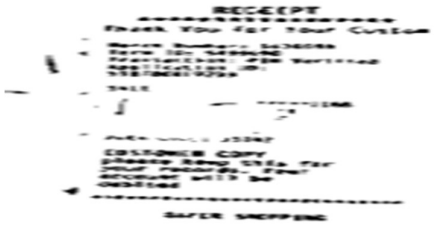


Then we will subrate the background form the orignal image

```
gsSub = Ibg- gs;
imshow(gsSub)
```

Since the image is black with white text we will re change the binary image

```
BWsub = ~imbinarize(gsSub);
imshow(BWsub)
```

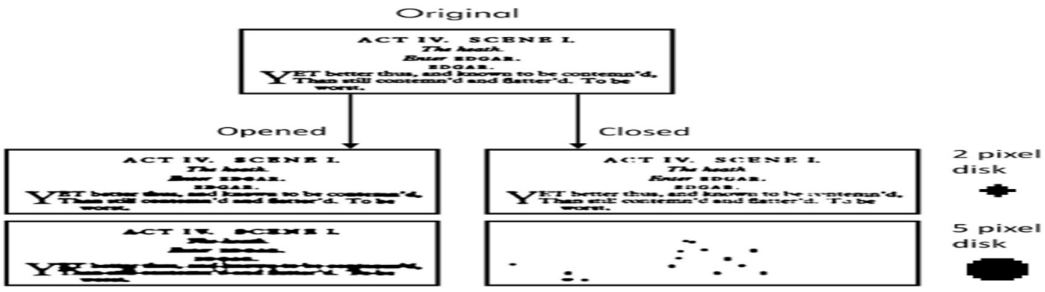


Then the backgroound is removed;

Enhancing text patterns:

The process from the previous section doesn't remove the grid in the background of this receipt. The grid pattern is too thin to be included in the background generated by the closing operation. Morphological operations are useful not only for removing features from an image but for augmenting features. You can use morphology to enhance the text in the binary image and improve the row sum signal.

Morphological opening expands the dark text regions, while closing diminishes them. Increasing the size of the structuring element increases these effects



Local Minima:

The local minimum is the input value for which the function gives the minimum output values. The function equation or the graph of the function is not sometimes sufficient to find the local minimum. The derivative of the function is very helpful in finding the local minimum of the function.

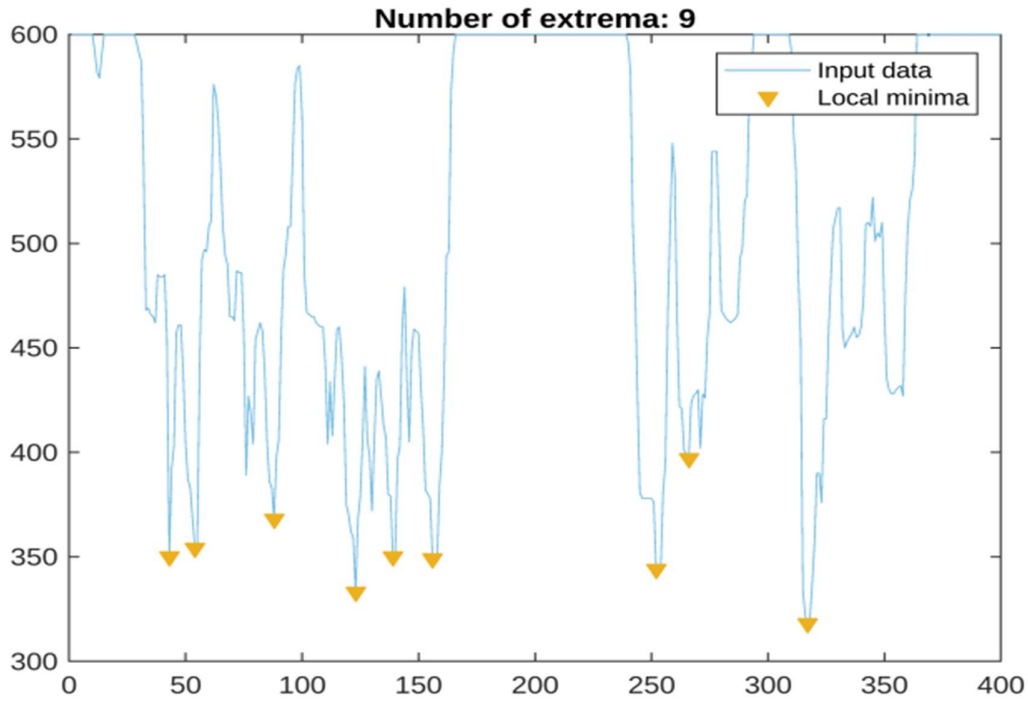
```
minIndices = islocalmin(S,'MinProminence',90,'ProminenceWindow',25);

% Display results
clf
plot(S,'Color',[109 185 226]/255,'DisplayName','Input data')
hold on

% Plot local minima
plot(find(minIndices),S(minIndices),'v','Color',[237 177 32]/255,...
    'MarkerFaceColor',[237 177 32]/255,'DisplayName','Local minima')
title(['Number of extrema: ' num2str(nnz(minIndices))])
```

1

```
hold off
legend
```



With minima we can use the classifiy as the minima valus is less than 9 then its not a receipt,if its greater than 9 then its receipts. Now you need to define a cutoff value so the algorithm can classify an image as either a receipt or non-receipt.Based on observations of the available images, 9 or more minima indicates a receipt.

So we Created a logical variable isReceipt that is 1 (true) if nMin is 9 or more and 0 (false) otherwise.

Result:

```
nMin = nnz(minIndices);
```

If logical is "1" then given image is Receipt otherwise its zero

```
isReceipt = nMin >= 9
```

```
isReceipt = logical
   1
```

Conclusion

In this projects we can undersstand that how the image is actually exits inside any storage device in the form of binary and we can undertand how each images flows certain pattern and we undertand that how the image classifications works with the help of minima we can able classify the images. The challenges that I faced during the projects is that my classification is single dimension it will detect only one type of images, instead of muliple images.2. Understanding the local minima for different images its was slight confusion. 3.Since I have used many matlab function whereves octave does nnot support sum of matlab function , I felt slight diffcuitly in using octave for my projects

Reference :

1. https://www.geeksforgeeks.org/matplotlib-pyplot-imshow-in-python/
2. https://octave.sourceforge.io/image/function/rgb2gray.html
3. https://in.mathworks.com/help/images/ref/imadjust.html
4. https://www.cuemath.com/calculus/local-minimum/
5. https://in.mathworks.com/help/images/ref/imbinarize.html
6. https://in.mathworks.com/help/images/filter-images-using-imfilter.html
7. https://in.mathworks.com/help/images/ref/imclose.html
8. https://in.mathworks.com/help/images/ref/imopen.html