|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  | 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**

A piece of paper with writing on it

Description automatically generated A paper with writing on it

Description automatically generated

A piece of paper with writing on it

Description automatically generated

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:

A person on a skateboard

Description automatically generatedA person doing a sit up exercise

Description automatically generated with medium confidenceA person doing sit ups in a gym

Description automatically generatedA person doing sit ups in a gym

Description automatically generatedA person doing sit ups

Description automatically generatedA person doing sit ups

Description automatically generated