Computer Vision

Home Work 10

Name – ROHIT DAS Student ID – NTNU_61047086s

Project – Zero Crossing Edge Detection

Language and library used: Python, OpenCV, Numpy.

Description: This program will perform the following functions while executing lena.bmp image file:

1. Convert image to Grayscale
2. Create a separate convolution function and a convolve value function for better readability of code
3. Implement 2 Laplacian Mask, Minimum Variance Laplacian, Laplacian of Gaussian, and Difference of Gaussian (inhibitory sigma=3, excitatory sigma=1, kernel size 11x11).
4. Please list the kernels and the thresholds (for zero crossing) you used.
5. Threshold Values listed below are for reference:
    a. Laplace Mask1 (0, 1, 0, 1, -4, 1, 0, 1, 0): 15
    b. Laplace Mask2 (1, 1, 1, 1, -8, 1, 1, 1, 1)
    c. Minimum variance Laplacian: 20
    d. Laplace of Gaussian: 3000
    e. Difference of Gaussian: 1
    f. Zero Cross Edge Detection with (e)

Parameters: None. Please Copy-paste the image path inside the program.

Algorithms Used –

Part 1: Convert Image to Grayscale

Principal Code:

```
image_file = r"F:\Fall 2021 NTNU\Computer Vision NTU\Chapter-7\HomeWork-
10\lena.bmp"
gray_image = cv2.imread(image_file, 0)
```

Part 2: Create a separate convolution function and a convolve value function for better readability of code

Principal Code:

```
def convolve(img, kernel):
    row_img, col_img = img.shape
    row_k, col_k = kernel.shape
    res = 0
    for h_row in range(row_img):
        for w_col in range(col_img):
            if 0 <= row_img - h_row - 1 < row_k and 0 <= col_img - w_col -
```

```
1 < col_k:
                res += img[h_row, w_col] * kernel[row_img - h_row - 1,
col_img - w_col - 1]
    return res


def convolution_image(gray_image,kernel, threshold):
    laplace_mask_img = np.copy(gray_image)
    # To avoid index out of bounds Error
    index_control = np.zeros((gray_image.shape[0] - kernel.shape[0] + 1,
gray_image.shape[1] - kernel.shape[1] + 1))
    for h_row in range(index_control.shape[0]):
        for w_col in range(index_control.shape[1]):
            value = convolve(gray_image[h_row:h_row + kernel.shape[0],
w_col:w_col + kernel.shape[1]], kernel)
            if value >= threshold:
                laplace_mask_img[h_row, w_col] = 1
            elif value <= -threshold:
                laplace_mask_img[h_row, w_col] = -1
            else:
                laplace_mask_img[h_row, w_col] = 255

    return laplace_mask_img
```

Part 3: Laplacian Mask 1

Principal Code :

```
# Laplace mask 1
laplace_mask1_kernel = np.array([
    [0, 1, 0],
    [1, -4, 1],
    [0, 1, 0]
])
laplace_mask1_image = convolution_image(gray_image,laplace_mask1_kernel,
15)
```

Part 4: Laplacian Mask 2

Principal Code:

```
# Laplace mask 1
laplace_mask1_kernel = np.array([
    [1, 1, 1],
    [1, -8, 1],
    [1, 1, 1]
])/3
laplace_mask1_image = convolution_image(gray_image,laplace_mask1_kernel,
15)
```

Part 4: Minimum Variance Laplacian Image

Principal Code:

```
minimum_variance_laplacian_kernel = np.array([
        [2., -1, 2],
        [-1, -4, -1],
        [2, -1, 2]
```

```
    ]) / 3
minimum_variance_laplacian_image = convolution_image(gray_image,
minimum_variance_laplacian_kernel, 15)
cv2.imshow('Minimum Variance Laplacian Image',
minimum_variance_laplacian_image)
cv2.waitKey(0)
```

Part 5: Laplace of Gaussian

Principal Code:

```
Laplace_of_Gaussian_kernel = np.array([
    [0, 0, 0, -1, -1, -2, -1, -1, 0, 0, 0],
    [0, 0, -2, -4, -8, -9, -8, -4, -2, 0, 0],
    [0, -2, -7, -15, -22, -23, -22, -15, -7, -2, 0],
    [-1, -4, -15, -24, -14, -1, -14, -24, -15, -4, -1],
    [-1, -8, -22, -14, 52, 103, 52, -14, -22, -8, -1],
    [-2, -9, -23, -1, 103, 178, 103, -1, -23, -9, -2],
    [-1, -8, -22, -14, 52, 103, 52, -14, -22, -8, -1],
    [-1, -4, -15, -24, -14, -1, -14, -24, -15, -4, -1],
    [0, -2, -7, -15, -22, -23, -22, -15, -7, -2, 0],
    [0, 0, -2, -4, -8, -9, -8, -4, -2, 0, 0],
    [0, 0, 0, -1, -1, -2, -1, -1, 0, 0, 0]
])
laplace_of_gaussian_image = convolution_image(gray_image,
Laplace_of_Gaussian_kernel, 3000)
cv2.imshow('Laplace of Gaussian Image', laplace_of_gaussian_image)
cv2.waitKey(0)
```

Part 6: Difference of Gaussian

Principal Code:

```
# Difference of Gaussian
difference_of_gaussian_kernel = np.array([
    [-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1],
    [-3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3],
    [-4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4],
    [-6, -11, -16, -16, 0, 15, 0, -16, -16, -11, -6],
    [-7, -13, -17, 0, 85, 160, 85, 0, -17, -13, -7],
    [-8, -13, -17, 15, 160, 283, 160, 15, -17, -13, -8],
    [-7, -13, -17, 0, 85, 160, 85, 0, -17, -13, -7],
    [-6, -11, -16, -16, 0, 15, 0, -16, -16, -11, -6],
    [-4, -8, -12, -16, -17, -17, -17, -16, -12, -8, -4],
    [-3, -5, -8, -11, -13, -13, -13, -11, -8, -5, -3],
    [-1, -3, -4, -6, -7, -8, -7, -6, -4, -3, -1],
])
difference_of_gaussian_image = convolution_image(gray_image,
difference_of_gaussian_kernel, 1)
cv2.imshow('Difference of Gaussian Image', difference_of_gaussian_image)
cv2.waitKey(0)
```

Part 7: Zero Crossing Edge Detection

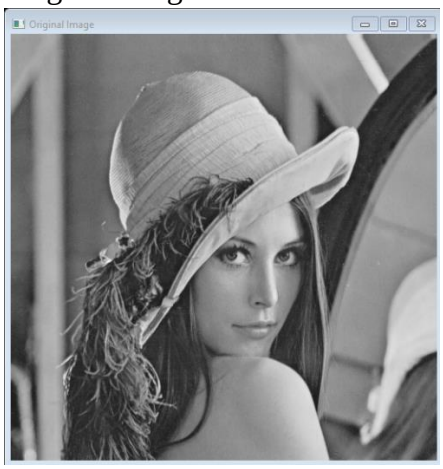
Principal Code:

```
def zero_crossing(gray_image):
    # 8 neighborhood zero crossing
```

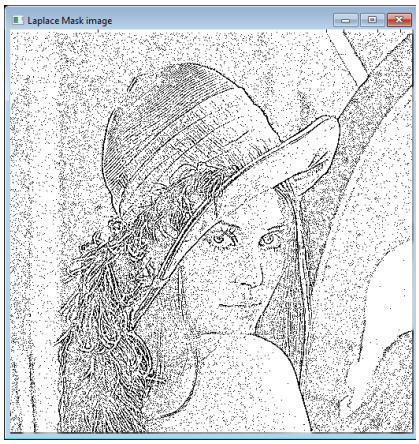
```
    zero_cross_image = np.zeros(gray_image.shape)
    for h_row in range(0, gray_image.shape[0] - 1):
        for w_col in range(0, gray_image.shape[1] - 1):
            if gray_image[h_row][w_col] > 0:
                if gray_image[h_row + 1][w_col] < 0 or gray_image[h_row +
1][w_col + 1] < 0 or gray_image[h_row][w_col + 1] < 0:
                    zero_cross_image[h_row, w_col] = 1
            elif gray_image[h_row][w_col] < 0:
                if gray_image[h_row + 1][w_col] > 0 or gray_image[h_row +
1][w_col + 1] > 0 or gray_image[h_row][w_col + 1] > 0:
                    zero_cross_image[h_row, w_col] = 1
    return zero_cross_image
```

Example:

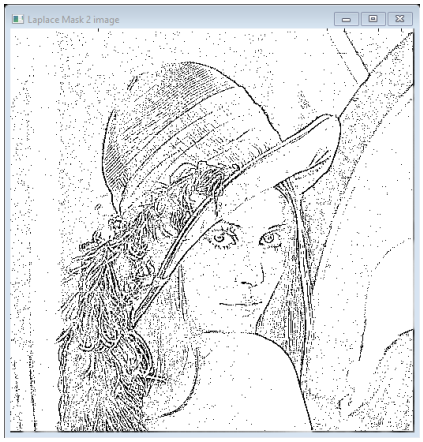
- Original image



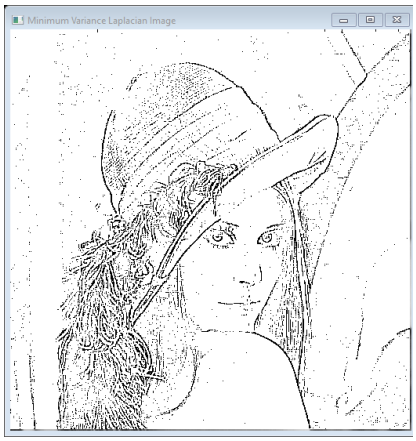
- Laplace Mask 1 at threshold 15



- Laplace Mask 2 at threshold 15



- Minimum Variance Laplacian threshold 15



- Laplace of Gaussian Image threshold 3000

- Difference of Gaussian Image threshold 1

- Zero Crossing Edge Detection