ST. FRANCIS INSTITUTE OF TECHNOLOGY

Mount Poinsur, S.V.P. Road, Borivali (W), Mumbai - 400 103.

CLASS / BATCH: TE- EXTCA /TA-3 SEM: VI

NAME: Om Kadam ROLL NO: 41

SUBJECT: IPMV

EXPERIMENT NO. : 06

TITLE: To Implement adaptive and global thresholding

DATE OF PERFORMANCE:

DATE OF SUBMISSION:

Correction Parameters	Marks Allotted	Correction Parameters	Marks Allotted
Submission on time [10%]		Experimental Result [30%]	
Conclusion / Result Analysis/ Discussion [30%]		Post Experiment Exercise [20%]	
Presentation of Write-Up [10%]			

TOTAL MARKS:

FACULTY SIGNATURE (WITH DATE)

Roll No._____

Experiment – 6: To implement Global and Adaptive Thresholding

Date: _____

1. **Aim : Write a program to implement Global and adaptive Thresholding**

2. **Requirements:** Python :

3. **Pre-Experiment Exercise**

3.1 Brief Theory

As the fact that we need only the histogram of the image to segment it, segmenting images with Threshold Technique does not involve the spatial information of the images.

Basic Global Thresholding: As the fact that we need only the histogram of the image to segment it, segmenting images with Threshold Technique does not involve the spatial information of the images. Therefore, some problem may be caused by noise, blurred edges, or outlier in the image. That is why we say this method is the simplest concept to segment images.

When the intensity distributions of objects and background pixels are sufficiently distinct, it is possible to use a single (global) threshold applicable over the entire image.

The following iterative algorithm can be used for this purpose:

1. Select an initial estimate for the global threshold, T.
2. Segment the image using T as

$$g(x,y)=\begin{cases}1 & \text{if f(x,y)}\geq T\\ 0 & \text{if f(x,y)}\leq T\end{cases}$$

This will produce two groups of pixels: G_1 consisting of all pixels with intensity values $> T$, and G_2 consisting of pixels with values $\leq T$.

3. Compute the average intensity values m_1 and m_2 for the pixels in G_1 and G_2.
4. Compute a new threshold values:

$$T=\frac{1}{2}(m_1+m_2)$$

5. Repeats Step2 through 4 until the difference between values of T in successive iterations is smaller than a predefined parameter.

Basic Adaptive thresholding technique: Images having uneven illumination make it difficult to segment using the histogram. In this case we have to divide the image in many sub images and then come up with different threshold to segment each sub image. The key issues are how to divide the image into sub images and utilize a different threshold to segment each sub image. The major drawback to threshold-based approaches is that they often lack the sensitivity and specificity needed for

accurate classification. Fig. 1 shows the comparison between adaptive and global thresholding and adaptive thresholding. Since the threshold for each pixel depends on its location within an image this technique is said to adaptive.

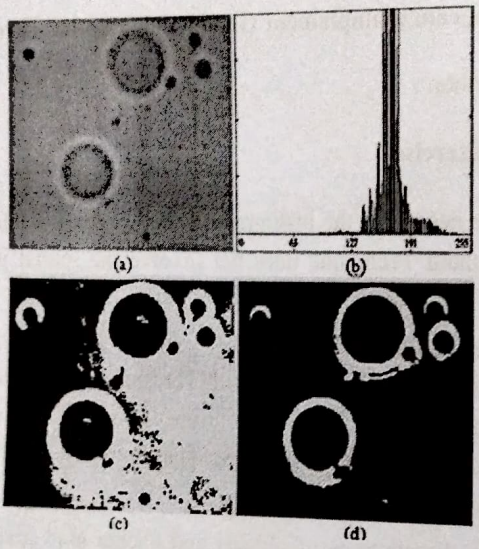


Fig. 1:. a) Original Image, b) Histogram of original image.
c) Global Thresholding d) Adaptive Thresholding

4. Laboratory Exercise

4.1. Algorithm :

1. Read the grayscale image and find the histogram.
2. Select initial estimate threshold T from histogram
3. Segment the image using T as a threshold as per the above steps of basic global thresholding algorithm and display the output binary image.
4. For basic adaptive thresholding, divide image into four sub-images apply global thresholding in each sub-image.
5. Connect all segmented sub-images according to their initial position to get full binary image, display final image.

5. *Post Experiment*

5.1 Conclusion

Global & Adaptive thresholding are two commonly used techniques for image segmentation. Both techniques have their strengths and weaknesses.

5.2 Questions

i. What is the need of adaptive thresholding?

ii. What is the role of illumination in image segmentation?

