7. **Write a Program to read a digital image. Split and display image into 4 quadrants, up, down, right and left.**

import cv2

# Function to split the image into four quadrants

def split\_image(image):

height, width, \_ = image.shape

half\_height = height // 2

half\_width = width // 2

# Split the image into four quadrants

top\_left = image[:half\_height, :half\_width]

top\_right = image[:half\_height, half\_width:]

bottom\_left = image[half\_height:, :half\_width]

bottom\_right = image[half\_height:, half\_width:]

return top\_left, top\_right, bottom\_left, bottom\_right

# Function to display images

def display\_images(images, window\_names):

for img, name in zip(images, window\_names):

cv2.imshow(name, img)

print("Press any key to terminate.")

cv2.waitKey(0)

cv2.destroyAllWindows()

# Read the image

image\_path = "city.jpg" # Replace "image.jpg" with the path to your image

image = cv2.imread(image\_path)

if image is None:

print("Failed to load the image.")

else:

# Split the image into quadrants

top\_left, top\_right, bottom\_left, bottom\_right = split\_image(image)

# Display the quadrants

display\_images([top\_left, top\_right, bottom\_left, bottom\_right], ["Top Left", "Top Right", "Bottom Left", "Bottom Right"])

## 8. . Write a program to show rotation, scaling, and translation on an image.

import cv2

import numpy as np

# Read the image

image\_path = "peacock.jpg" # Replace "your\_image.jpg" with the path to your image

image = cv2.imread(image\_path)

if image is None:

print("Failed to load the image.")

else:

# Display the original image

cv2.imshow("Original Image", image)

# Rotation

angle = 45 # Rotation angle in degrees

center = (image.shape[1] // 2, image.shape[0] // 2) # Center of rotation

rotation\_matrix = cv2.getRotationMatrix2D(center, angle, 1.0) # Rotation matrix

rotated\_image = cv2.warpAffine(image, rotation\_matrix, (image.shape[1], image.shape[0]))

# Scaling

scale\_factor = 0.5 # Scaling factor (0.5 means half the size)

scaled\_image = cv2.resize(image, None, fx=scale\_factor, fy=scale\_factor)

# Translation

translation\_matrix = np.float32([[1, 0, 100], [0, 1, -50]]) # Translation matrix (100 pixels right, 50 pixels up)

translated\_image = cv2.warpAffine(image, translation\_matrix, (image.shape[1], image.shape[0]))

# Display the transformed images

cv2.imshow("Rotated Image", rotated\_image)

cv2.imshow("Scaled Image", scaled\_image)

cv2.imshow("Translated Image", translated\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

## 9. Read an image and extract and display low-level features such as edges, textures using filtering techniques.

import cv2

# Read the image

image\_path = "art-1.png" # Replace "your\_image.jpg" with the path to your image

image = cv2.imread(image\_path)

if image is None:

print("Failed to load the image.")

else:

# Display the original image

cv2.imshow("Original Image", image)

# Apply blur to the image

blur\_kernel\_size = (5, 5) # Kernel size for blur filter

blurred\_image = cv2.blur(image, blur\_kernel\_size)

# Display the blurred image

cv2.imshow("Blurred Image", blurred\_image)

# Apply Gaussian blur to the image

gaussian\_blur\_kernel\_size = (5, 5) # Kernel size for Gaussian blur filter

gaussian\_blurred\_image = cv2.GaussianBlur(image, gaussian\_blur\_kernel\_size, 0)

# Display the Gaussian blurred image

cv2.imshow("Gaussian Blurred Image", gaussian\_blurred\_image)

# Apply median blur to the image

median\_blur\_kernel\_size = 5 # Kernel size for median blur filter (should be odd)

median\_blurred\_image = cv2.medianBlur(image, median\_blur\_kernel\_size)

# Display the median blurred image

cv2.imshow("Median Blurred Image", median\_blurred\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

## 10. Write a program to blur and smoothing an image.

import cv2

# Read the image

image\_path = "annavru-1.jpeg" # Replace "your\_image.jpg" with the path to your image

image = cv2.imread(image\_path)

if image is None:

print("Failed to load the image.")

else:

# Convert the image to grayscale

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Apply adaptive thresholding

\_, thresh = cv2.threshold(gray\_image, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)

# Find contours in the thresholded image

contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Draw contours on the original image

contour\_image = image.copy()

cv2.drawContours(contour\_image, contours, -1, (0, 255, 0), 2) # Draw all contours with green color and thickness 2

# Display the original image with contours

cv2.imshow("Image with Contours", contour\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

## 11. Write a program to contour an image.

import cv2

# Read the image

image\_path = "annavru-1.jpeg" # Replace "your\_image.jpg" with the path to your image

image = cv2.imread(image\_path)

if image is None:

print("Failed to load the image.")

else:

# Convert the image to grayscale

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Apply adaptive thresholding

\_, thresh = cv2.threshold(gray\_image, 0, 255, cv2.THRESH\_BINARY\_INV + cv2.THRESH\_OTSU)

# Find contours in the thresholded image

contours, \_ = cv2.findContours(thresh, cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

# Draw contours on the original image

contour\_image = image.copy()

cv2.drawContours(contour\_image, contours, -1, (0, 255, 0), 2) # Draw all contours with green color and thickness 2

# Display the original image with contours

cv2.imshow("Image with Contours", contour\_image)

cv2.waitKey(0)

cv2.destroyAllWindows()

## 12. Write a program to detect a face/s in an image.

import cv2

# Load the pre-trained Haar Cascade classifier for face detection

face\_cascade = cv2.CascadeClassifier('./haarcascade\_frontalface\_default.xml')

# Read the image

image\_path = "ucl-2.png" # Replace "ucl.png" with the path to your image

image = cv2.imread(image\_path)

if image is None:

print("Failed to load the image.")

else:

# Convert the image to grayscale

gray\_image = cv2.cvtColor(image, cv2.COLOR\_BGR2GRAY)

# Detect faces in the image

faces = face\_cascade.detectMultiScale(gray\_image, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30))

# Draw rectangles around the detected faces

for (x, y, w, h) in faces:

cv2.rectangle(image, (x, y), (x+w, y+h), (0, 255, 0), 2)

# Display the image with detected faces

cv2.imshow("Image with Detected Faces", image)

cv2.waitKey(0)

cv2.destroyAllWindows()