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

EXPT. NO. : 7 **DATE:06-11-2020**

TITLE : TO PERFORM EDGE DETECTION USING VARIOUS MASK

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:

1. Study of Image segmentation techniques
2. Implement edge detection using Gradient Operators using MATLAB
   a. Robert's operator
   b. Prewitt operator
   c. Sobel operator

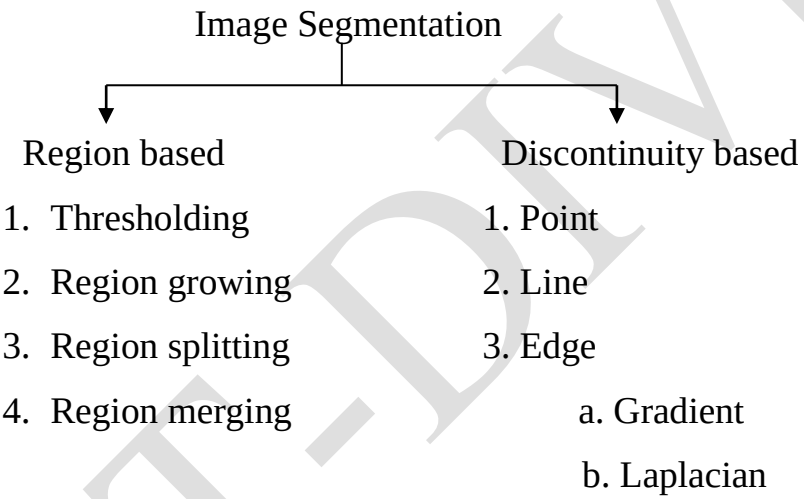
SOFTWARE REQUIREMENT: Matlab 7.0 or above or Python

THEORY:

7.1 Image Segmentation

The first step in image analysis is to segment the image.

Segmentation is the process to subdivide the image into its constituent parts or objects. The level to which subdivision is carried depends on the problem being solved.



Segmentation algorithms for monochrome images are generally based on one of the two basic properties of grey level values:

1. Discontinuity :

   In this category the approach is to partition the image based on abrupt changes in grey level.

2. Similarity :

   In this category the approaches are based in thresholding, region growing, region splitting and merging.

7.2 Detection of Discontinuities:

In this section, we present several techniques for detecting three basic types of discontinuities in digital image :

1. Point 2. Line 3. Edge

7.2.1 Point Detection :

The detection of isolated points in image is straight forward. Using the mask shown below, we can say that a point has been detected at the location on which mask is centered if

$|R| > T$ where T is non negative threshold and

R is computed as sum of products of coefficients of mask with the grey level values of image encompassed by the mask.

-1	-1	-1
-1	8	-1
-1	-1	-1

7.2.2 Line Detection:

If the first mask were moved around an image, it would respond more strongly to lines (one pixel thick) oriented horizontally. A similar experiment would reveal that

The second mask responds best to the lines oriented at 45^0 ; the third mask to vertical lines; and the fourth mask to line in the -45^0 direction.

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

+45

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

2	-1	-1
-1	2	-1
-1	-1	2

-45^0

7.2.3 Edge Detection :

Edge detection is by far the most common approach for detecting meaningful discontinuities in gray level. The reason is that isolated points and the thin lines are not frequent occurrences in most practical applications. An edge is the boundary between two regions with relative distinct gray level properties. Note from the profile that an edge (transition from dark to light) is modeled as a smooth, rather than as an abrupt, change of gray level. This model reflects the fact that edges in digital images are generally slightly blurred as a result of sampling.

7.3 Gradient Operator:

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

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}.$$

Gradient vector always points in the direction of maximum rate of change of f at co-ordinates (x,y). Magnitude of the vector is the main quantity in edge detection and given by,

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2}.$$

Different Gradient Operators:

Use of different gradient operators for the following 3x3 grey level image is discussed below.

Z1	Z2	Z3
Z4	Z5	Z6
Z8	Z8	Z9

3x3 Grey Level Image

1. Roberts Operator:

One of the simplest way to implement first order partial derivative is to use Roberts cross-gradient operator.

The mask for Roberts operator are given by,

-1	0
0	-1

0	-1
1	0

The two cross differences for x and y gradient components are given by equation,

Gx = Z9 - Z5 and Gy = Z6 - Z8.

Then we can implement first order derivative ▼f at point Z5.

Mask of size 2x2 is awkward to implement because they don't have clear center. So generally 3x3 masks are used.

2. Prewitt Operator :

In this formulation, the difference between first and third row of 3x3 image approximates the derivative in x direction and difference between first and third column of 3x3 image approximates the derivative in y direction.

The mask for Prewitt operator are shown below,

-1	-1	-1
0	0	0
1	1	1

-1	0	1
-1	0	1
-1	0	1

Thus 1st order derivative at point Z5 with Prewitt operator is

Gx = (Z8+Z8+Z9)-(Z1+Z2+Z3)

Gy = (Z3+Z6+Z9)-(Z1+Z4+Z8).

Prewitt is mostly used for computing digital gradient and is simple to implement than sobel mask.

3. Sobel Operator :

Sobel operator has advantage of providing both of differencing and smoothing effect. Since derivative enhanced noise, the smoothing effect is the most attractive feature of this operator.

Mask of Sobel Operator are,

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

The 1st order derivative at point Z5 using Sobel operator is given by,

Gx = (Z8+2Z8+Z9)-(Z1+2Z2+Z3)

Gy = (Z3+2Z6+Z9)-(Z1+2Z4+Z8).

7.4 Algorithms for Edge Detection:

1 Start.
2 Read the image in the matrix form.
3 Subdivide the image into blocks of 3*3(sobel and prewitt) and 2*2(robertz).
4 Get choice from user to use Prewitt, Sobel or Roberts edge detector.
5 Multiply the mask in x direction with the subdivided image and calculate Gx.
6 Multiply the mask in y direction with the subdivided image and calculate Gy.
7 Calculate ▼f = mag (▼f) = $|Gx| + |Gy|$ and compare the value with threshold.
8 Assign the value to centre pixel based on comparison result.
9 Continue this procedure till the last pixel.
10 Display the new image.
11 End.

7.5 Conclusion:

In this experiment edge detection is performed using following operators:

- Laplacian: It uses laplacian operator and is having fixed characteristics in all directions. It is always used with zero crossing to find edges correctly.
- LoG: here we apply Gaussian filter first before applying laplacian filter, hence it is more efficient in finding correct places of edges. It fails to find orientation of edges.
- DoG: here we apply Gaussian filter with variable kernel size and take difference of those two results, hence it gives accurate results.
- Sobel , Prewitt: It is simpler to use, highly sensitive to noise.
- Canny: It uses Gaussian operator for approximation, hence more immune to noise. It is time consuming process.
- Roberts: It is simpler to use and it uses detects cross edges.

7.6 References:

i. Gonzalez R, Woods R, “Digital image processing”, Pearson Prentice Hall, 2008.

ii. Gonzalez R, Woods R, Steven E, “Digital Image Processing Using MATLAB®”, McGraw Hill Education, 2010.

iii. Jayaraman S, Esakkirajan S and Veerakumar T, “Digital Image Processing” Tata McGraw Hill, 2010

iv. Joshi, Madhuri A. “Digital Image Processing: an algorithm approach”, PHI Learning Pvt. Ltd., 2006.

v. Pictures taken from:
http://www.imageprocessingplace.com/root_files_V3/image_databases.html

(Course Teacher)

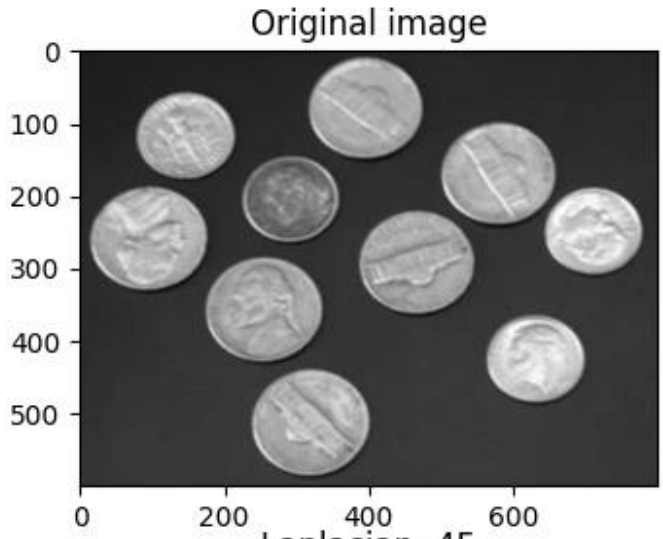
CLASS	**:**	**B.E (E &TC)**	**COURSE**	**:**	**DIVP**
AY	**:**	**2020-21 (SEM- I)**	**DATE**	**:**	**06-11-2020**
EXPT. NO.	**:**	**7**	**CLASS & ROLL NO**	**:**	**BE VIII 42428**
TITLE	**:**	**TO PERFORM EDGE DETECTION USING VARIOUS MASKS**			

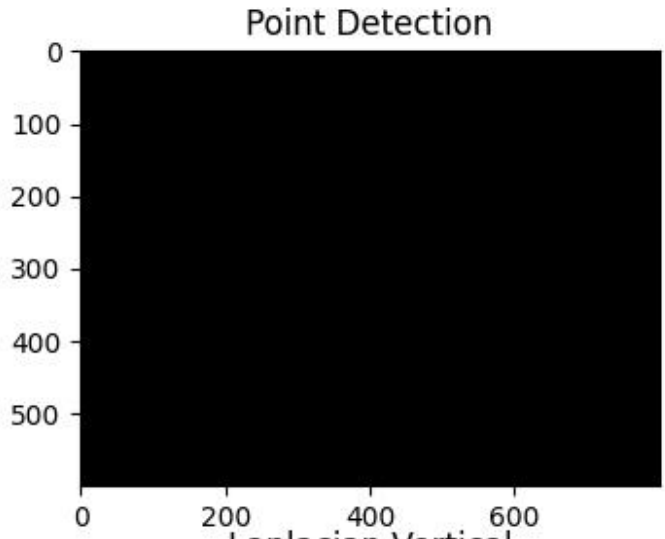
I. CODE:

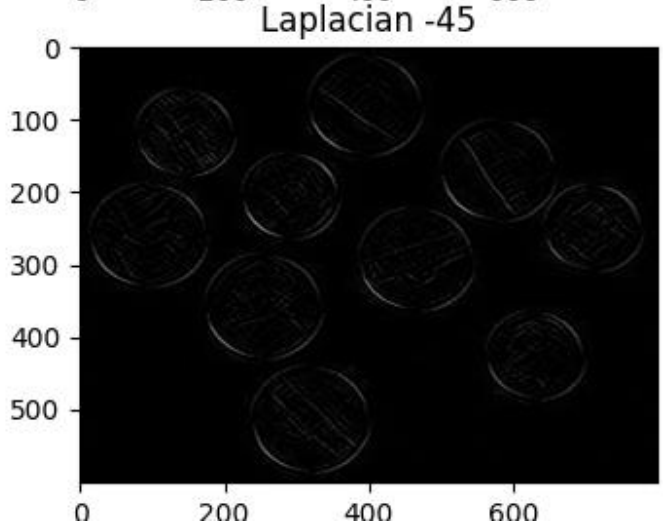
```
import cv2
import numpy as np
import matplotlib.pyplot as plt

image = cv2.imread('Images/laplacian/coin.jpg',0)
image = cv2.resize(image,(800,600))

 # Robertsedge operator
kernel_Roberts_x = np.array([
    [1, 0],
    [0, -1]
    ])
kernel_Roberts_y = np.array([
    [0, -1],
    [1, 0]
    ])
 # Sobel edge operator
kernel_Sobel_x = np.array([
    [-1, 0, 1],
    [-2, 0, 2],
    [-1, 0, 1]])
kernel_Sobel_y = np.array([
    [1, 2, 1],
    [0, 0, 0],
    [-1, -2, -1]])
 # Prewitt edge operator
kernel_Prewitt_x = np.array([
    [-1, 0, 1],
    [-1, 0, 1],
    [-1, 0, 1]])
kernel_Prewitt_y = np.array([
    [1, 1, 1],
    [0, 0, 0],
    [-1, -1, -1]])

kernel_Laplacian_point = np.array([
    [0, 1, 0],
    [1, -8, 1],
    [0, 1, 0]])
kernel_Laplacian_horizontal = np.array([
    [-1,-1,-1],
    [2, 2, 2],
    [-1,-1,-1]])
```