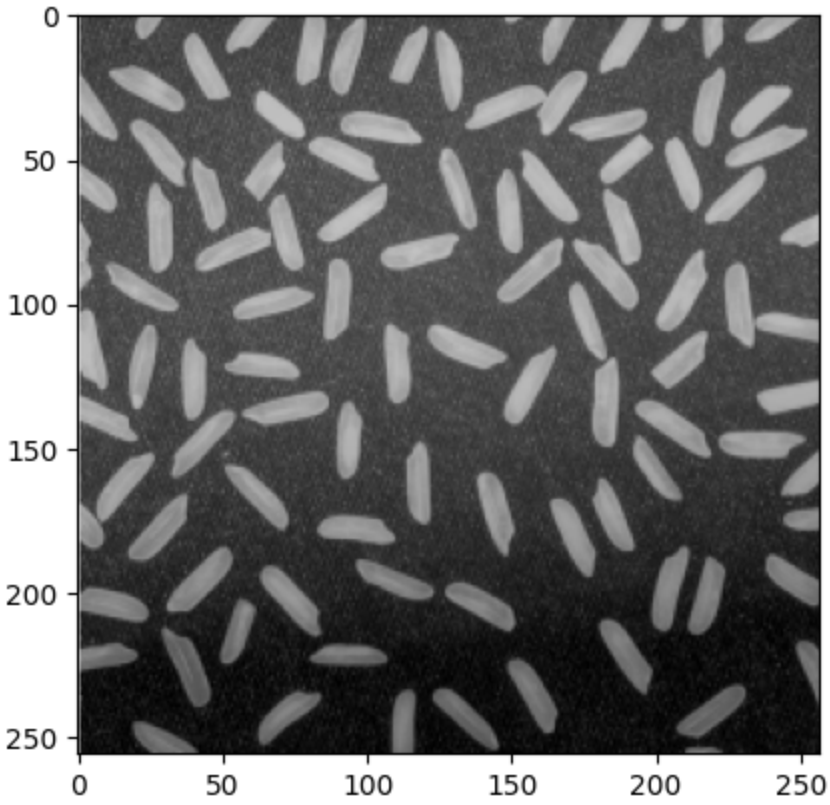
1) Adaptive thresholding is a technique used in image processing to convert a grayscale image it to a binary image. It is useful when the image has varying illumination conditions, as it adjusts the threshold value dynamically for different regions of the image. This improves the segmentation of the image, making it easier to extract useful information.

2) Illumination plays a crucial role in image segmentation. It can affect the contrast and brightness of an image, making it harder to distinguish between different regions or objects. A well-lit image can enhance the differences in intensity levels, allowing for more accurate segmentation.

In [ ]:
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from io import BytesIO
from PIL import Image
from google.colab.patches import cv2_imshow
from google.colab import files

# uploaded = files.upload()
```

In [ ]:
```
image = cv2.imread('rice.png', cv2.IMREAD_GRAYSCALE)
plt.imshow(image, cmap=plt.cm.gray)
```

Out[ ]: <matplotlib.image.AxesImage at 0x7b247a6b75e0>



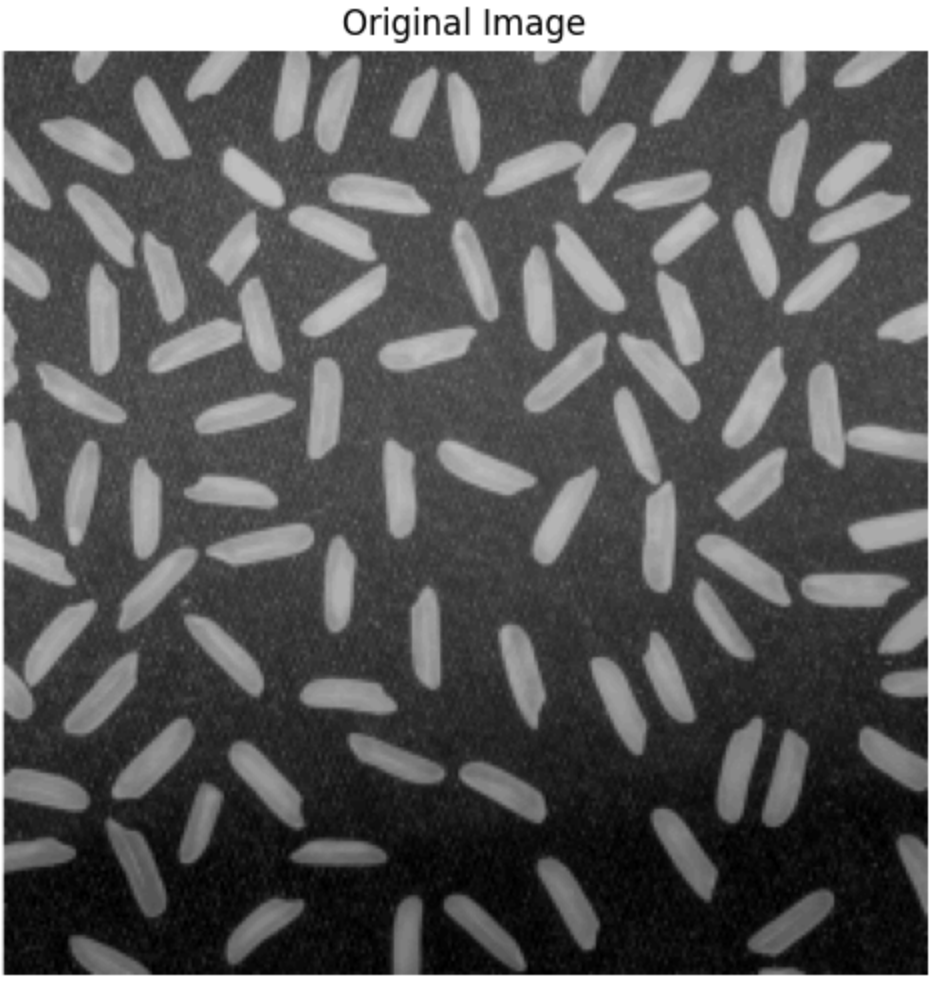
In [ ]:
```
def basic_adaptive_thresholding(image):
    # Compute histogram
    hist = cv2.calcHist([image], [0], None, [256], [0, 256])

