CLASS : B.E. E &TC **SUBJECT: DIVP**

EXPT. NO. : 8 **DATE: 06/11/2020**

TITLE : TO PERFORM GLOBAL THRESHOLDING

CO 2:	Given a gray image, select an appropriate technique (similarity based or discontinuity based) to segment it. Derive the mask coefficients of First order Derivative (FoD) and Second order Derivative (SoD) to detect an edge in an image. Considering an appropriate test case, analyze and compare the performance of FoD and SoD using parameters like response to constant intensity and isolated intensities in an image.
CO4:	Carry out experiments as an individual and in a team, comprehend and write a laboratory record and draw conclusions at a technical level.

AIM: To implement Global Thresholding

SOFTWARES REQUIRED: Matlab 7.0 or above

THEORY:

8.1 Thresholding:

Thresholding is one of the most important approaches to image segmentation.

It is the last step in image segmentation. Different types of thresholding are,

1. Global (depends on intensity value)
2. Adaptive (depends on intensity and local property)
3. Optimal (depends on intensity, position and local property)
4. Local Thresholding function is,

T = [f(x, y), p(x, y), (x, y)]

where f(x, y) = intensity of pixel

p(x, y) = local property (neighborhood)

(x, y) = position of pixel

- When T depends only on f(x, y) → global threshold
- When T depends on both f(x, y) and p(x, y) → local threshold

8.2 Segmentation by thresholding

Thresholding is the simplest segmentation method.

The pixels are partitioned depending on their intensity value I. Global thresholding, using an appropriate threshold T:

g(x, y) =1, if f (x, y) > T

=0, if f (x, y) _ T

Variable thresholding, if T can change over the image, Local or regional thresholding, if T depends on a neighborhood of (x, y). I adaptive thresholding, if T is a function of (x, y).

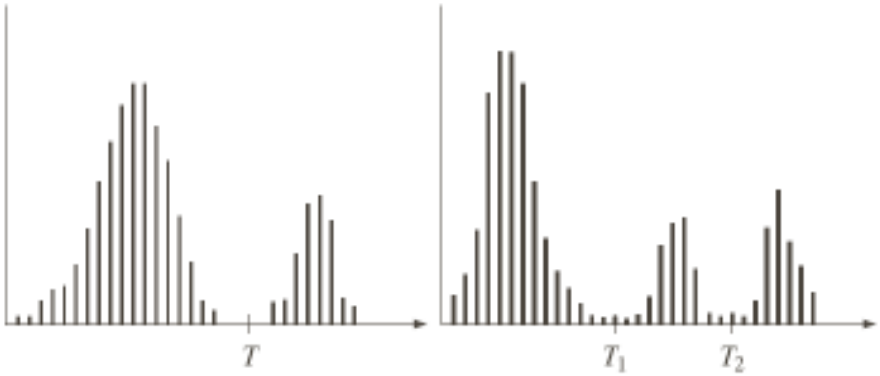
Multiple thresholding:

g(x, y) =a, if f (x, y) > T_2

=b, if T_1 < f (x, y) <T_2

=c, if f (x, y) <T_1

8.3 Choosing the thresholds



Peaks and valleys of the image histogram can help in choosing the appropriate value for the threshold(s). Some factors affect the suitability of the histogram for guiding the choice of the threshold:

- The separation between peaks;
- The noise content in the image;
- The relative size of objects and background;
- The uniformity of the illumination;
- The uniformity of the reflectance.

8.4 Global thresholding:

In practice global thresholding can be expected to be more successful in highly controlled environments such as industrial inspection application, where illumination control is feasible. Usually a successful segmentation is highly depends on the choice of thresholds.

Steps for Global Thresholding:

1) Initial estimate of T
2) Segmentation using T:
    - G_1, pixels brighter than T;
    - G_2, pixels darker than (or equal to) T.

3) Computation of the average intensities m_1 and m_2 of G_1 and G_2.

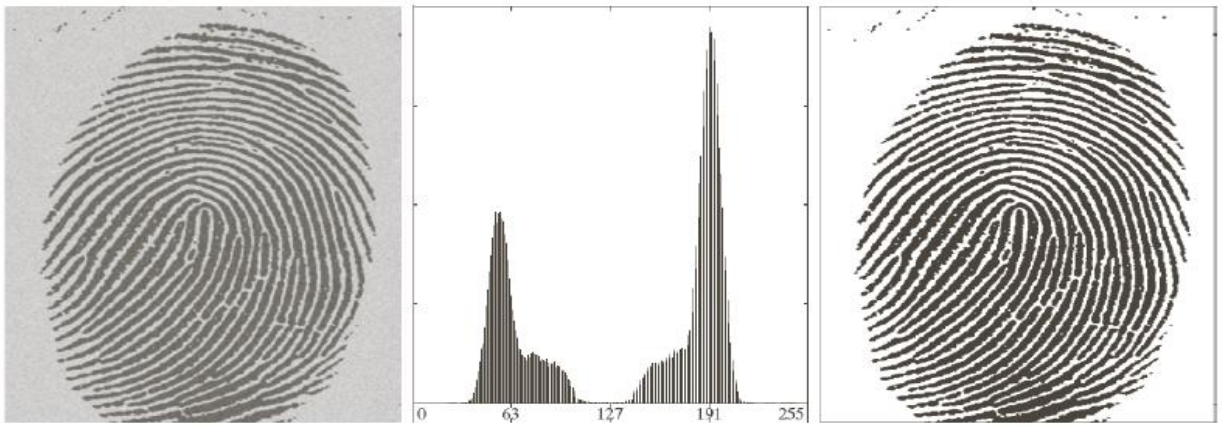
4) New threshold value:

$$T_{new} = (m_1 + m_2)/ 2$$

5) If $|T - T_{new}| >=$ delta (T), back to step 2, otherwise stop.

This simple steps works well in situations where there is a reasonably clear valley between the modes of the histogram related to objects and background. Parameter delta(T) is used to control the number of iterations in situations where speed is important. This initial threshold must be chosen greater than the minimum and less than maximum intensity level in the image.

Global thresholding : An example



8.5 Algorithm:

1) Read the image.

2) Input the threshold value and delta (T)(Delta T is error which can be tolerated).

3) Initial estimate of T

4) Segmentation using T:

- G_1, pixels brighter than T;
- G_2, pixels darker than (or equal to) T.

5) Computation of the average intensities m_1 and m_2 of G_1 and G_2.

6) New threshold value:

$$T_{new} = (m_1 + m_2)/ 2$$

7) If $|T - T_{new}| >= delta(T)$, back to step 4, otherwise stop. (T is of previous iteration)

8) Assign '0' for pixel values less than newly obtained threshold value of original image and '255' for the pixel value greater than threshold.

8.6 Conclusion:

I In this experiment I applied global and adaptive thresholding technique on a grey image to it to get a binary image. I conclude that using thresholding techniques makes the analysis of a gray image easier as it leads to rounding up unwanted pixel intensities and highlights the desired range. It can also be said that it is best suited for images with separate or nearly separated clusters of pixel intensities.

8.7 References:

1." Digital Image Processing " , by Gonzalez and Woods.

2. "Digital Image Processing " , S. Jayaraman, S. Esakkirajan, T. Veerakumar.

3. Pictures taken from:

http://www.imageprocessingplace.com/root_files_V3/image_databases.html

(Course Teacher)

CLASS	**:**	**B.E (E &TC)**	**COURSE**	**:**	**DIVP**
AY	**:**	**2020-21 (SEM- I)**	**DATE**	**:**	**06/11/2020**
EXPT. NO.	**:**	**8**	**CLASS & ROLL NO**	**:**	**BE VIII 42410**
TITLE	**:**	**TO PERFORM GLOBAL THRESHOLDING**			

I. CODE:

1. Global Thresholding

```
# -*- coding: utf-8 -*-
import cv2
import matplotlib.pyplot as plt
import numpy as np
path="C:/Users/Danish/Desktop/DIVP/Images/eye2.jpg"
img = cv2.imread(path,0)
plt.hist(img.ravel(),256,[0,256])
plt.show()
ret, thresh1 = cv2.threshold(img, 250, 255, cv2.THRESH_BINARY)
ret, thresh2 = cv2.threshold(img, 250, 255, cv2.THRESH_BINARY_INV)
row, column = img.shape
img1 = np.zeros((row,column))
for i in range(row):
    for j in range(column):
        if img[i, j] >= 250:
            img1[i, j] = 255
        else:
            img1[i,j] = 0
cv2.imshow('Input Image', img)
cv2.imshow('Inbuilt Binary Thresholded', thresh1)
cv2.imshow('Binary Thresholded', img1)
cv2.imshow('Binary Threshold Inverted', thresh2)
cv2.waitKey(30000)
cv2.destroyAllWindows()
```

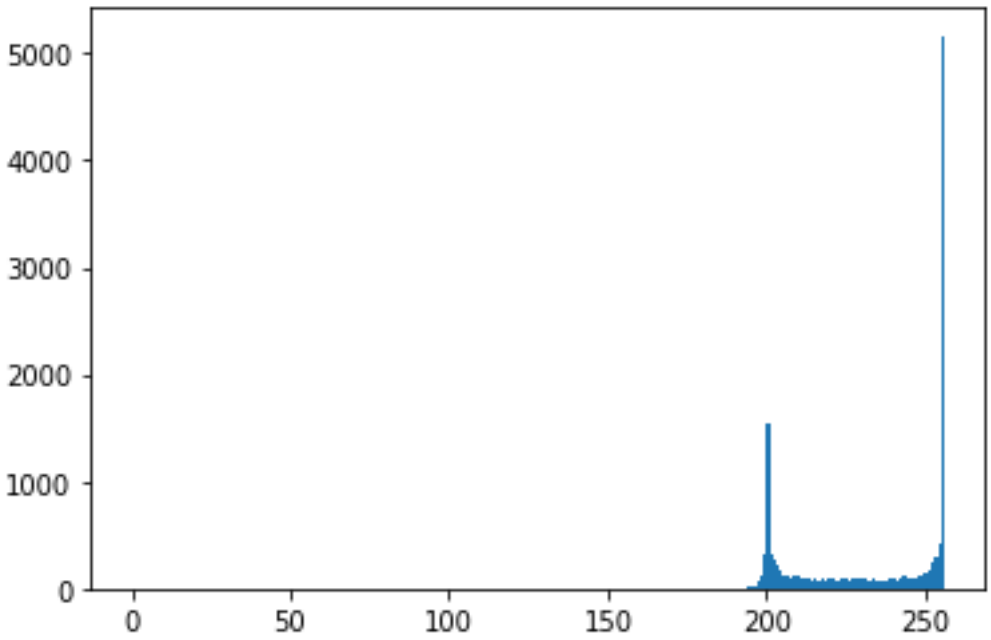
2. Adaptive Thresholding

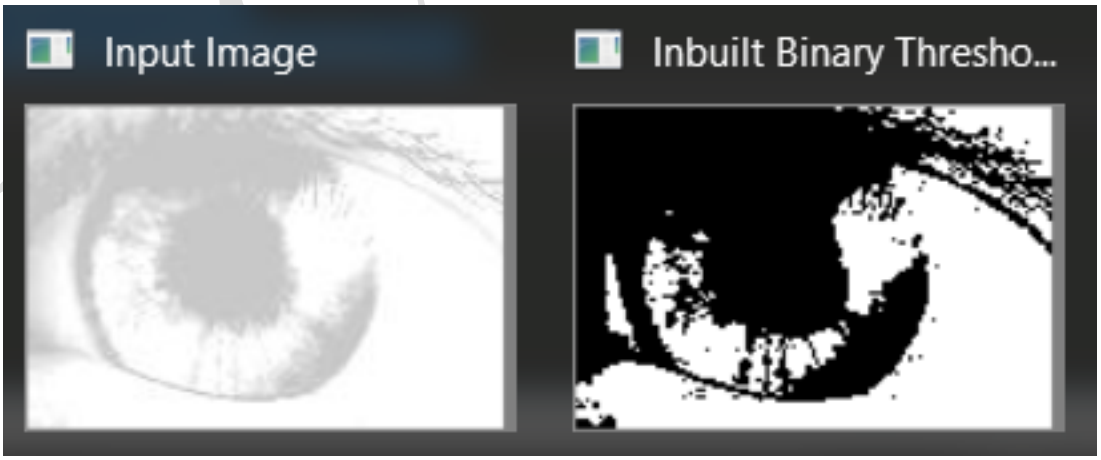
```
# -*- coding: utf-8 -*-
import cv2
import matplotlib.pyplot as plt
import numpy as np
path="C:/Users/Danish/Desktop/DIVP/Images/eye2.jpg"
img = cv2.imread(path,0)

thresh1 = cv2.adaptiveThreshold(img, 255,
cv2.ADAPTIVE_THRESH_MEAN_C,cv2.THRESH_BINARY, 19, 3)
```

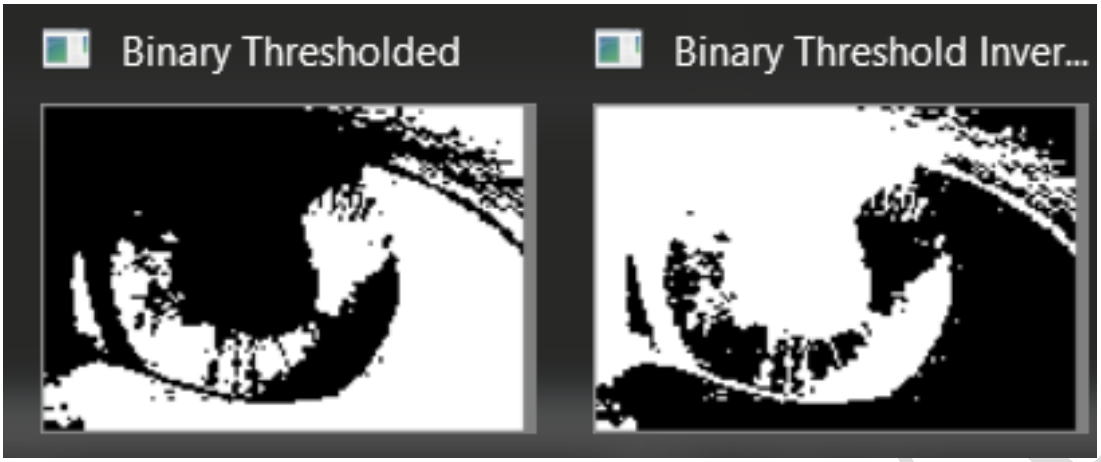
```
thresh2 = cv2.adaptiveThreshold(img, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C,cv2.THRESH_BINARY, 19, 3)
cv2.imshow('Adaptive Mean', thresh1)
cv2.imshow('Adaptive Gaussian', thresh2)
cv2.waitKey(30000)
cv2.destroyAllWindows()
```

II. RESULTS:

1. Global Thresholding







2. Adaptive Thresholding