1)

import cv2

import numpy as np

kernel = np.ones((5,5),np.uint8)

print(kernel)

path = r'C:\Users\sandh\Downloads\panda.jpg'

img =cv2.imread(path)

imgGray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

cv2.imshow("GrayScale",imgGray)

cv2.waitKey(0)

al:

 **Create a kernel** for image processing (5x5 matrix).

 **Read the image** from the specified path.

 **Convert the image to grayscale** using cvtColor().

 **Display the grayscale image** using imshow().

 **Wait for a key press** to close the image window.

2)

import cv2

import numpy as np

kernel = np.ones((5,5),np.uint8)

print(kernel)

path = r"C:\Users\sandh\Downloads\panda.jpg"

img =cv2.imread(path)

imgGray = cv2.cvtColor(img,cv2.COLOR\_BGR2GRAY)

imgBlur = cv2.GaussianBlur(imgGray,(7,7),0)

cv2.imshow("Img Blur",imgBlur)

cv2.waitKey(0)

al:

 **Create a 5x5 kernel** of ones for image processing.

 **Read the image** from the specified file path.

 **Convert the image to grayscale** using cvtColor().

 **Apply Gaussian blur** to the grayscale image using GaussianBlur().

 **Display the blurred image** using imshow().

3)

import cv2

path = r"C:\Users\sandh\Downloads\panda.jpg"

img = cv2.imread(path)

imgLarger = cv2.resize(img, None, fx=2, fy=2)

imgSmaller = cv2.resize(img, None, fx=0.5, fy=0.5)

cv2.imshow("Larger Image", imgLarger)

cv2.imshow("Smaller Image", imgSmaller)

cv2.waitKey(0)

al:

 **Read the image** from the specified file path.

 **Resize the image to a larger size** by setting scale factors (fx=2, fy=2).

 **Resize the image to a smaller size** by setting scale factors (fx=0.5, fy=0.5).

 **Display the larger image** using imshow().

 **Display the smaller image** using imshow().

4)

import cv2

import numpy as np

path = r"C:\Users\sandh\Downloads\panda.jpg"

img = cv2.imread(path)

imgGray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

imgEqualized = cv2.equalizeHist(imgGray)

cv2.imshow("Original Image", imgGray)

cv2.imshow("Equalized Image", imgEqualized)

cv2.waitKey(0)

al:

1. **Read the image** from the specified file path.
2. **Convert the image to grayscale** using cvtColor().
3. **Apply histogram equalization** to the grayscale image using equalizeHist().
4. **Display the original grayscale image** using imshow().
5. **Display the equalized image** using imshow().

5)

import cv2

import numpy as np

def plot\_color\_histogram\_opencv(image\_path):

img = cv2.imread(image\_path)

hist\_img = np.zeros((300, 256, 3), dtype=np.uint8)

for i, col in enumerate(('b', 'g', 'r')):

hist = cv2.calcHist([img], [i], None, [256], [0, 256])

hist = cv2.normalize(hist, hist, 0, 300, cv2.NORM\_MINMAX)

for j in range(1, 256):

cv2.line(hist\_img, (j - 1, 300 - int(hist[j - 1])),

(j, 300 - int(hist[j])), (255 \* (i == 0), 255 \* (i == 1), 255 \* (i == 2)), 1)

cv2.imshow("Color Histogram", hist\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

plot\_color\_histogram\_opencv(r"C:\Users\sandh\Downloads\panda.jpg")

al:

Read Image: Load the image with cv2.imread(image\_path).

Create Blank Image: Initialize a blank image for the histogram (hist\_img).

Calculate Histogram: For each color channel, calculate and normalize the histogram.

Draw Histogram: Draw lines for each intensity level of the histogram in the blank image, color-coded by channel.

Display Image: Show the histogram image with cv2.imshow and close on key press.

6)

import cv2

def rotate\_90\_y\_axis(image\_path):

img = cv2.imread(image\_path)

rotated\_img = cv2.transpose(cv2.flip(img, 1))

cv2.imshow("90-degree Rotation on Y-axis", rotated\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

rotate\_90\_y\_axis(r"C:\Users\sandh\Downloads\panda.jpg")

al:

Read Image: Load the image using cv2.imread(image\_path).

Flip Image: Apply horizontal flip with cv2.flip(img, 1).

Transpose Image: Apply transpose with cv2.transpose to swap rows and columns.

Display Rotated Image: Show the transformed image using cv2.imshow.

Close Window: Wait for key press and close the display window with cv2.destroyAllWindows()

7)

import cv2

def rotate\_180\_y\_axis(image\_path):

img = cv2.imread(image\_path)

rotated\_img = cv2.flip(img, 1)

cv2.imshow("180-degree Rotation on Y-axis", rotated\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

rotate\_180\_y\_axis(r"C:\Users\sandh\Downloads\panda.jpg")

al:

Read Image: Load the image using cv2.imread(image\_path).

Flip Image: Apply horizontal flip with cv2.flip(img, 1) to achieve 180-degree rotation along the y-axis.

Display Rotated Image: Show the flipped image using cv2.imshow.

Wait for Key Press: Use cv2.waitKey(0) to keep the window open until a key is pressed.

Close Window: Close the display window with cv2.destroyAllWindows().

8)

import cv2

def rotate\_270\_y\_axis(image\_path):

img = cv2.imread(image\_path)

rotated\_img = cv2.transpose(cv2.flip(img, 0))

cv2.imshow("270-degree Rotation on Y-axis", rotated\_img)

cv2.waitKey(0)

rotate\_270\_y\_axis(r"C:\Users\sandh\Downloads\panda.jpg")

al:

Read Image: Load the image using cv2.imread(image\_path).

Flip Image Vertically: Use cv2.flip(img, 0) to mirror the image across the x-axis.

Transpose Image: Apply cv2.transpose to swap rows and columns, achieving a 270-degree rotation.

Display Rotated Image: Show the rotated image with cv2.imshow.

Close Window: Wait for a key press with cv2.waitKey(0) and close the window using cv2.destroyAllWindows().

This process produces a 270-degree clockwise rotation effect along the y-axis.

9)

import cv2

import numpy as np

def apply\_erosion(image\_path, kernel\_size=3, iterations=1):

img = cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE)

kernel = np.ones((kernel\_size, kernel\_size), np.uint8)

eroded\_img = cv2.erode(img, kernel, iterations=iterations)

cv2.imshow("Original Image", img)

cv2.imshow("Eroded Image", eroded\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

apply\_erosion(r"C:\Users\sandh\Downloads\panda.jpg", kernel\_size=3, iterations=1)

al:

Read Image: Load the image in grayscale using cv2.imread(image\_path, cv2.IMREAD\_GRAYSCALE).

Define Kernel: Create a kernel (structuring element) with a specified size using np.ones((kernel\_size, kernel\_size), np.uint8).

Apply Erosion: Use cv2.erode(img, kernel, iterations=iterations) to erode the image.

Display Images: Show both the original and eroded images using cv2.imshow.

Close Window: Wait for a key press and close the window with cv2.waitKey(0) and cv2.destroyAllWindows().

10)

import cv2

import numpy as np

def translate\_image(image\_path, tx, ty):

img = cv2.imread(image\_path)

rows, cols = img.shape[:2]

translation\_matrix = np.float32([[1, 0, tx], [0, 1, ty]])

translated\_img = cv2.warpAffine(img, translation\_matrix, (cols, rows))

cv2.imshow("Original Image", img)

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

cv2.waitKey(0)

translate\_image(r"C:\Users\sandh\Downloads\panda.jpg", 100, 50)

al:

Read Image: Load the image using cv2.imread(image\_path).

Get Image Dimensions: Retrieve the image's rows and columns using img.shape[:2].

Define Translation Matrix: Create the translation matrix [[1, 0, tx], [0, 1, ty]], where tx and ty are the horizontal and vertical shifts.

Apply Translation: Use cv2.warpAffine with the translation matrix to shift the image.

Display Image: Show both the original and translated images using cv2.imshow.

11)

import cv2

import numpy as np

def apply\_smoothing\_filter(image\_path, kernel\_size=3):

img = cv2.imread(image\_path)

kernel = np.ones((kernel\_size, kernel\_size), np.float32) / (kernel\_size \* kernel\_size)

smoothed\_img = cv2.filter2D(img, -1, kernel)

cv2.imshow("Original Image", img)

cv2.imshow("Smoothed Image", smoothed\_img)

cv2.waitKey(0)

cv2.destroyAllWindows()

apply\_smoothing\_filter(r"C:\Users\sandh\Downloads\panda.jpg", kernel\_size=5)

al:

Read Image: Load the image using cv2.imread(image\_path).

Create Kernel: Define a square kernel of size kernel\_size x kernel\_size filled with equal values, and normalize it by dividing by kernel\_size^2.

Apply Filter: Use cv2.filter2D(img, -1, kernel) to convolve the image with the kernel and apply the smoothing.

Display Images: Show the original and smoothed images using cv2.imshow.

Close Window: Wait for a key press and close the window using cv2.waitKey(0) and cv2.destroyAllWindows().

12)

import cv2

import numpy as np

def modify\_roi(image\_path):

img = cv2.imread(image\_path)

x, y, w, h = 100, 100, 200, 150

roi = img[y:y+h, x:x+w]

img[y:y+h, x:x+w] = [0, 255, 0]

cv2.imshow("Original Image", img)

cv2.imshow("Modified Image", img)

cv2.waitKey(0)

modify\_roi(r"C:\Users\sandh\Downloads\panda.jpg")

al:

Read Image: Load the image using cv2.imread(image\_path).

Define ROI Coordinates: Specify the top-left corner (x, y) and the dimensions (width, height) of the ROI.

Extract ROI: Use slicing syntax img[y:y+h, x:x+w] to extract the ROI from the image.

Modify ROI: Apply the desired operation (e.g., filling, changing color) to the ROI: img[y:y+h, x:x+w] = [value].

Display Image: Show the modified image using cv2.imshow and close with cv2.waitKey(0) and cv2.destroyAllWindows().

13)

import cv2

def click\_event(event, x, y, flags, param):

if event == cv2.EVENT\_LBUTTONDOWN:

print(f"Coordinates: ({x}, {y})")

cv2.putText(img, f"({x}, {y})", (x, y), cv2.FONT\_HERSHEY\_SIMPLEX, 1, (0, 255, 0), 2)

cv2.imshow("Image", img)

img = cv2.imread(r"C:\Users\sandh\Downloads\panda.jpg")

cv2.imshow("Image", img)

cv2.setMouseCallback("Image", click\_event)

cv2.waitKey(0)

cv2.destroyAllWindows()

al:

Read Image: Load the image using cv2.imread(image\_path).

Set Mouse Callback: Use cv2.setMouseCallback("WindowName", click\_event) to handle mouse click events.

Capture Click Event: In the callback function, check for left mouse clicks (cv2.EVENT\_LBUTTONDOWN) and get coordinates (x, y).

Overlay Coordinates: Use cv2.putText to display the coordinates on the image at the clicked point.

Display Image: Use cv2.imshow to show the updated image and cv2.waitKey(0) to keep the window open.

14)

import cv2

import numpy as np

def bgr\_palette():

img = np.zeros((300, 512, 3), np.uint8)

cv2.namedWindow("BGR Palette")

cv2.createTrackbar("Blue", "BGR Palette", 0, 255, lambda x: None)

cv2.createTrackbar("Green", "BGR Palette", 0, 255, lambda x: None)

cv2.createTrackbar("Red", "BGR Palette", 0, 255, lambda x: None)

while True:

blue = cv2.getTrackbarPos("Blue", "BGR Palette")

green = cv2.getTrackbarPos("Green", "BGR Palette")

red = cv2.getTrackbarPos("Red", "BGR Palette")

img[:] = [blue, green, red]

cv2.imshow("BGR Palette", img)

if cv2.waitKey(1) == 27: break

cv2.destroyAllWindows()

bgr\_palette()

al:

Create Window: Initialize a window using cv2.namedWindow("WindowName").

Create Trackbars: Use cv2.createTrackbar for Blue, Green, and Red channels, setting the range from 0 to 255.

Update Values: Continuously retrieve trackbar values using cv2.getTrackbarPos for each color channel.

Modify Image: Update the image with the selected BGR values, e.g., img[:] = [blue, green, red].

Display Image: Show the updated image using cv2.imshow and exit with the Esc key (cv2.waitKey(1)).

15)

import cv2

def find\_contours(image\_path):

img = cv2.imread(image\_path)

gray = cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY)

edges = cv2.Canny(gray, 50, 150)

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

for contour in contours:

print(contour) # Print contour coordinates

cv2.drawContours(img, [contour], -1, (0, 255, 0), 2)

cv2.imshow("Contours", img)

cv2.waitKey(0)

find\_contours(r"C:\Users\sandh\Downloads\panda.jpg")

or)

import cv2

def find\_contours(image\_path):

img = cv2.imread(image\_path)

contours, \_ = cv2.findContours(cv2.Canny(cv2.cvtColor(img, cv2.COLOR\_BGR2GRAY), 50, 150), cv2.RETR\_EXTERNAL, cv2.CHAIN\_APPROX\_SIMPLE)

for contour in contours: print(contour)

cv2.drawContours(img, contours, -1, (0, 255, 0), 2)

cv2.imshow("Contours", img)

cv2.waitKey(0)

find\_contours(r"C:\Users\sandh\Downloads\panda.jpg")

al:

Read Image: Load the image using cv2.imread(image\_path).

Convert to Grayscale: Convert the image to grayscale using cv2.cvtColor().

Edge Detection: Apply edge detection with cv2.Canny() on the grayscale image.

Find Contours: Use cv2.findContours() to detect contours from the edge-detected image.

Display Results: Loop through contours, print coordinates, and draw contours using cv2.drawContours() and display the image.

16)

import cv2

import numpy as np

# Load the image in grayscale

image = cv2.imread('input.jpg', cv2.IMREAD\_GRAYSCALE)

# Apply Sobel edge detection

sobel\_x = cv2.Sobel(image, cv2.CV\_64F, 1, 0, ksize=3) # X direction

sobel\_y = cv2.Sobel(image, cv2.CV\_64F, 0, 1, ksize=3) # Y direction

# Combine the X and Y gradients

sobel\_combined = cv2.magnitude(sobel\_x, sobel\_y)

# Display results

cv2.imshow("Original", image)

cv2.imshow("Sobel Edge Detection", sobel\_combined)

cv2.waitKey(0)

cv2.destroyAllWindows()

al:

 **Convert the image to grayscale.**

 **Apply the Sobel operator** in the x-direction to detect vertical edges.

 **Apply the Sobel operator** in the y-direction to detect horizontal edges.

 **Combine the gradients** from both x and y directions to get the overall edge intensity.

 **Display or save the final edge-detected image.**

17)

import cv2

import numpy as np

# Load the image in grayscale

image = cv2.imread('input.jpg', cv2.IMREAD\_GRAYSCALE)

# Define the Roberts cross kernels

roberts\_x = np.array([[1, 0], [0, -1]], dtype=np.float32)

roberts\_y = np.array([[0, 1], [-1, 0]], dtype=np.float32)

# Apply the Roberts kernels

edge\_x = cv2.filter2D(image, -1, roberts\_x)

edge\_y = cv2.filter2D(image, -1, roberts\_y)

# Combine the edges

roberts\_combined = cv2.magnitude(edge\_x, edge\_y)

# Display results

cv2.imshow("Original", image)

cv2.imshow("Roberts Edge Detection", roberts\_combined)

cv2.waitKey(0)

cv2.destroyAllWindows()

al:

 **Load the image in grayscale.**

 **Define the Roberts cross kernels** for x and y directions.

 **Apply convolution** with the kernels to compute edge\_x and edge\_y.

 **Combine the gradients** using magnitude to get roberts\_combined.

 **Display the original and edge-detected images**.

18)

import cv2

import numpy as np

image = cv2.imread('input.jpg', cv2.IMREAD\_GRAYSCALE)

# Prewitt kernels

kx = np.array([[1, 0, -1], [1, 0, -1], [1, 0, -1]], dtype=np.float32)

ky = np.array([[1, 1, 1], [0, 0, 0], [-1, -1, -1]], dtype=np.float32)

# Apply kernels and combine edges

edge\_x = cv2.filter2D(image, -1, kx)

edge\_y = cv2.filter2D(image, -1, ky)

edges = cv2.magnitude(edge\_x, edge\_y)

cv2.imshow("Prewitt Edge Detection", edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

al:

 **Load the image in grayscale.**

 **Define the Prewitt kernels** for horizontal and vertical edges.

 **Apply the kernels** to the image using convolution to get the gradients in both directions.

 **Combine the gradients** by calculating the magnitude of the horizontal and vertical edges.

 **Display the resulting edge-detected image**.

19)

import cv2

# Load the image in grayscale

image = cv2.imread('input.jpg', cv2.IMREAD\_GRAYSCALE)

# Apply Canny edge detection

edges = cv2.Canny(image, 100, 200)

# Display results

cv2.imshow("Canny Edge Detection", edges)

cv2.waitKey(0)

cv2.destroyAllWindows()

al:

 **Convert the image to grayscale.**

 **Apply Gaussian blur** to smooth the image and reduce noise.

 **Compute the gradient** of the image using Sobel operators for edge detection.

 **Apply non-maximum suppression** to thin the edges.

 **Apply double thresholding** to identify strong and weak edges, then perform edge tracking by hysteresis.

20)

import cv2

# Open video

cap = cv2.VideoCapture('input\_video.mp4')

fps = cap.get(cv2.CAP\_PROP\_FPS)

# Capture frames and reverse

frames = []

while cap.isOpened():

ret, frame = cap.read()

if not ret: break

frames.append(frame)

cap.release()

# Write reversed video

out = cv2.VideoWriter('output\_video.mp4', cv2.VideoWriter\_fourcc(\*'mp4v'), fps, (frames[0].shape[1], frames[0].shape[0]))

for frame in reversed(frames):

out.write(frame)

out.release()

al:

1. **Open the video file** and extract its frames.
2. **Store the frames** in a list while reading the video.
3. **Release the video capture** once all frames are collected.
4. **Reverse the order** of the frames.
5. **Write the reversed frames** to a new video file.

21)

import cv2

# Load the face detector

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

# Read image and detect faces

image = cv2.imread('input.jpg')

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

faces = face\_cascade.detectMultiScale(gray)

# Draw rectangles around faces

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

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

# Show the result

cv2.imshow("Faces", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

al:

 **Load the pre-trained face detection model** (Haar Cascade).

 **Read the input image** and convert it to grayscale.

 **Detect faces** using the detectMultiScale() function.

 **Draw rectangles** around the detected faces in the image.

 **Display the image** with the faces highlighted.

22)

import cv2

vehicle\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_car.xml')

cap = cv2.VideoCapture('input\_video.mp4')

while cap.isOpened():

ret, frame = cap.read()

if not ret: break

gray = cv2.cvtColor(frame, cv2.COLOR\_BGR2GRAY)

vehicles = vehicle\_cascade.detectMultiScale(gray)

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

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

cv2.imshow("Vehicle Detection", frame)

if cv2.waitKey(1) & 0xFF == ord('q'): break

cap.release()

cv2.destroyAllWindows()

al:

 **Load the pre-trained vehicle detection model** (Haar Cascade).

 **Open the video** and read frames.

 **Convert each frame to grayscale** for easier processing.

 **Detect vehicles** using the detectMultiScale() method.

 **Draw rectangles around detected vehicles** and display the frame.

23)

import cv2

# Load the pre-trained Haar Cascade Classifier for eye detection

eye\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_eye.xml')

# Read the image

image = cv2.imread('input.jpg')

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

# Detect eyes

eyes = eye\_cascade.detectMultiScale(gray)

# Draw rectangles around the eyes

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

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

# Display the result

cv2.imshow("Eye Detection", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

al:

1. **Load the pre-trained eye detection model** (Haar Cascade).
2. **Read and convert the image to grayscale**.
3. **Detect eyes** using the detectMultiScale() function.
4. **Draw rectangles** around detected eyes in the image.
5. **Display the image** with the highlighted eyes.

24)

import cv2

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

face\_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade\_frontalface\_default.xml')

# Read the image and convert it to grayscale

image = cv2.imread('input.jpg')

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

# Detect faces in the image

faces = face\_cascade.detectMultiScale(gray)

# Draw rectangles around faces and count them

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

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

# Print the number of faces detected

print(f"Number of faces detected: {len(faces)}")

# Display the result

cv2.imshow("Face Detection", image)

cv2.waitKey(0)

cv2.destroyAllWindows()

al:

1. **Load the pre-trained face detection model** (Haar Cascade).
2. **Read and convert the image to grayscale**.
3. **Detect faces** using the detectMultiScale() method.
4. **Draw rectangles** around each detected face.
5. **Count and display the number of detected faces**.

25)

import cv2

img = cv2.imread("C:/Users/divya/Downloads/Girl with a Cat.png")

wm = cv2.imread("C:/Users/divya/OneDrive/Pictures/Saved Pictures/logo.jfif")

h\_wm, w\_wm = wm.shape[:2]

h\_img, w\_img = img.shape[:2]

center\_x = int(w\_img/2)

center\_y = int(h\_img/2)

top\_y = center\_y- int(h\_wm/2)

left\_x = center\_x- int(w\_wm/2)

bottom\_y = top\_y + h\_wm

right\_x = left\_x + w\_wm

roi = img[top\_y:bottom\_y, left\_x:right\_x]

result = cv2.addWeighted(roi, 1, wm, 0.3, 0)

img[top\_y:bottom\_y, left\_x:right\_x] = result

cv2.imshow("Watermarked Image", img)

cv2.waitKey(0)

cv2.destroyAllWindows()

26)

import cv2

import numpy as np

image = cv2.imread("C:/Users/divya/OneDrive/Pictures/Saved Pictures/cat.jfif")

img2 = cv2.imread('C:/Users/divya/OneDrive/Pictures/Saved Pictures/logo.jfif')

print(image.shape) # Print image shape

cv2.imshow("original", image)

imageCopy = image.copy()

cv2.circle(imageCopy, (100, 100), 30, (255, 0, 0),-1) cv2.imshow('image', image)

cv2.imshow('image copy', imageCopy)

cropped\_image = image[80:280, 150:330]

cv2.imshow("cropped", cropped\_image)

cv2.imwrite("Cropped Image.jpg", cropped\_image)

dst = cv2.addWeighted(image, 0.5, img2, 0.7, 0)

img\_arr = np.hstack((image, img2))

cv2.imshow('Input Images',img\_arr)

cv2.imshow('Blended Image',dst)

cv2.waitKey(0)

cv2.destroyAllWindows()