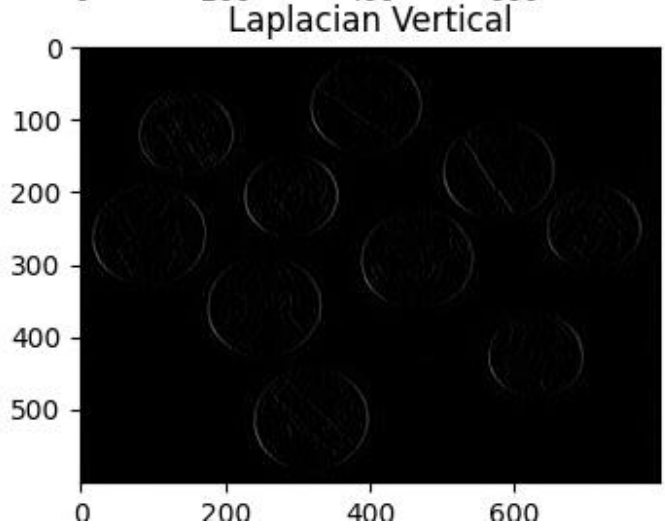
```
kernel_Laplacian_45 = np.array([
   [-1, -1, 2],
   [-1, 2, -1],
   [2, -1, -1]])
kernel_Laplacian_neg_45 = np.array([
   [2, -1, -1],
   [-1, 2, -1],
   [-1, -1, 2]])
kernel_Laplacian_vertical = np.array([
   [-1, 2, -1],
   [-1, 2, -1],
   [-1, 2, -1]])

 # convolution

output_laplacian_point = cv2.filter2D(image, -1, kernel_Laplacian_point)
output_laplacian_horizontal = cv2.filter2D(image, -1, kernel_Laplacian_horizontal)
output_laplacian_45 = cv2.filter2D(image, -1, kernel_Laplacian_45)
output_laplacian_neg_45 = cv2.filter2D(image, -1, kernel_Laplacian_neg_45)
output_laplacian_vertical = cv2.filter2D(image, -1, kernel_Laplacian_vertical)
output_Roberts_x = cv2.filter2D(image, -1, kernel_Roberts_x)
output_Roberts_y = cv2.filter2D(image, -1, kernel_Roberts_y)
output_prewitt_x = cv2.filter2D(image, -1, kernel_Prewitt_x)
output_prewitt_y = cv2.filter2D(image, -1, kernel_Prewitt_y)
output_sobel_x = cv2.filter2D(image, -1, kernel_Sobel_x)
output_sobel_y = cv2.filter2D(image, -1, kernel_Sobel_y)

output_Roberts = output_Roberts_x + output_Roberts_y
output_prewitt = output_prewitt_x + output_prewitt_y
output_sobel = output_sobel_x + output_sobel_y

plt.subplot(3, 3, 1), plt.imshow(image, 'gray'),plt.title('Original image')
plt.subplot(3, 3, 2), plt.imshow(output_laplacian_point, 'gray'),plt.title('Point Detection')
plt.subplot(3, 3, 3), plt.imshow(output_laplacian_horizontal, 'gray'),plt.title('Laplacian Horizaonta
l')
plt.subplot(3, 3, 4), plt.imshow(output_laplacian_45, 'gray'),plt.title('Laplacian +45')
plt.subplot(3, 3, 5), plt.imshow(output_laplacian_neg_45, 'gray'),plt.title('Laplacian -45')
plt.subplot(3, 3, 6), plt.imshow(output_laplacian_vertical, 'gray'),plt.title('Laplacian Vertical')
plt.subplot(3, 3, 7), plt.imshow(output_Roberts, 'gray'),plt.title('Roberts operator')
plt.subplot(3, 3, 8), plt.imshow(output_prewitt, 'gray'),plt.title('Prewitt operator')
plt.subplot(3, 3, 9), plt.imshow(output_sobel, 'gray'),plt.title('Sobel operator')
plt.show()
```

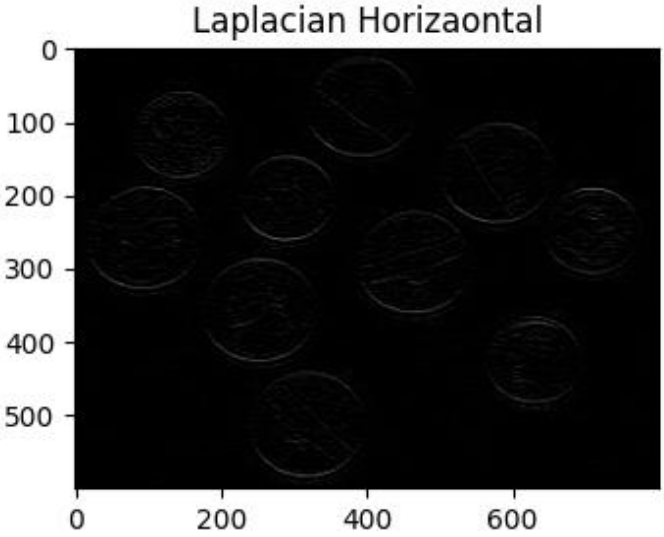
II. RESULTS:

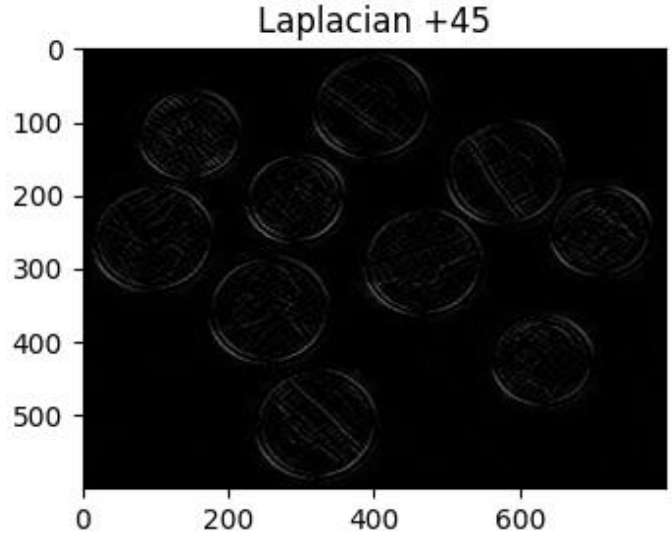

Original image
0
100
200
300
400
500
0
200
400
600

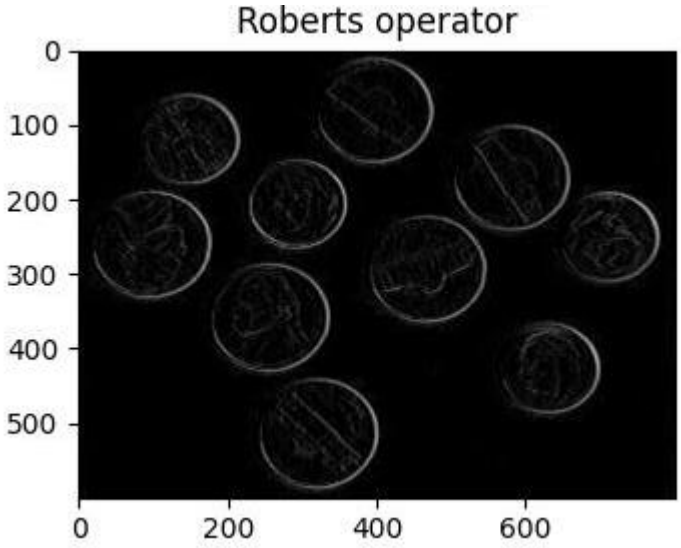

Point Detection
0
100
200
300
400
500
0
200
400
600

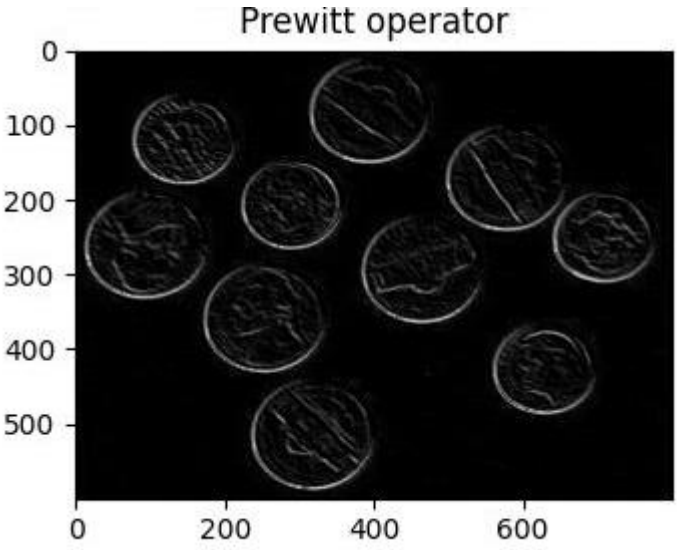

Laplacian -45
0
100
200
300
400
500
0
200
400
600

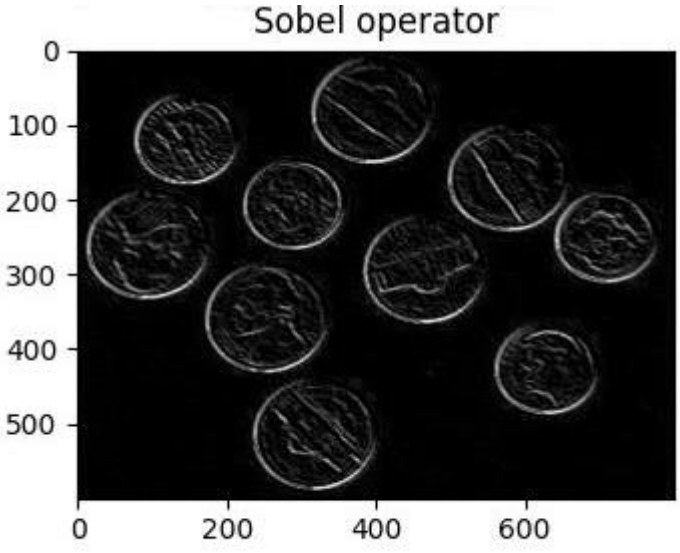

Laplacian Vertical
0
100
200
300
400
500
0
200
400
600


Laplacian Horizaontal
0
100
200
300
400
500
0
200
400
600


Laplacian +45
0
100
200
300
400
500
0
200
400
600


Roberts operator
0
100
200
300
400
500
0
200
400
600


Prewitt operator
0
100
200
300
400
500
0
200
400
600


Sobel operator
0
100
200
300
400
500
0
200
400
600