    # Select initial estimate threshold T from histogram
    T = np.argmax(hist)

    # Segment the image using T as the threshold
    _, binary_image = cv2.threshold(image, T, 255, cv2.THRESH_BINARY)

    # Divide image into four sub-images
    rows, cols = image.shape
    sub_images = [image[:rows//2, :cols//2],
                  image[:rows//2, cols//2:],
```

```
                image[rows//2:, :cols//2],
                image[rows//2:, cols//2:]]

    # Apply global thresholding independently to each sub-image
    binary_sub_images = [cv2.threshold(sub_image, T, 255, cv2.THRESH_BINARY)[1] for

    # Combine segmented sub-images
    segmented_image = np.vstack((np.hstack((binary_sub_images[0], binary_sub_images
                                np.hstack((binary_sub_images[2], binary_sub_images

    # Connect segmented sub-images to get the full binary image
    full_binary_image = np.zeros_like(image)
    full_binary_image[:rows//2, :cols//2] = binary_sub_images[0]
    full_binary_image[:rows//2, cols//2:] = binary_sub_images[1]
    full_binary_image[rows//2:, :cols//2] = binary_sub_images[2]
    full_binary_image[rows//2:, cols//2:] = binary_sub_images[3]

    return T, binary_image, binary_sub_images, segmented_image, full_binary_image,
```

In [ ]:
```
# Perform basic adaptive thresholding
T, binary_image, binary_sub_images, segmented_image, full_binary_image, hist = basi
```

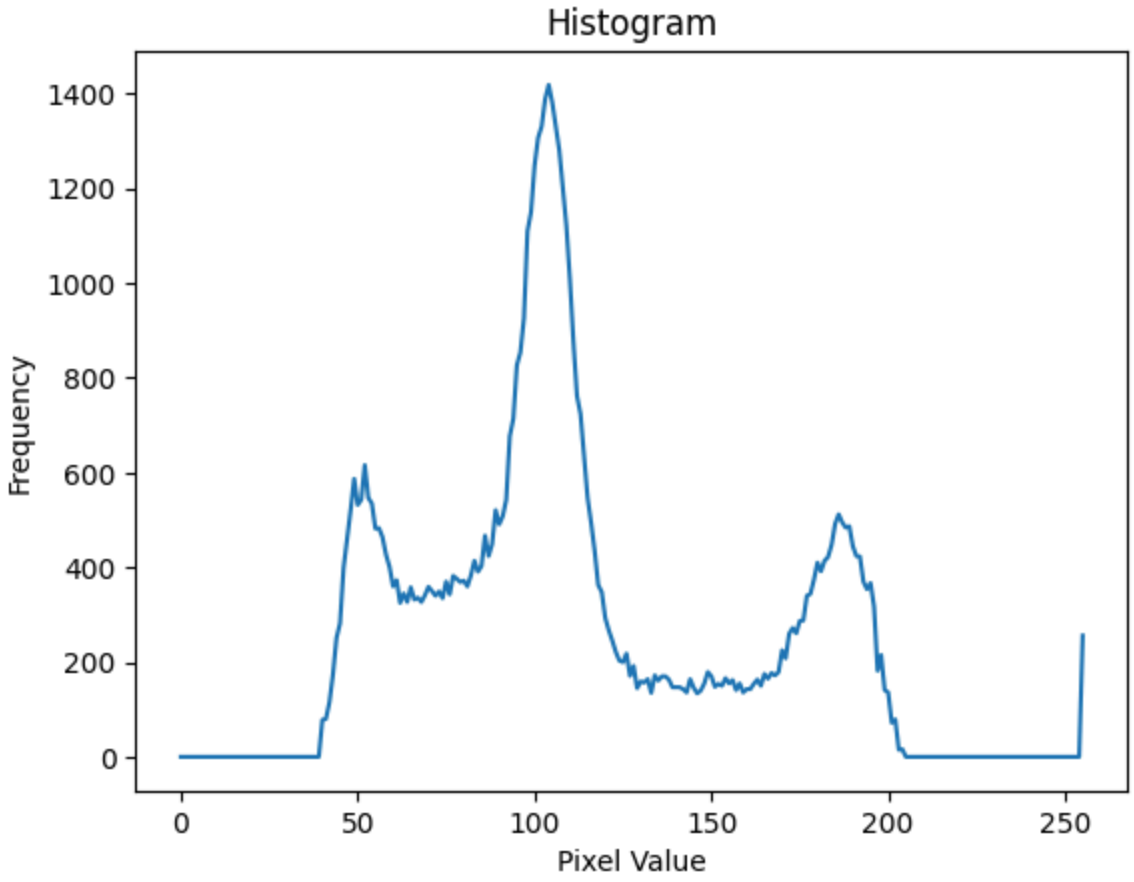
In [ ]:
```
# Display the original image
plt.figure(figsize=(8, 6))
plt.imshow(image, cmap='gray')
plt.title('Original Image')
plt.axis('off')
```

Out[ ]: (-0.5, 256.5, 255.5, -0.5)



In [ ]:
```
# Display the histogram
plt.plot(hist)
plt.title('Histogram')
plt.xlabel('Pixel Value')
plt.ylabel('Frequency')
```

Out[ ]:
```
Text(0, 0.5, 'Frequency')
```

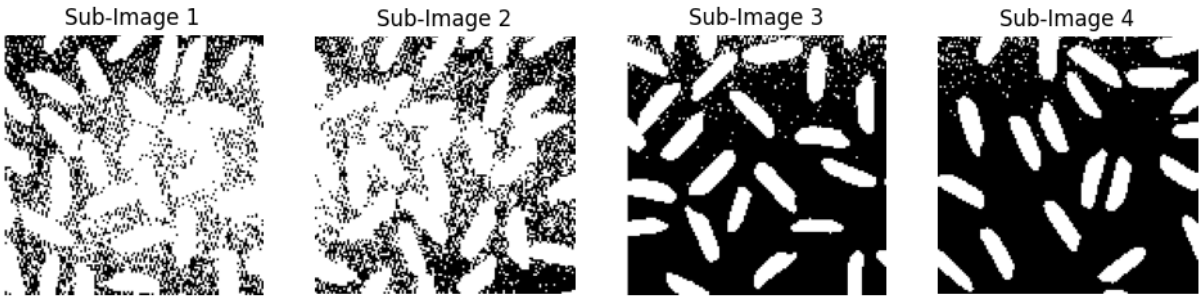


In [ ]:
```
# Display the binary image
plt.imshow(binary_image, cmap='gray')
plt.title('Binary Image (Threshold = {})'.format(T))
plt.axis('off')
```

Out[ ]: (-0.5, 256.5, 255.5, -0.5)

In [ ]:
```
# Display the sub-images after thresholding
fig, ax = plt.subplots(figsize=(12, 8))
for i in range(4):
    ax = plt.subplot(1, 4, i+1)
    ax.imshow(binary_sub_images[i], cmap='gray')
    ax.set_title('Sub-Image {}'.format(i+1))
    ax.axis('off')
```

```
<ipython-input-8-22808f59bf9e>:4: MatplotlibDeprecationWarning: Auto-removal of over
lapping axes is deprecated since 3.6 and will be removed two minor releases later; e
xplicitly call ax.remove() as needed.
  ax = plt.subplot(1, 4, i+1)
```



In [ ]:
```
# Display the segmented image
plt.imshow(segmented_image, cmap='gray')
plt.title('Segmented Image')
plt.axis('off')
```

Out[ ]:
```
(-0.5, 256.5, 255.5, -0.5)
```

In [ ]:
```
# Display the full binary image
plt.imshow(full_binary_image, cmap='gray')
plt.title('Full Binary Image')
plt.axis('off')

plt.tight_layout()
plt.show()
```

In [ ]: