Digital Image processing Laboratory(18AIL67)

"Transforming pixels into insights: Discovering the world through the power of image processing"

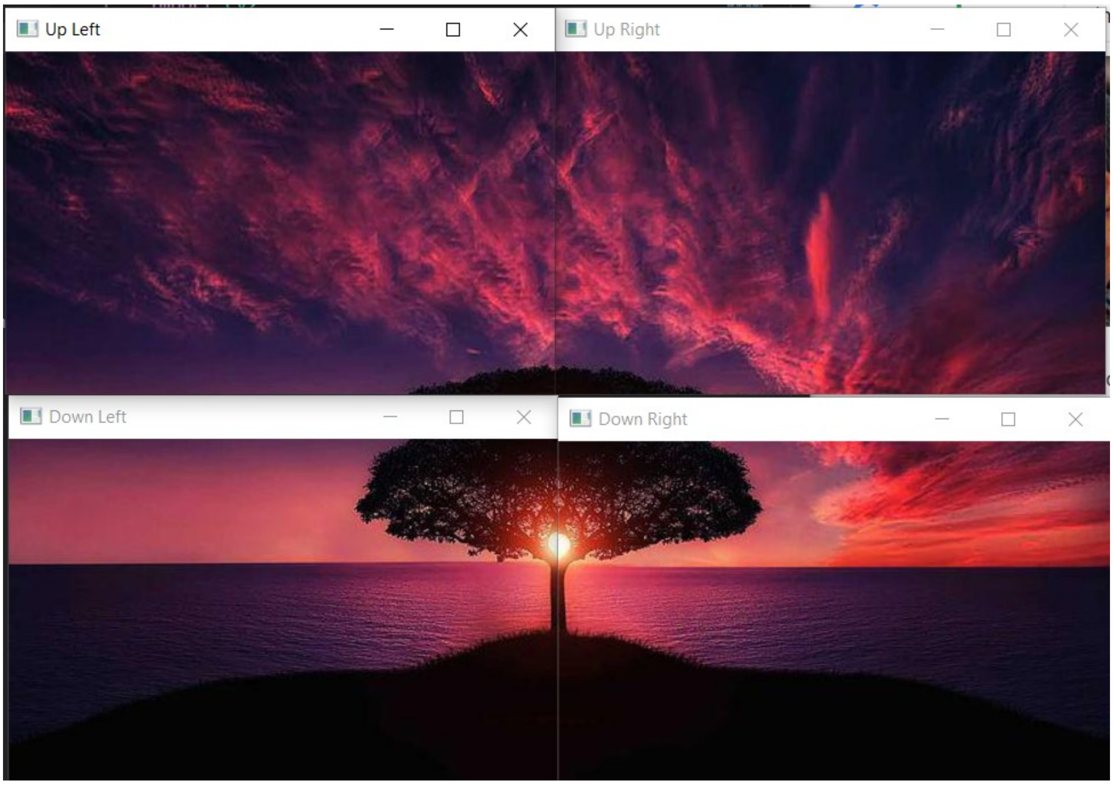
INDEX

PROGRAM NO.1

Write a program to read a digital image.split and display image into 4 quadrants up,down,right and left.

```
import cv2

# Load the image
image = cv2.imread("img.jpg")

# Get the dimensions of the image
height, width = image.shape[:2]

# Split the image into four quadrants
up_left = image[0:height//2, 0:width//2]
up_right = image[0:height//2, width//2:width]
down_left = image[height//2:height, 0:width//2]
down_right = image[height//2:height, width//2:width]

# Display the quadrants
cv2.imshow("Up Left", up_left)
cv2.imshow("Up Right", up_right)
cv2.imshow("Down Left", down_left)
cv2.imshow("Down Right", down_right)

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

OUTPUT



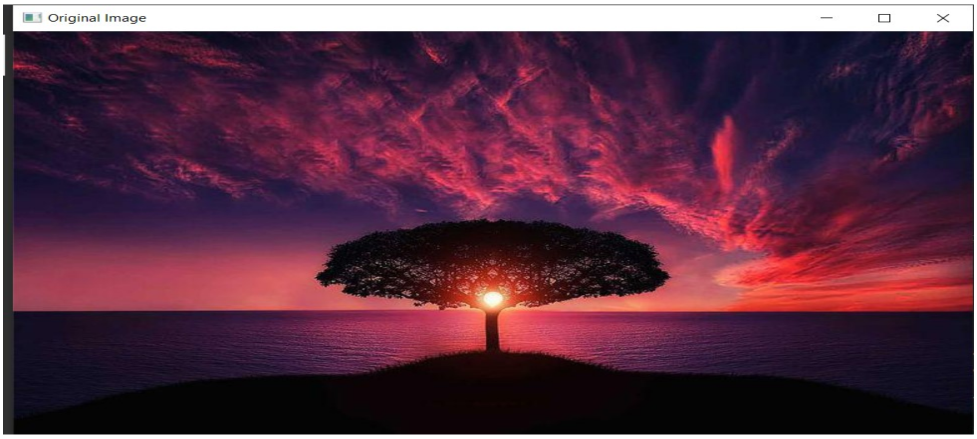
PROGRAM NO.2

Write a program to show rotation,scaling and translation of an image

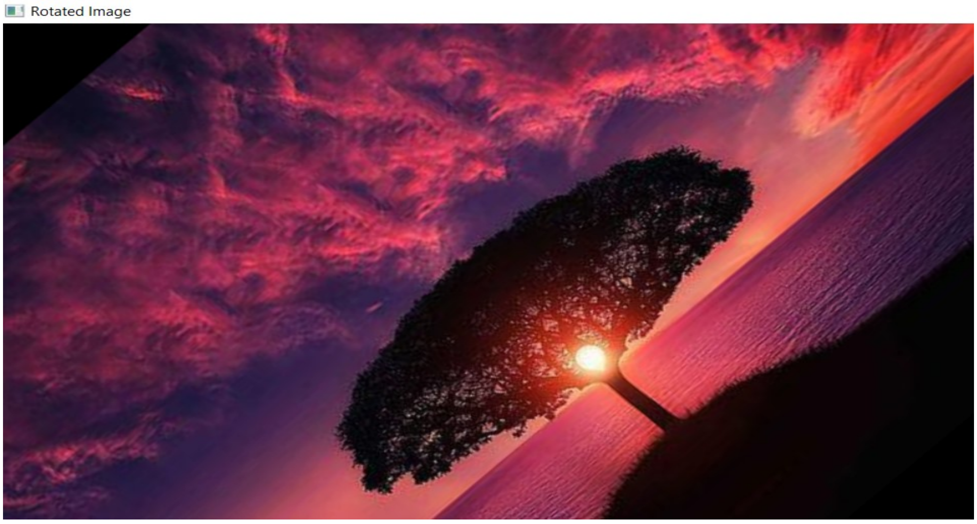
```
import cv2
import numpy as np

# Load the image
image = cv2.imread("iamge.jpg")

# Get the dimensions of the image
height, width = image.shape[:2]

# Define the rotation angle, scaling factor, and translation offsets
angle = 45
scale = 1.5
dx = 50
dy = -100

# Perform rotation, scaling, and translation on the image
rotation_matrix = cv2.getRotationMatrix2D((width/2, height/2), angle, scale)
rotated_image = cv2.warpAffine(image, rotation_matrix, (width, height))
translation_matrix = np.float32([[1, 0, dx], [0, 1, dy]])
translated_image = cv2.warpAffine(rotated_image, translation_matrix, (width, height))

# Display the original image, rotated image, scaled image, and translated image
cv2.imshow("Original Image", image)
cv2.imshow("Rotated Image", rotated_image)
cv2.imshow("Scaled Image", cv2.resize(image, None, fx=scale, fy=scale))
cv2.imshow("Translated Image", translated_image)

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

OUTPUT

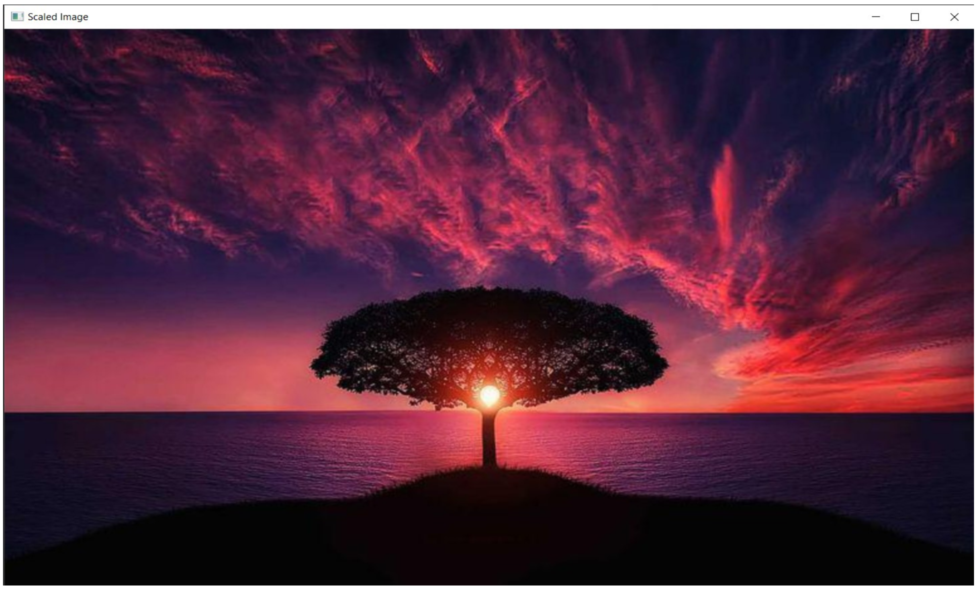
ORIGINAL IMAGE



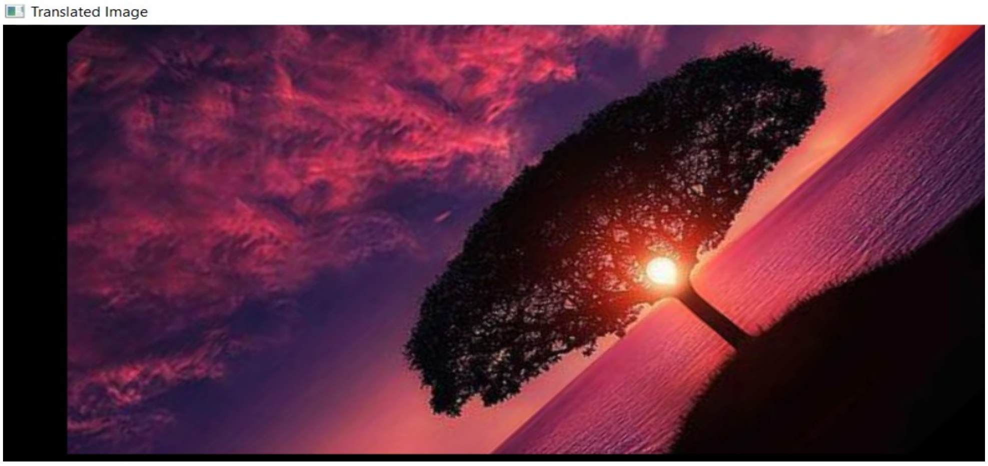
ROTATED IMAGE



SCALED IMAGE



TRANSLATED IMAGE



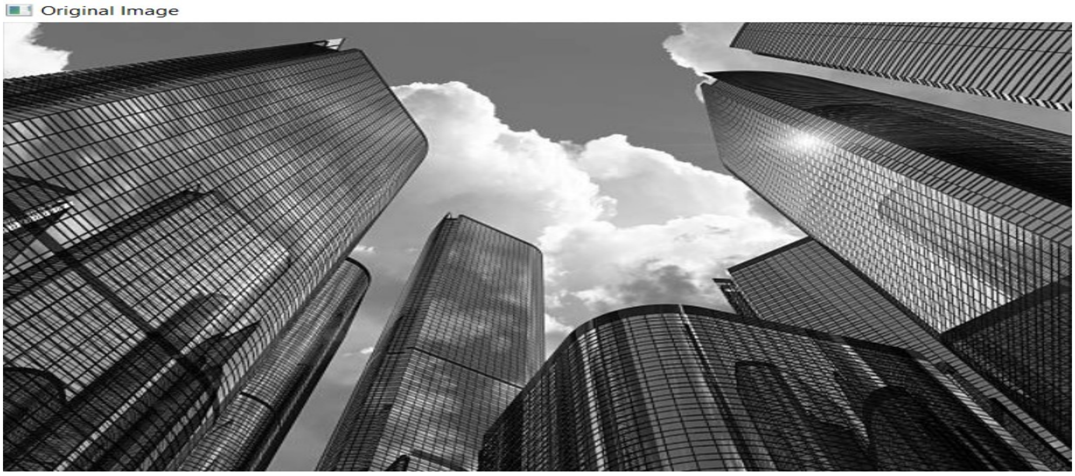
PROGRAM NO.3

Read an image,first apply erosion to the image and then subtract the result from the original,demonstrate the difference the edge if you use dilation instead of erosion

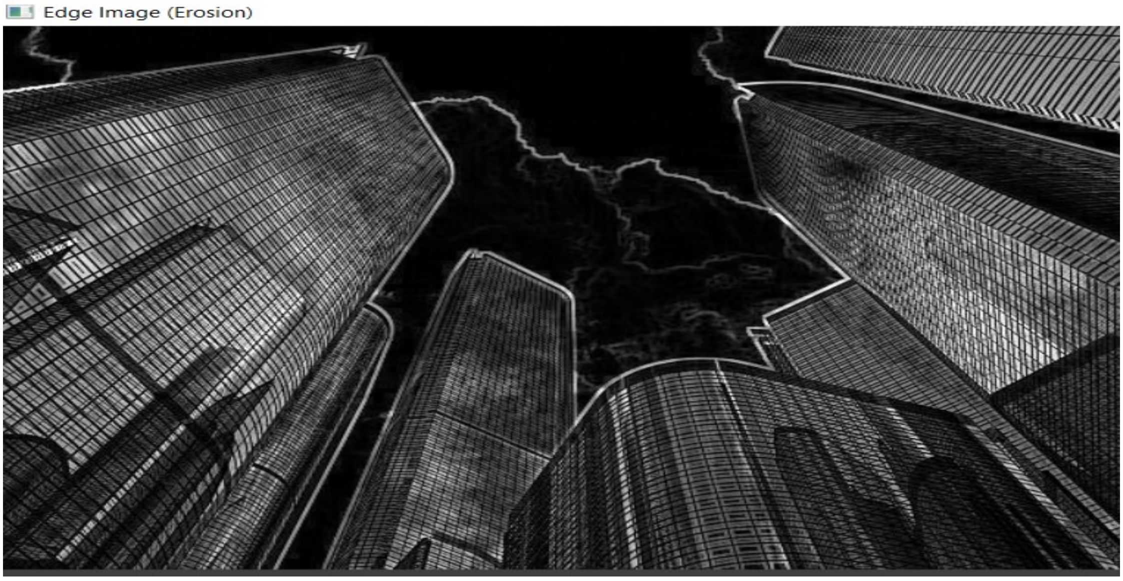
```
import cv2
import numpy as np

# Load the image
image = cv2.imread("image.jpg", cv2.IMREAD_GRAYSCALE)

# Define the erosion kernel and apply it to the image
kernel = np.ones((5, 5), np.uint8)
eroded_image = cv2.erode(image, kernel, iterations=1)

# Subtract the eroded image from the original image
edge_image = cv2.absdiff(image, eroded_image)

# Display the original image and edge image
cv2.imshow("Original Image", image)
cv2.imshow("Edge Image (Erosion)", edge_image)

# Apply dilation to the image
dilated_image = cv2.dilate(image, kernel, iterations=1)

# Subtract the dilated image from the original image
edge_image_dilation = cv2.absdiff(image, dilated_image)

# Display the edge image with dilation
cv2.imshow("Edge Image (Dilation)", edge_image_dilation)

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

OUTPUT

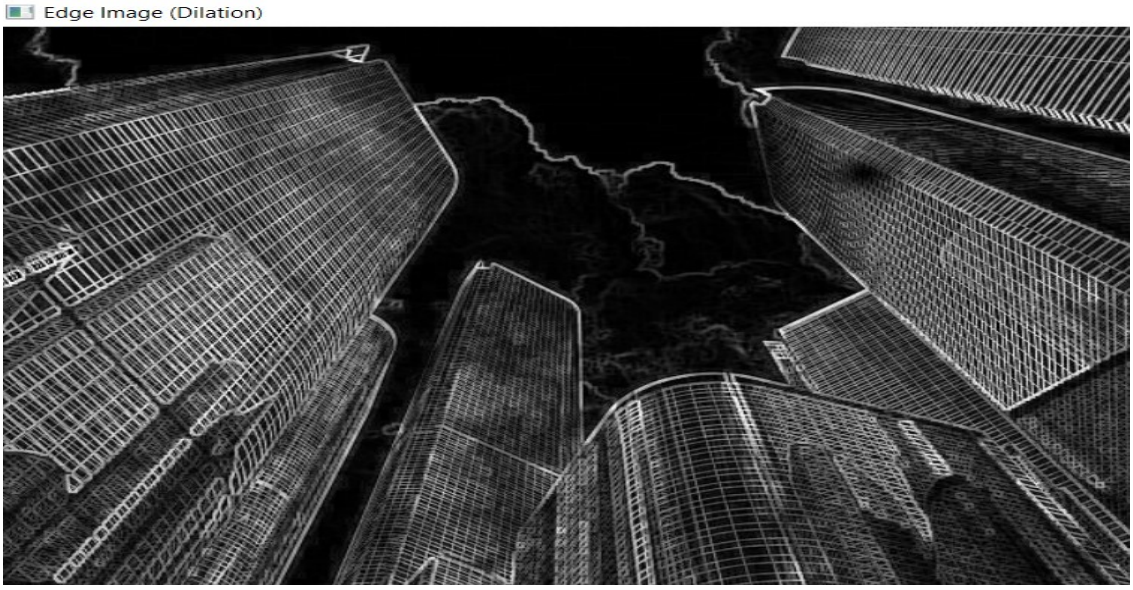
ORIGINAL IMAGE



EROSION IMAGE



DILATION IMAGE



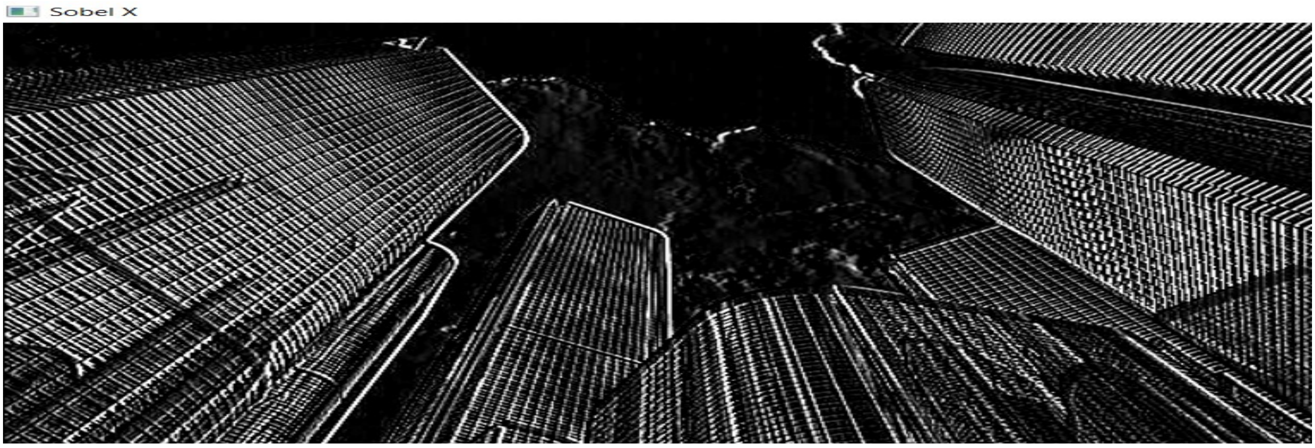
PROGRAM NO.4

Read an image and extract and display low-level feature such as edges,textures using filtering technique

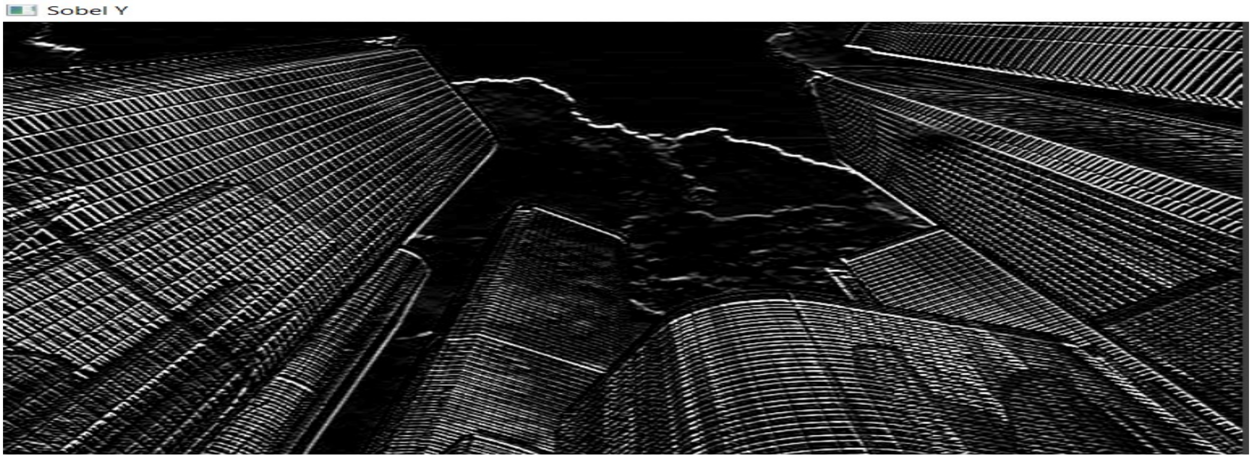
```
import cv2
import numpy as np

# Load the image
image = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Define the Sobel edge filter kernels
sobel_x = np.array([[-1, 0, 1], [-2, 0, 2], [-1, 0, 1]])
sobel_y = np.array([[-1, -2, -1], [0, 0, 0], [1, 2, 1]])

# Apply the Sobel edge filters to the image
edge_x = cv2.filter2D(image, -1, sobel_x)
edge_y = cv2.filter2D(image, -1, sobel_y)

# Display the edge images
cv2.imshow('Sobel X', edge_x)
cv2.imshow('Sobel Y', edge_y)

# Define the texture filter kernel
texture = np.array([[0, 1, 0], [1, -4, 1], [0, 1, 0]])

# Apply the texture filter to the image
texture_image = cv2.filter2D(image, -1, texture)

# Display the texture image
cv2.imshow('Texture', texture_image)

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

OUTPUT

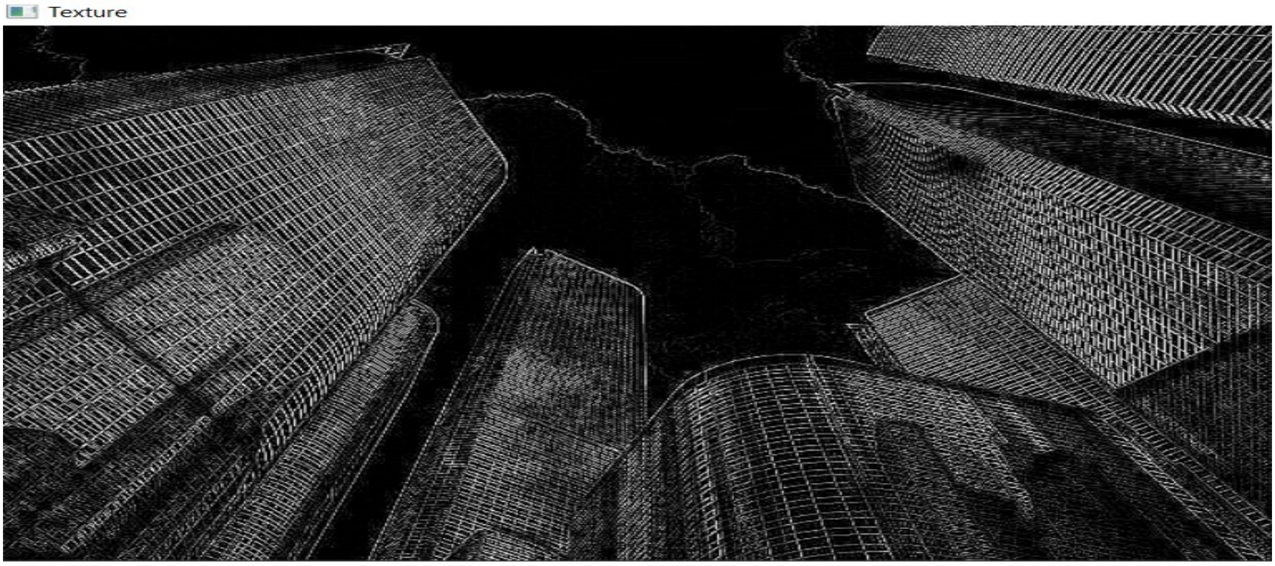
Edge SOBEL X



Edge SOBEL Y



TEXTURE



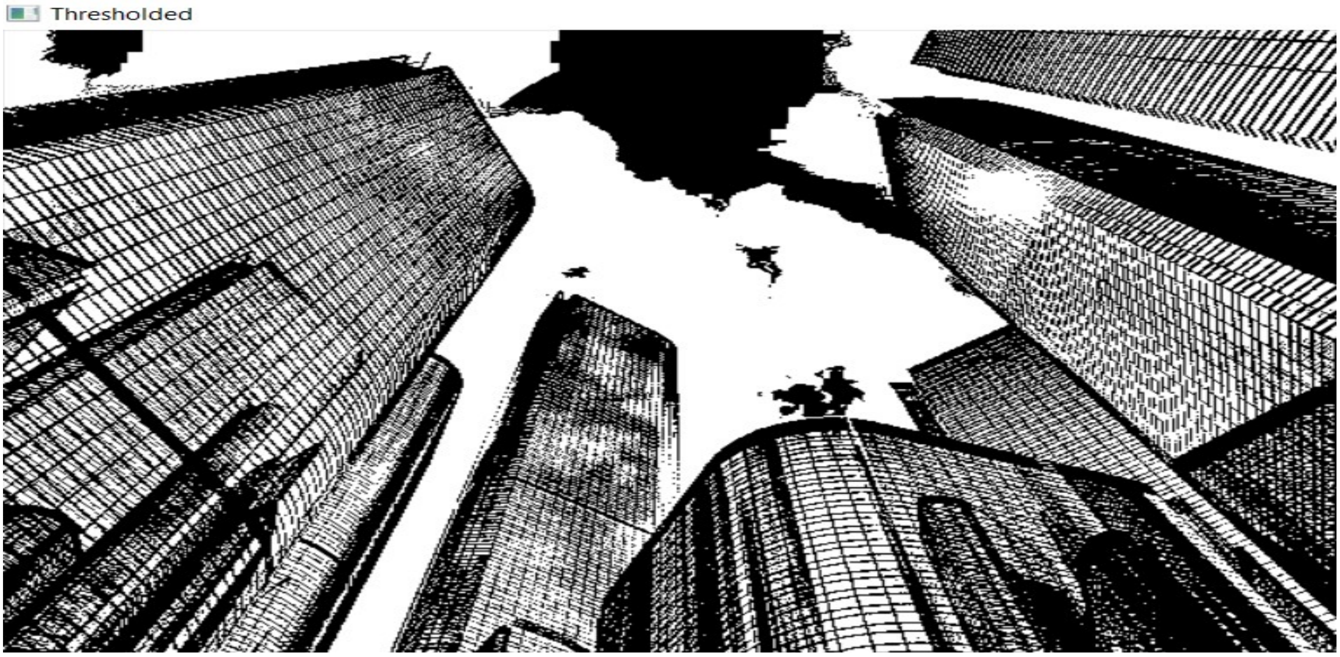
PROGRAM NO.5

Demonstrate enhancing and segmentation low contrast 2D images

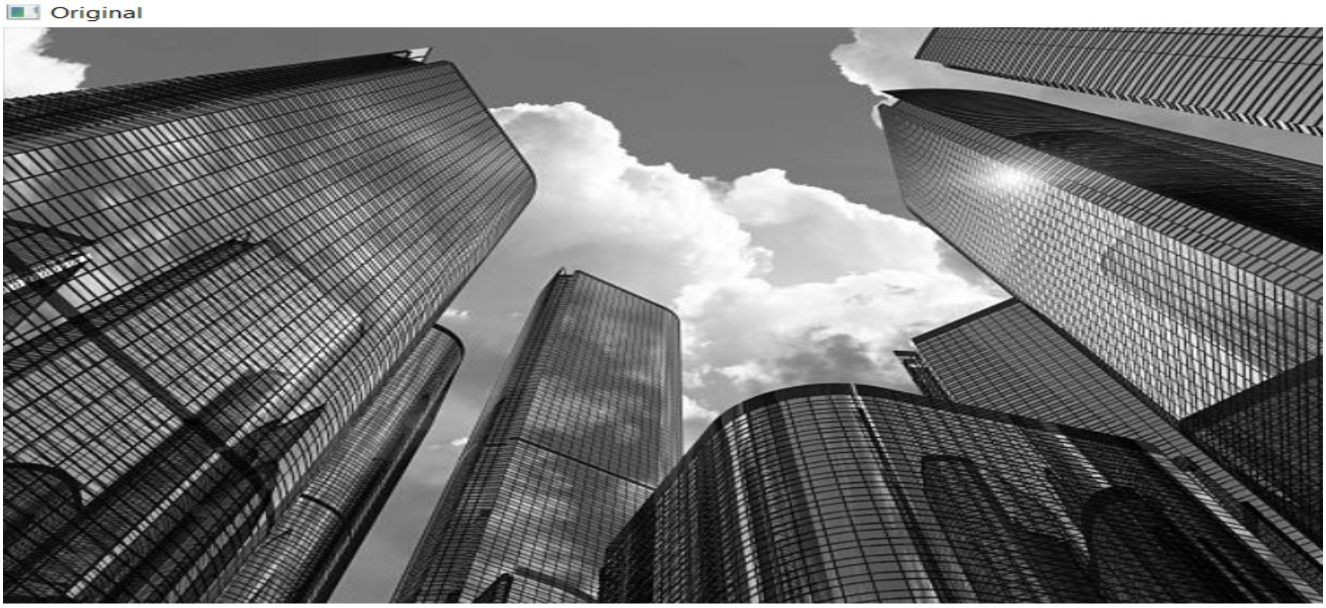
```
import cv2
import numpy as np

# Load the low contrast image
image = cv2.imread('image.jpg', cv2.IMREAD_GRAYSCALE)

# Apply adaptive histogram equalization to enhance the contrast
clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize=(8,8))
enhanced_image = clahe.apply(image)

# Display the original and enhanced images
cv2.imshow('Original', image)
cv2.imshow('Enhanced', enhanced_image)

# Apply Otsu's thresholding to segment the image
_, threshold_image = cv2.threshold(enhanced_image, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# Display the thresholded image
cv2.imshow('Thresholded', threshold_image)

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

OUTPUT

THRESHOLDED



ORIGINAL



ENHANCED