Bansilal Ramnath Agarwal Charitable Trust's

Vishwakarma Institute of Information Technology

Department of Artificial Intelligence and Data Science

Name: Siddhesh Dilip Khairnar		
Class: TY	Division: B	Roll No: 372028
Semester: V	Academic Year: 2023-2024	
Subject Name & Code: Image Processing: ADUA31205(B)		
Title of Assignment: Perform edge detection from an image using derivatives and filters.		
Date of Performance: 29-08-2023	Date of Submission: 12-09-2023	

ASSIGNMENT NO. 4

Name: siddhesh Dilip khairnar
PRN no: 22110398
Roll no: 372028

Page No.
Date

IP Assignment no 4

Aim: To perform edge detection from an image using derivative & filter

Learning Objective:-

i) To learn about edge detectives
ii) To learn types of derivative & filter available for edge detection.
iii) To learn how to implement edge detection in python.

Theory:-

Edge detection is an image processing discipline that incoperates mathematics method to find edges (first order derivative) in an digital image. There are 2 form of edge detection:-

i) Search based edge detection (first order derivative)
ii) Zero crossing based edge detection (second order derivative)

Some commonly known edge detection method are:-

i) Laplacian operator (second order derivative)
ii) Canny edge detector (first order derivative)
iii) Prewitt operator (first order derivative)
iv) Robel operator (first order derivative)

Edge detection operator are of 2 types:-

i) Gradient based operator which computer first order derivative in a digital image like sobel operator, prewitt operator etc.
ii) Gaussian based operator which computes second order derivative in a digital image.

A) Sobel operator: It is a discrete differentiation operator that computes the gradient approximation of image intensity function.
Mask: →

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

B) Prewitt operator: This operator is almost similar to the sobel operator. It detect vertical & horizontal edges in an image.
Mask: →

$$M_x = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \qquad M_y = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

C) Robert operator: This gradient based operator computes the sum of square of the difference b/w diagonally adjacent pixel in an image through discrete differentiation.
Mask: →

$$M_x = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad M_y = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

D) Marr-Hildreth operator or laplacian of Gaussian: It is a gaussian based operator which uses the laplacian to take the second derivatives of an image.

Page No.
Date

Gaussian fun^n: →

$$G(x,y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(\frac{x^2+y^2}{2\sigma^2}\right)$$ where σ = standard deviation

The LoG operator is computed from: →

$$LOG = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2} = \frac{x^2+y^2-2\sigma^2}{\sigma^4} \exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$$

e) Canny operator: It is a gaussian based operator in detecting edges. This operator is not suspective to noise

Conclusion: → Thus, we successfully performed edges detection from an image using derivative & filter.

21/10/23

Program Code:

```
import cv2
import numpy as np

# Read the image
image = cv2.imread('/content/pexels-jonathan-borba-3076516.jpg',
cv2.IMREAD_GRAYSCALE)

# Define Robert operator kernels
robert_x = np.array([[1, 0], [0, -1]])
robert_y = np.array([[0, 1], [-1, 0]])

# Apply Robert operator
robert_x_edges = cv2.filter2D(image, cv2.CV_64F, robert_x)
robert_y_edges = cv2.filter2D(image, cv2.CV_64F, robert_y)

# Calculate edge magnitude
robert_magnitude = cv2.magnitude(robert_x_edges, robert_y_edges)

# Display the original image and the edge magnitude
cv2.imwrite('Original_Image.jpg', image)
cv2.imwrite('Robert_Edge_Magnitude.jpg', robert_magnitude)

# Wait for a key press and close windows
cv2.waitKey(0)
cv2.destroyAllWindows()

# ... (Read image and import libraries)

# Apply Sobel operator
sobel_x_edges = cv2.Sobel(image, cv2.CV_64F, 1, 0, ksize=3)
sobel_y_edges = cv2.Sobel(image, cv2.CV_64F, 0, 1, ksize=3)

# Calculate edge magnitude
sobel_magnitude = cv2.magnitude(sobel_x_edges, sobel_y_edges)

# Display the original image and the edge magnitude
#cv2.imwrite('Original_Image', image)
cv2.imwrite('Sobel_Edge_Magnitude.jpg', sobel_magnitude)

# Wait for a key press and close windows
cv2.waitKey(0)
cv2.destroyAllWindows()

# ... (Read image and import libraries)

# Define Prewitt operator kernels
prewitt_x = np.array([[-1, 0, 1], [-1, 0, 1], [-1, 0, 1]])
```

```
prewitt_y = np.array([[-1, -1, -1], [0, 0, 0], [1, 1, 1]])

# Apply Prewitt operator
prewitt_x_edges = cv2.filter2D(image, cv2.CV_64F, prewitt_x)
prewitt_y_edges = cv2.filter2D(image, cv2.CV_64F, prewitt_y)

# Calculate edge magnitude
prewitt_magnitude = cv2.magnitude(prewitt_x_edges, prewitt_y_edges)

# Display the original image and the edge magnitude
#cv2.imshow('Original Image', image)
cv2.imwrite('Prewitt_Edge_Magnitude.jpg', prewitt_magnitude)

# Wait for a key press and close windows
cv2.waitKey(0)
cv2.destroyAllWindows()

# ... (Read image and import libraries)

# Apply Laplacian operator
laplacian_edges = cv2.Laplacian(image, cv2.CV_64F)

# Display the original image and the Laplacian edges
#cv2.imshow('Original Image', image)
cv2.imwrite('Laplacian_Edges.jpg', laplacian_edges)

# Wait for a key press and close windows
cv2.waitKey(0)
cv2.destroyAllWindows()

# ... (Read image and import libraries)

# Apply Canny edge detection
canny_edges = cv2.Canny(image, threshold1=100, threshold2=200)

# Display the original image and the Canny edges
#cv2.imshow('Original Image', image)
cv2.imwrite('Canny_Edges.jpg', canny_edges)

# Wait for a key press and close windows
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Output: